UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

Uvod v programski paket

SPICE OPUS

Árpád Bűrmen

Ljubljana, 2006

Recenzenta: prof.dr. Tadej Tuma, prof.dr. Tomaž Dogša.

Uvod

Učbenik obravnava analizo vezij s programskim paketom SPICE OPUS, ki je zasnovan na enem najuspešnejših programov s tega področja - SPICE3. Namenjen je študentom smeri elektronika, pa tudi ostalim, ki jih tovrstna tematika zanima. Za razumevanje vsebine je potrebno predznanje teorije vezij in osnov uporabe osebnega računalnika. Povod za izdajo učbenika je bila želja študentov, da bi imeli zbrano celotno snov, ki se obravnava na laboratorijskih vajah pri predmetu Računalniško načrtovanje vezij, pregledno zbrano na enem mestu.

Posebej izpostavljeno je praktično delo s paketom SPICE OPUS. Izpuščeni so opisi modelov, saj jih zainteresirani bralec zlahka najde v literaturi. Učbenik se začne s kratkim uvodom v okolje SPICE OPUS in zgledom uporabe. Sledita poglavji o opisu vezja in analizah, ki jih paket ponuja. Zadnje poglavje je namenjeno programskemu jeziku NUTMEG, ki služi avtomatizaciji postopka zahtevnejših analiz in se uporablja tudi pri optimizaciji vezij.

Veliko dodatnega materiala je na voljo na spletni strani http://fides.fe.uni-lj.si/spice, kjer je dostopna tudi najnovejša inačica paketa. Na strani so posebej izpostavljene razlike med prvotnim programom SPICE3 in paketom SPICE OPUS. Učbenik se nanaša na inačico SPICE OPUSa, ki je bila zgrajena oktobra 2006 in je priložena na CD-ROMu.

Zahvaljujem se vsem, ki so s svojimi pripombami prispevali k prvi izdaji tega učbenika. Za vsa opozorila na napake bom še naprej hvaležen. Za morebitne napake, ki so še ostale, se že vnaprej opravičujem. Vsem bralcem želim veliko uspeha pri delu s programi iz družine SPICE.

Árpád Bűrmen

Kazalo

1

Okolje SPICE OPUS

Eden najbolj uspešnih brezplačnih programov za analizo vezij je program SPICE. Veliko komercialnih simulatorjev temelji na prvotni izvorni kodi simulatorjev SPICE2 [1] in SPICE3 [2], ki sta bila razvita v ZDA na univerzi v Berkeleyu. Za simulacijo vezij z diskretnimi komponentami sta zelo popularna PSPICE [4] in IsSpice [3]. HSPICE [5] je namenjen predvsem simulaciji integriranih vezij. Obstajajo tudi simulatorji, ki so napisani na novo, kot na primer Spectre [6].

Okolje SPICE OPUS [15] je izpeljanka simulatorja SPICE3. Prva inačica SPICE OPUSa je bila namenjena okolju Microsoft Windows. Razvoja okolja je šel najprej v smeri odstranjevanja hroščev, ki so povzročali, da se je program sesul po večjem številu analiz. Na ta način je simulator postal primeren za optimizacijo vezij, kjer je potrebno izvršiti po več tisoč analiz.

Zaradi precejšnjega interesa uporabnikov se je kmalu pojavila tudi inačica za okolje Linux. Ob tej priložnosti se je začel razvoj novega uporabniškega vmesnika, ki je še vedno v uporabi. Vmesnik temelji na knjižnici Qt. Zaradi uporabe te knjižnice sta obe inačici že na videz zelo podobni. Med inačicama se lahko pojavljajo minimalne razlike v rezultatih zaradi različnega obnašanja sistemskih numeričnih podprogramov pod obema operacijskima sistemoma.

SPICE OPUS ne ponuja programa za shematski vnos vezij, saj je le teh dovolj na razpolago na svetovnem spletu. Boljše med tovrstnimi programi se da z minimalno količino truda prilagoditi tako, da lahko izvozijo vezje v obliki, ki jo razume SPICE OPUS. Namen paketa SPICE OPUS je predvsem ponuditi zanesljiv in zmogljiv simulator vezij tako za študente, kot tudi za profesionalne uporabnike.

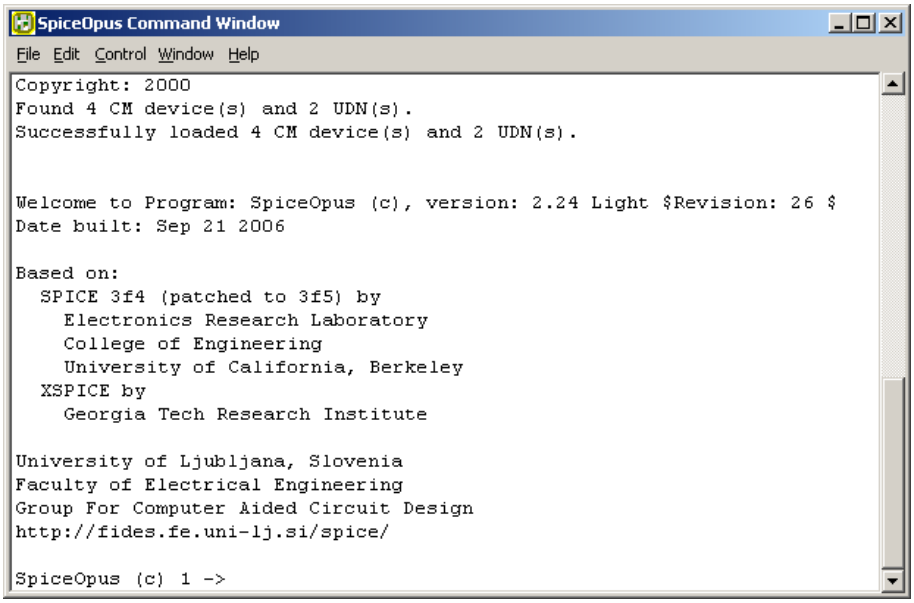
Simulatorju vezje opišemo s takoimenovano vhodno datoteko, ki jo lahko napišemo s katerimkoli urejevalnikom besedil ali pa ustvarimo s programom za shematski vnos vezij. Okolje je namenjeno interaktivnemu delu. To pomeni, da vse funkcije programa nadzorujemo z ukazi, ki jih tipkamo v ukazno okno. Jezik, v katerem vnašamo ukaze se imenuje NUTMEG.

Po zagonu programa vhodno datoteko z ustreznim ukazom naložimo. Z ukazi nato zaganjamo analize ter obdelujemo in pregledujemo rezultate. Okolje ponuja tudi možnost izrisovanja grafov. Možno je spreminjati parametre elementov vezja. Na spremenjenem vezju lahko ponovimo analize in tako ugotovimo vpliv spremembe parametrov na obnašanje vezja. Zbirka ukazov (program) je lahko zapisana tudi v vhodni datoteki (ukazni blok `.control`). Tak program se izvrši vsakič, ko naložimo vezje.

2

Zgled uporabe

Program zaženemo s klikom na ustrezno ikono v meniju Start (sistem Windows) oziroma z ukazom `spiceopus` (sistem Linux). Po zagonu programa se znajdemo v ukaznem oknu (SpiceOpus Command Window, slika 2.1), kjer tipkamo ukaze na katere nato dobivamo odgovore od okolja SPICE OPUS. V oknu lahko poljubno besedilo označimo z miško. Besedilo skopiramo na odložišče s kombinacijo tipk **CTRL+C** ali s klikom na izbiro **Edit/Copy** v glavnem meniju. Besedilo z odložišča prenesemo v ukazno vrstico s pritiskom **CTRL+V** oziroma izbiro **Edit/Paste**. Izbira **Edit/Select All** (**CTRL+A**) nam označi celotno besedilo v ukaznem oknu. Ker se v ukaznem oknu lahko nabere veliko vsebine, ki nas več ne zanima, se le te lahko znebimo z izbiro **Edit/Clear Terminal History** (**CTRL+T**). Vsebino ukaznega okna natisnemo z izbiro **File/Print** (**CTRL+P**).



Slika 2.1: Ukazno okno SPICE OPUS

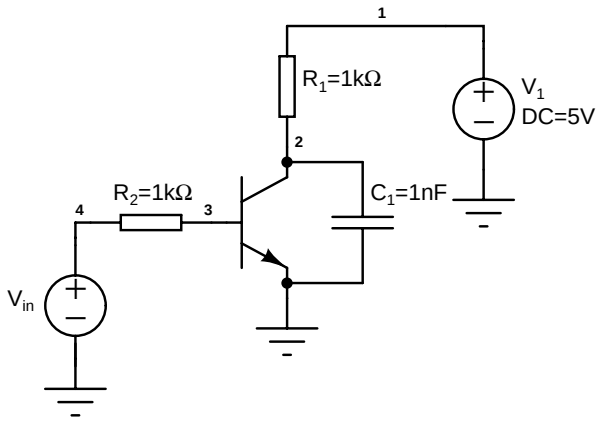
Pri tipkanju ukazov v ukaznem oknu lahko uporabimo kurzorski tipki ← in → za premikanje po natipkanem ukazu. Funkcijo brisanja znaka pred in za kurzorjem imata tipki **Backspace** in **Delete**. Če želimo zbrisati celoten natipkan ukaz, pritisnemo tipko **Esc**. Ukazno okno si zapomni celotno zgodovino vtipkanih ukazov, med katerimi lahko izbiramo s kurzorskima tipkama ↑ in ↓. Tipki **Page Up** in **Page Down** nam omogočata premikanje navzgor in navzdol po vsebini ukaznega okna.

SPICE OPUS nam ponuja možnost beleženja vsebine ukaznega okna. Le to vklopimo tako, da program zaženemo z opcijo `-o imeDatoteke`. Zapisovanje v datoteko začasno ustavimo, oziroma spet poženemo, z izbiro

Edit/Logging.

Med delom se lahko odpre veliko število grafičnih oken. Le ta najhitreje zapremo z izbiro **Window/Close All Plot Windows**.

2.1 Zgled vezja



Slika 2.2: Preprosto vezje

Na sliki 2.2 je preprost tranzistorski ojačevalnik. Opišemo ga s tekstovno datoteko, katere format je precej podoben pri vseh simulatorjih iz družine SPICE.

```
Moje prvo vezje
v1 (1 0) dc=5
r1 (1 2) r=1k
c1 (2 0) c=1n
q1 (2 3 0) 2n2222
rin (3 4) r=1k
vin (4 0) dc=0.68 acmag=1 pulse=(0.66 0.76 5u 1n)
* Tu se zacnejo modeli
.model 2n2222 npn
+ is=19f bf=150 vaf=100 ikf=0.18 ise=50p
+ ne=2.5 br=7.5 var=6.4 ikr=12m isc=8.7p
+ nc=1.2 rb=50 re=0.4 rc=0.3 cje=26p tf=0.5n
+ cjc=11p tr=7n xtb=1.5 kf=0.032f af=1
.end
```

Prva vrstica opisa je ime vezja (`Moje prvo vezje`). Sledijo opisi elementov, pri čemer je vsak element podan v svoji vrstici. Tako na primer vrstica `r1...` podaja upor z imenom `r1`, ki je med vozliščema `1` in `2`, njegova upornost pa znaša 1kΩ. Vozlišče `0` je rezervirano za maso. Njegov potencial je vedno 0V. Vse vrstice, ki se začnejo z zvezdico, so komentarji. Dodatna razlaga je potrebna za napetostni vir `vin`. Parameter `pulse` pove, da je to pulzni vir, katerega vrednost se spremeni iz 0.66V v 0.76V ob času 5μs s prehodom, ki traja 1ns. Parameter `ac` se uporablja v malosignalni analizi in pomeni, da bo pri tej analizi vir predstavljen s kompleksorjem amplitude 1V in faze 0°.

Poseben pomen ima vrstica `.model...`, kjer je podan opis tranzistorja 2n2222. Kot prvo opazimo, da je opis modela razdeljen na več vrstic. Če želimo kako vrstico pripojiti k prejšnji, jo začnemo z znakom `+`. V opisu je podano ime modela (`2n2222`), tip modela (`npn`) in parametri modela (na primer parameter `bf`, ki pomeni

tokovno ojačenje tranzistorja. Model je abstrakten opis tranzistorja, konkreten tranzistor pa je podan v vrstici `q1...` iz katere izvemo, da je `q1` bipolaren tranzistor, ki uporablja parametre modela `2n2222`, in je priklopljen med vozlišča `2` (kolektor), `3` (baza) in `0` (emitor). V opisu vezja bi lahko imeli več tranzistorjev (na primer `q1` in `q2`), ki bi vsi uporabljali model `2n2222`.

2.2 Nalaganje vezja in analiza delovne točke

Opis vezja na sliki 2.2 shranimo v datoteko `primer1`. Zaženemo okolje SPICE OPUS in vtipkamo `source primer1`. Na ukaz dobimo odgovor

```
SpiceOpus (c) 2 ->
```

To pomeni, da se je vezje naložilo in da SPICE čaka na naš naslednji ukaz.

Izračunajmo sedaj enosmerno delovno točko našega vezja. Vtipkamo `op` in dobimo

```
SpiceOpus (c) 2 -> op
```

Analize je konec v delčku sekunde, saj je naše vezje zelo preprosto. Vrednosti neodvisnih virov pri tej analizi določa parameter `dc`. Tako ima vir `v1` vrednost 5V, vir `vin` pa 0.68V. Poglejmo si sedaj napetost vozlišča `2`. V ta namen uporabimo ukaz `print`.

```
SpiceOpus (c) 3 -> print v(2)
v(2) = 3.663130e+000
```

Oglejmo si še tokovno porabo vezja tako, da odčitamo tok, ki teče skozi `v1`.

```
SpiceOpus (c) 4 -> print i(v1)
i(v1) = -1.33687e-003
```

Dobimo negativno vrednost za tok, ker SPICE računa, da je pozitiven tisti tok, ki teče v + sponko napetostnega vira. Omeniti velja še, da toka skozi upor `r2` ne moremo izpisati z ukazom `print i(r2)`. SPICE namreč (če tega ne povemo drugače) izračuna samo vozliščne napetosti in tokove skozi neodvisne napetostne vire. Če želimo pomeriti še kak drug tok, je najbolj splošen način, da dodamo v vejo, katere tok želimo meriti, neodvisen napetostni vir oblike

```
vmerim (vozl1 vozl2) dc=0
```

Njegov tok dobimo z ukazom `print i(vmerim)`. Tak vir nima vpliva na vezje, povroči pa, da SPICE izračuna njegov tok. Ničelni napetostni vir se torej obnaša kot idealen ampermeter. Opozoriti velja, da ima lahko samo en element vezja ime `vmerim`. Za vajo na ta način izmerite tok, ki teče iz emitorja `q1`. Ime notranjega vozlišča, ki nastane zaradi vrivanja ničelnega napetostnega vira, si lahko kar izmislite (recimo `30`).

2.3 Odvisnost delovne točke od napetosti napetostnega vira in grafično okno

Zanima nas, kako se spreminja delovna točka vezja, če se spreminja enosmerna napetost vira `vin` v območju od 0V do 1.4V s korakom 0.01V. V ta namen vtipkajmo

```
SpiceOpus (c) 5 -> dc vin 0 1.4 0.01V
```

Analize je spet hitro konec. Na videz se ni nič zgodilo. Pa si poglejmo, kaj smo dobili. Natipkajmo

```
SpiceOpus (c) 6 -> display
Here are the vectors currently active:

Title: Preprosto vezje
Name: dc1 (DC transfer characteristic)
Date: Fri Jul 16 10:53:49  2004

    V(1)                : voltage, real, 140 long
    V(2)                : voltage, real, 140 long
    V(3)                : voltage, real, 140 long
    V(4)                : voltage, real, 140 long
    q1#base             : voltage, real, 140 long
    q1#collector        : voltage, real, 140 long
    q1#emitter          : voltage, real, 140 long
    sweep               : voltage, real, 140 long [default scale]
    v1#branch           : current, real, 140 long
    vin#branch          : current, real, 140 long
```

Vidimo, da se je izračunala skupina rezultatov (vektorjev) z imenom `dc1`. Vektorji v tej skupini so realni in vsebujejo vsak po 140 vrednosti. Vektor `sweep` hrani skalo (vrednosti vira, za katere je bila analizirana delovna točka). To je privzeti vektor v tej skupini, kar razpoznamo po napisu `default scale`. Vektorji `v(...)` hranijo vozliščne napetosti za vse vrednosti vira `vin` iz vektorja `sweep`. Vektorji v*ime*`#branch` hranijo tokove, ki tečejo v pozitivne sponke napetostnih virov. Do teh vektorjev lahko pridemo tudi s sintakso `i(...)`. Vektorji `q1#base`, `q1#collector` in `q1#emitter` predstavljajo vozliščne napetosti internih vozlišč tranzistorja `q1`. Ta vozlišča za nas nimajo kakega večjega pomena.

Tabelirajmo vrednosti vektorja `v(2)`.

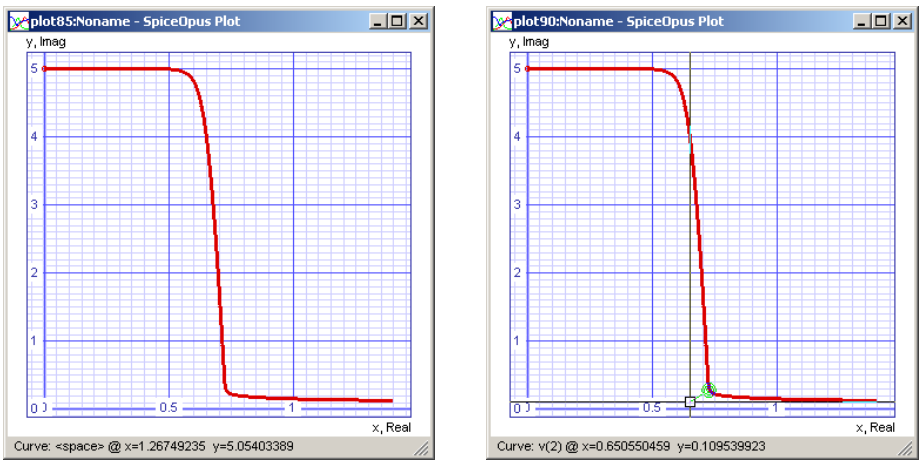
```
SpiceOpus (c) 7 -> print v(2)
                 Preprosto vezje
                 DC transfer characteristic  Fri Jul 16 10:53:49  2004
------------------------------------------------------------------------
Index   sweep           v(2)
------------------------------------------------------------------------
0  0.000000e+000  5.000000e+000
1  1.000000e-002  5.000000e+000
2  2.000000e-002  5.000000e+000
3  3.000000e-002  5.000000e+000
4  4.000000e-002  5.000000e+000
5  5.000000e-002  5.000000e+000
6  6.000000e-002  5.000000e+000
7  7.000000e-002  5.000000e+000
8  8.000000e-002  5.000000e+000
9  9.000000e-002  5.000000e+000
...
```

V tabeli se izpišejo tudi pripadajoče vrednosti indeksa in privzetega vektorja.

Narišimo sedaj odvisnost izhodne napetosti v vozlišču 2 od vrednosti vira `vin`.

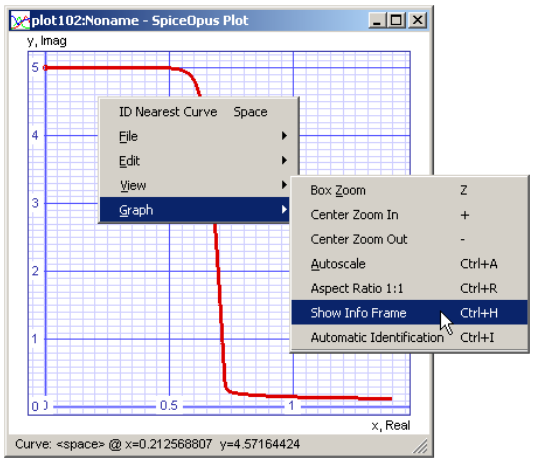
```
SpiceOpus (c) 8 -> plot v(2)
```

Pojavi se grafično okno (slika 2.3, levo) s krivuljo odvisnosti napetosti `v(2)` od privzetega vektorja `sweep` (`sweep` hrani enosmerne vrednosti vira `vin`). Spodnja vrstica grafičnega okna prikazuje koordinate kurzorja. Ime kurzorju najbližje krivulje dobimo tako, da ji primaknemo kurzor in pritisneto preslednico (**Space**). Ime krivulje se pojavi na levem koncu spodnje vrstice okna (slika 2.3, desno). Od kurzorja do krivulje poteka zelena črta. Ko premaknemo kurzor, črta izgine in z njo tudi ime krivulje.



Slika 2.3: Levo: grafično okno SPICE OPUS. Desno: izpis imena krivulje.

Če kliknemo z desno tipko na površino grafa, dobimo meni (slika 2.4). Meni vsebuje naslednje opcije:



Slika 2.4: Meni, ki ga dobimo z desnim klikom na grafično okno.

ID Nearest Curve (**Space**) izpiše ime krivulje, ki je najbližja kurzorju.

File/Print (**CTRL+P**) natisne graf.

File/Close (**ALT+F4**) zapre okno z grafom.

File/Close All Plots zapre vsa okna z grafi.

Edit/Copy (**CTRL+C**) kopira označeno besedilo na odložišče. Besedilo lahko označimo v merilnem oknu.

Edit/Freeze Messages (**CTRL+F**) zamrzne/odmrzne merilno okno, da lahko v njem označimo besedilo in ga skopiramo.

Edit/Clear Marker (**Esc**) izbriše referenčno točko (marker), ki jo dobimo z levim klikom na graf.

View/Box Zoom (**Z**) vklopi način v katerem lahko označimo del grafa, ki ga želimo povečati. To naredimo tako, da gremo v en vogal območja, pritisnemo in držimo levo tipko miške in se pomaknemo v diagonalno nasproten vogal. Ko levo tipko spustimo, se označeno območje poveča.

View/Center Zoom In (**+**) poveča območje grafa okoli kurzorja.

View/Center Zoom Out (**-**) pomanjša območje grafa v okolici kurzorja.

View/Autoscale (**CTRL+A**) samodejno določi območje grafa tako, da so v celoti vidne vse krivulje.

View/Aspect Ratio 1:1 (**CTRL+R**) vklopi/izklopi način prikazovanja, v katerem je razmerje med enotami na obeh koordinatnih oseh 1:1. Če opcijo vklopimo, so krogi res krogi in ne elipse.

View/Show Info Frame (**CTRL+H**) vklopi/izklopi merilno okno v spodnji polovici grafičnega okna.

View/Automatic Identification (**CTRL+I**) vklopi/izklopi samodejno identifikacijo krivulj. Če je ta opcija vklopljena, se po krajšem času mirovanja kurzorja samodejno izvrši opcija **ID Nearest Curve**. Ker zna biti postopek pri večjem številu krivulj z veliko točkami zamuden je včasih priporočljivo to opcijo izklopiti.

Graph/Curve Representation vsebuje opcije za nastavitev načina risanja krivulj **Points** (izrišejo se samo točke), **Line** (izrišejo se črte med točkami) in **Comb** (izrišejo se črte od x-osi do točk krivulj) ter opcije za nastavitev debeline črt oziroma velikosti točk (ta je lahko 0, 1, 2 ali 3).

Graph/Grid/X Log Scale (for x-y grid) vklopi/izklopi logaritemsko skalo na x-osi v x-y načinu prikazovanja.

Graph/Grid/Y Log Scale (for x-y grid) vklopi/izklopi logaritemsko skalo na y-osi v x-y načinu prikazovanja.

Graph/Grid/x-y Grid vklopi x-y način prikazovanja s pravokotno mrežo črt.

Graph/Grid/Polar Grid vklopi način prikazovanja s polarno mrežo.

Graph/Grid/Smith Z Grid vklopi način prikazovanja s Smithovo impedančno mrežo.

Graph/Grid/Smith Y Grid vklopi način prikazovanja s Smithovo admitančno mrežo.

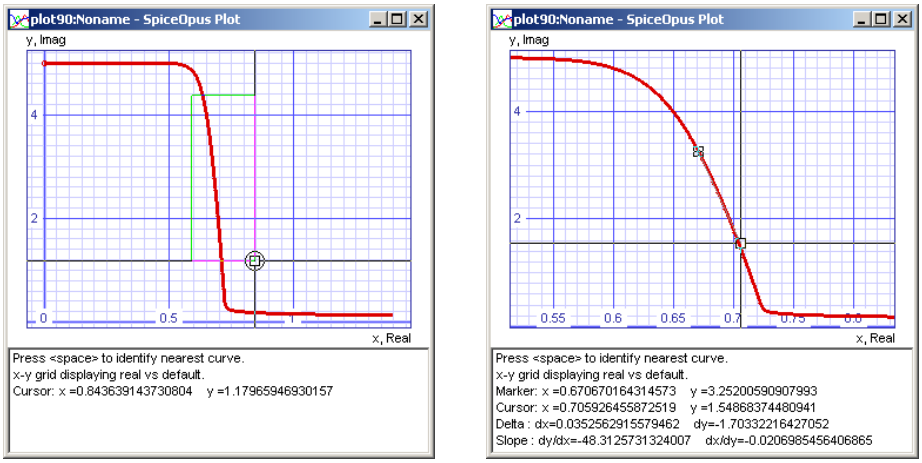
Ostale opcije se nanašajo na prikazovanje kompleksnih vektorjev.

Vklopimo sedaj merilno okno s pritiskom na **CTRL+H** ali v meniju (slika 2.4). Povečajmo strmo padajoči del grafa (slika 2.5, levo). To naredimo s pritiskom na **CTRL+Z**, kar povzroči, da se kurzor spremeni (pojavi se krožec, ki nas poziva, da označimo območje, ki ga želimo povečati). Pomaknimo se v en vogal območja, pritisnemo in držimo levo tipko miške, se pomaknemo v diagonalno nasproten vogal območja in sputimo tipko miške. Ko spustimo tipko, se označeno območje poveča in zavzame celotno površino grafa.

Referenčno točko (marker) izberemo s klikom na krivuljo (slika 2.5, desno). Na označenem mestu se pojavi kvadrat od katerega se vleče črta do kurzorja. Referenčno točko izbrišemo s pritiskom na **Esc**. V merilnem oknu se prikazujejo koordinate referenčne točke (marker x, y), kurzorja (cursor x, y), razdalja med referenčno točko in kurzorjem (delta, dx, dy) in strmina črte, ki povezuje referenčno točko s kurzorjem (slope, dy/dx, dx/dy). Če sedaj črto, ki povezuje referenčno točko in kurzor postavimo vzporedno s tangento na krivuljo v referenčni točki, dobimo pod oznako dy/dx strmino, ki predstavlja napetostno ojačenje ojačevalnika na sliki 2.2.

2.4 Malosignalna analiza in kompleksni grafi

Pri malosignalni analizi se določi najprej delovna toča vezja na enak način, kot pri `op` analizi. Nato se vezje linearizira. V lineariziranem vezju imamo tudi reaktance (kapacitivnosti in induktivnosti). Analiza poteka v frekvenčnem prostoru (kapacitivnosti in induktivnosti se modelirajo z admitancami velikosti $\mathrm{i}\omega C$ in $1/(\mathrm{i}\omega L)$, $\omega = 2\pi f$). Neodvisne vire predstavljajo kompleksorji $acmag \cdot \mathrm{e}^{\mathrm{i}\cdot\pi\cdot acphase/180}$. Ker parameter *acmag* ni podan za vir `v1`, se uporabi privzeta vrednost 0, ki pomeni, da bo pri analizi v frekvenčnem prostoru vir predstavljal kratek



Slika 2.5: Levo: izbira območja, ki ga bomo povečali. Desno: merilno okno.

stik. Vir `vin` ima vrednost parametra *acmag* enako 1, za fazo pa se uporabi vrednost 0, ker parameter *acphase* ni podan. Ta vir predstavlja vzbujanje pri analizi v frekvenčnem prostoru. Recimo, da nas zanima obnašanje vezja v frekvenčnem območju od 100Hz do 1GHz. Naredili bomo malosignalno (`ac`) analizo, ki bo to območje preletela logaritemsko in na vsako dekado izračunala odziv vezja v 10 točkah.

```
SpiceOpus (c) 9 -> ac dec 10 100 1G
```

Po končani analizi si poglejmo rezultate.

```
SpiceOpus (c) 10 -> display
Here are the vectors currently active:

Title: Preprosto vezje
Name: ac1 (AC Analysis)
Date: Fri Jul 16 09:13:28  2004

    V(1)                : voltage, complex, 71 long
    V(2)                : voltage, complex, 71 long
    V(3)                : voltage, complex, 71 long
    V(4)                : voltage, complex, 71 long
    frequency           : frequency, complex, 71 long, grid = xlog [default scale]
    q1#base             : voltage, complex, 71 long
    q1#collector        : voltage, complex, 71 long
    q1#emitter          : voltage, complex, 71 long
    v1#branch           : current, complex, 71 long
    vin#branch          : current, complex, 71 long
```

Vidimo, da so sedaj vektorji kompleksni, privzeti vektor pa se imenuje `frequency` in ima logaritemsko skalo (`grid = xlog`). Frekvenca je sicer kompleksna, vendar je imaginarni del zmeraj enak 0. Vsaka točka vektorjev predstavlja kompleksor, ki predstavlja odziv vezja pri pripadajoči frekvenci. Izpišimo odziv vezja (kompleksor vozliščne napetosti `v(2)`):

```
SpiceOpus (c) 11 -> print v(2)
                                Preprosto vezje
                                AC Analysis  Fri Jul 16 09:13:28  2004
--------------------------------------------------------------------------
```

```
Index   frequency                       v(2)
-------------------------------------------------------------------------
0  1.000000e+002, 0.000000e+000 -4.720902e+001, 1.214921e-002
1  1.258925e+002, 0.000000e+000 -4.720901e+001, 1.529495e-002
2  1.584893e+002, 0.000000e+000 -4.720901e+001, 1.925520e-002
3  1.995262e+002, 0.000000e+000 -4.720901e+001, 2.424085e-002
4  2.511886e+002, 0.000000e+000 -4.720900e+001, 3.051742e-002
5  3.162278e+002, 0.000000e+000 -4.720899e+001, 3.841915e-002
...
```

Vidimo, da je privzeti vektor kompleksen in da so imaginarni deli frekvenc enaki 0.

Ker je kompleksor napetosti na vhodu (`v(4)`) enak 1 za vse frekvence (določa ga neodvisen napetostni vir `vin`), je ojačenje od vhoda na vozlišču `4` do izhoda na vozlišču `2` enako kar kompleksorju `v(2)`. V nasprotnem primeru bi morali deliti oba kompleksorja (`v(2)/v(4)`), da bi dobili napetostno ojačenje.

Izpišimo še absolutno vrednost (v decibelih) in fazo napetostnega ojačenja. V ta namen najprej preklopimo kotne enote na stopinje.

```
SpiceOpus (c) 12 -> set units=degrees
SpiceOpus (c) 13 -> print db(v(2)) ph(v(2))
                              Preprosto vezje
                              AC Analysis  Fri Jul 16 11:02:55  2004
-------------------------------------------------------------------------
Index   frequency                       db(v(2))        ph(v(2))
-------------------------------------------------------------------------
0  1.000000e+002, 0.000000e+000  3.348050e+001  1.799853e+002
1  1.258925e+002, 0.000000e+000  3.348050e+001  1.799814e+002
2  1.584893e+002, 0.000000e+000  3.348050e+001  1.799766e+002
3  1.995262e+002, 0.000000e+000  3.348050e+001  1.799706e+002
4  2.511886e+002, 0.000000e+000  3.348050e+001  1.799630e+002
5  3.162278e+002, 0.000000e+000  3.348050e+001  1.799534e+002
...
```

Narišimo sedaj amplitudno in fazno karakteristiko (slika 2.6).

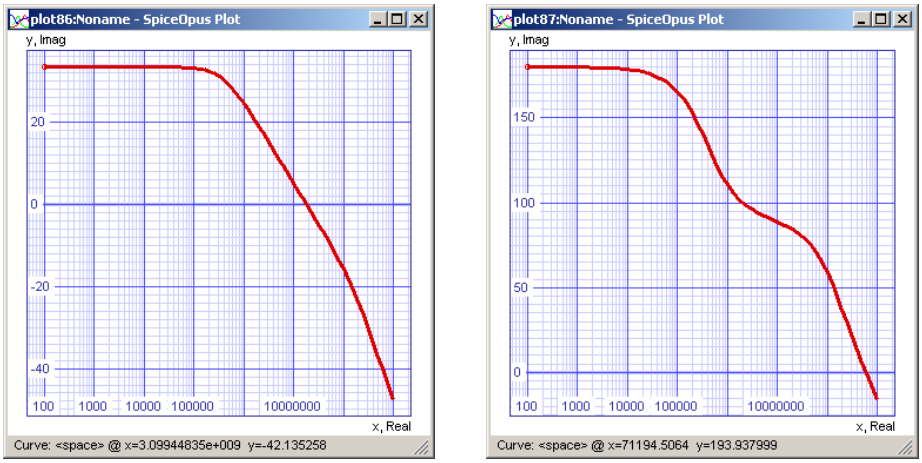
```
SpiceOpus (c) 14 -> plot db(v(2))
SpiceOpus (c) 15 -> plot ph(v(2))
```

Narišimo še Nyquistov diagram (slika 2.7).

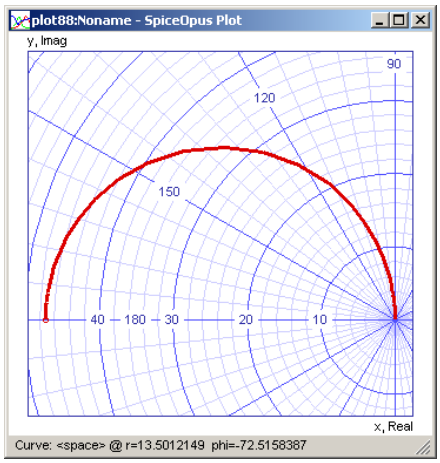
```
SpiceOpus (c) 16 -> plot mode cx v(2)/v(4) polar
```

Tokrat je izrisani vektor kompleksen. Polarno mrežo dobimo tako, da v ukazu `plot` dodamo besedo `polar`. Opcija `mode cx` pove, da se naj vektor tolmači kot kompleksen. Pri tem se realni del nanaša na os x, imaginarni del pa na os y. V meniju, ki ga dobimo z desnim klikom na graf in nato na podmeni **Graph**, je še nekaj opcij, ki jih nismo opisali. Te opcije pridejo prav, kadar rišemo kompleksne vektorje.

Opcije **Real (real)**, **Imaginary (imag)**, **Magnitude (mag)** in **Phase (phase)** izberejo prikazovanje realnega dela, imaginarnega dela, absolutne vrednosti in faze kompleksorja. Z opcijama **Units/Angle: rad** in **Units/Angle: deg** izbiramo enote za prikazovanje faze v načinu **Phase**. Opcije **Units/Magnitude: normal**, **Units/Magnitude: dB (20 log10)** in **Units/Magnitude: dB (10 log10)** izbiramo enote za prikazovanje absolutne vrednosti. Prva prikazuje samo absolutno vrednost, druga decibele ojačenja toka oziroma napetosti in tretja decibele ojačenja moči.



Slika 2.6: Amplitudna karakteristika (v decibelih) in fazna karakteristika (v stopinjah) ojačevalnika.



Slika 2.7: Nyquistov diagram ojačevalnika.

Opcije **R (realz)**, **X (imagz)**, **G (realy)** in **B (imagy)** vklopijo prikazovanje normirane rezistance, reaktance, konduktance in susceptance, pri čemer se kompleksne vrednosti podanega vektorja tolmačijo kot odbojnosti.

Opciji **Complex (cx), x-y grid** in **Complex (cx), polar grid** prikazujeta točke na kompleksni ravnini. Prva nariše pravokotno mrežo, druga pa polarno mrežo.

Opciji **Complex (cx), Smith Z grid** in **Complex (cx), Smith Y grid** narišeta Smithovo impedančno in admitančno mrežo. Pri risanju krivulje predstavljata realni in imaginarni del podanega vektorja koordinati x in y v kompleksni ravnini in hkrati tudi kompleksno odbojnost.

Imena navedena v oklepajih zgoraj omenjenih opcij se uporabljajo pri ukazu `plot` z opcijo `mode`. Tako smo pri risanju Nyquistovega diagrama uporabili opcijo `mode cx`, ki izbere prikazovanje kompleksnih vektojev z realnim delom na osi x in imaginarnim delom na osi y.

2.5 Analiza v časovnem prostoru

Pri analizi v časovnem prostoru določa časovni potek neodvisnega vira `vin` parameter `pulse`, ki nastavi pulz z začetkom pri 0.68V in vrhom pri 0.78V, ki se začne ob času 5μs in ima dvižni čas 1ns. Vir `v1` nima podanega

časovnega poteka, zato je njegova vrednost konstanta (5V). Pred simulacijo v časovnem prostoru se določi delovna točka vezja upoštevajoč vrednosti neodvisnih virov ob času 0. Ta delovna točka služi kot začetno stanje vezja. Naredimo sedaj analizo v časovnem prostoru od 0s do 10μs s korakom 10ns.

```
SpiceOpus (c) 17 -> tran 10n 10u
```

Poglejmo si, kaj smo dobili.

```
SpiceOpus (c) 18 -> display
Here are the vectors currently active:

Title: Preprosto vezje
Name: tran1 (Transient Analysis)
Date: Fri Jul 16 11:02:55  2004

    V(1)                : voltage, real, 212 long
    V(2)                : voltage, real, 212 long
    V(3)                : voltage, real, 212 long
    V(4)                : voltage, real, 212 long
    q1#base             : voltage, real, 212 long
    q1#collector        : voltage, real, 212 long
    q1#emitter          : voltage, real, 212 long
    time                : time, real, 212 long [default scale]
    v1#branch           : current, real, 212 long
    vin#branch          : current, real, 212 long
```

Tokrat je privzetemu vektorju ime `time` in hrani časovne točke, ki jim pripadajo izračunane vrednosti v vezju. Izpišimo časovni potek vektorjev `v(4)` in `v(2)`.

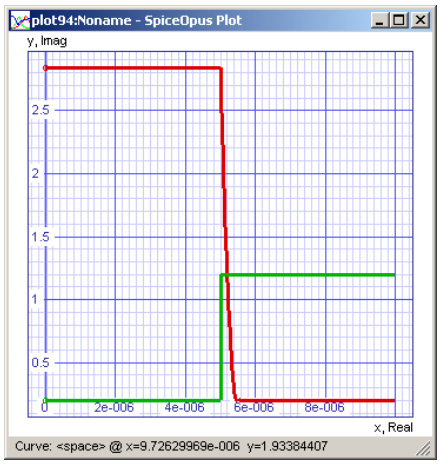
```
SpiceOpus (c) 19 -> print v(4) v(2)
                        Preprosto vezje
                         Transient Analysis  Fri Jul 16 11:02:55  2004
--------------------------------------------------------------------------
Index   time            v(4)            v(2)
--------------------------------------------------------------------------
0  0.000000e+000  6.800000e-001  2.837273e+000
1  2.500000e-009  6.800000e-001  2.837335e+000
2  5.000000e-009  6.800000e-001  2.837335e+000
3  1.500000e-008  6.800000e-001  2.837327e+000
...
103  5.000000e-006  6.800000e-001  2.837113e+000
104  5.000250e-006  7.050000e-001  2.837288e+000
105  5.000500e-006  7.300000e-001  2.837066e+000
106  5.000750e-006  7.550000e-001  2.836595e+000
107  5.001000e-006  7.800000e-001  2.835752e+000
108  5.001250e-006  7.800000e-001  2.834272e+000
109  5.002250e-006  7.800000e-001  2.826255e+000
110  5.006250e-006  7.800000e-001  2.782055e+000
111  5.022250e-006  7.800000e-001  2.603403e+000
112  5.072250e-006  7.800000e-001  2.102254e+000
...
```

Časovni korak ni ravno tak, kot smo ga nastavili. SPICE OPUS ima vgrajene dokaj zapletene algoritme za njegovo spreminjanje. Vidimo tudi, da ko se pojavi hitra sprememba (kot je začetek pulza), se časovni korak prilagodi spremembi in zato zmanjša. Ko je spremembe konec (ob 5001ns) se korak začne spet večati.

Narišimo časovni potek vhodne in izhodne napetosti (slika 2.8).

```
SpiceOpus (c) 20 -> plot v(2) (v(4)-0.66)*10
```

Potek vhodne napetosti smo skalirali tako, da je na grafu viden.



Slika 2.8: Časovni potek izhoda (rdeče) in vhoda (zeleno).

2.6 Zagonske opcije

Ob zagonu lahko programu podamo opcije, ki določajo njegovo delovanje. Za opcijami lahko navedem še ime datoteke, ki jo naj SPICE OPUS prebere po končanem zagonu. Sintaksa je torej:
`spice3 [`*opcija1 opcija2* `...]` *ime_datoteke*

Na voljo so naslednje opcije:

`-b`
Paketni način delovanja. Po nalaganju datoteke in končanem izvajanju ukazov iz ukaznega bloka program samodejno konča z delom.

`-c`
Konzolni način delovanja. Pod sistemom Windows se odpre novo konzolno okno. Pod sistemi LINUX se program izvaja v konzoli iz katere je bil zagnan. V tem načinu ni podpore za grafiko (ukaz `plot`, ...).

`-g`
Okenski način delovanja. Nasprotje opcije `-c`. Ta opcija je privzeta.

`-gf` *velikost*
Nastavi velikost znakov v oknu. Učinek ima samo v okenskem načinu.

`-o` *ime*
Nastavi ime datoteke v katero se shranjujejo izpisana sporočila. V konzolnem načinu delovanja opcija povzroči, da se sporočila ne izpisujejo na zaslon.

`-n`
Ne preberi datoteke `spinit` ob zagonu.

`-r` *ime*
Nastavi privzeto ime za shranjevanje skupin rezultatov v ukazu `write`.

`-pl` *ime*
Nastavi pot do mape `lib`, kjer se nahaja datoteka `spinit`.

`-pw` *ime*
Nastavi pot kamor se SPICE OPUS premakne takoj po končanem zagonu. Opcija je ekvivalent ukaza `cd` *ime* vtipkanega takoj po zagonu.

3

Opis vezja

3.1 Zgradba, zapis števil in vzporedna vezava enakih elementov

Opis vezja je tekstovna datoteka, katere zgradba je

```
Ime vezja
...
.end
```

Pri opisu vezja SPICE ne loči med velikimi in malimi črkami. Med imenom vezja in besedo `.end` posamezne vrstice opisujejo elemente in njihove modele. Elementi so na primer `100k` upor `R1`, dioda `D1` tipa `1N4148`, tranzistor `Q1` tipa `2N2222`, itd. Primeri modelov so model tranzistorja `2N2222`, model diode `1N4148`, itd. Medtem ko vsak dodan element poveča vezje, dodani modeli ne spremenijo vezja samega. V večini primerov (to velja predvsem za polprevodniške elemente) moramo za vsak element vezja (kot na primer tranzistor `Q1`) navesti model, ki ga ta element uporablja. V modelu so definirane vrednosti parametrov, kot je npr. tokovno ojačenje. Več elementov (npr. tranzistorji `Q1`, `Q2` in `Q3`) lahko uporablja en model (npr. `2N2222`). Imena modelov so lahko poljubna zaporedja črk, številk in podčrtajev.

Elemente priključimo med vozlišča. Imena vozlišč so nizi iz črk, cifer in podčrtajev. Če se ime vozlišča začne s cifro, lahko vsebuje samo cifre, če pa se začne s črko ali podčrtajem, lahko vsebuje poljubno zaporedje črk, cifer in podčrtajev. Poseben pomen ima vozlišče `0`, ki predstavlja maso. Potencial tega vozlišča je zmeraj 0V. V vsakem vezju mora biti definirana masa, kar pomeni, da mora obstajati pot za enosmerne signale od vsakega vozlišča do mase. Tako na primer `100k` upor `R1` med vozliščema `10` in `20` podamo kot

```
R1 (10 20) r=100k
```

Pogosto se pri opisih vezja pojavijo zelo dolge vrstice. Take vrstice lahko razbijemo na več vrstic s pomočjo znaka `+`.

```
To je zelo
+ dolga vrstica.
```

Na koncu vrstice, ki jo nadaljujemo z znakom +, je potrebno pustiti en presledek, ali pa postaviti presledek takoj za znak +. Tako bosta besedi `zelo` in `dolga` ločeni. Če tega ne storimo, dobimo sestavljeno besedo `zelodolga`.

V opisu vezja se lahko pojavljajo tudi komentarji. Komentarji obsegajo celotno vrstico, ki se mora začeti z znakom `*`.

Predpona	Faktor	Predpona	Faktor
`m`	10^{-3}	`T`	10^{12}
`u`	10^{-6}	`G`	10^{9}
`n`	10^{-9}	`Meg`	10^{6}
`p`	10^{-12}	`k`	10^{3}
`f`	10^{-15}		

Tabela 3.1: Predpone za enote v SPICE-u.

```
* To je komentar.
```

Števila v SPICE-u lahko podajamo v zapisu, ki se vsesplošno uporablja v računalništvu (zapisi `25000`, `25000.0`, `2.5e4`, `250000e-1` vsi predstavljajo vrednost `25000`). Poleg tega standardnega zapisa SPICE pozna tudi zapis s SI predponami (npr. `25k`). Seznam predpon je v tabeli 3.1.

Številu lahko sledi tudi enota (npr. `1kOhm`). Pisanje enot ni priporočljivo, saj lahko vodi v napake. Če na primer napišemo za kapacitivnost `0.1F`, si to SPICE razlaga kot `0.1f` oziroma 10^{-16}. SPICE ne loči med velikimi in malimi črkami. Tako `5M` ne pomeni $5 \cdot 10^{6}$, ampak $5 \cdot 10^{-3}$. Če potrebujemo vrednost $5 \cdot 10^{6}$, pišemo `5Meg`. Kjer to ni posebej omenjeno, se podajajo parametri v SI enotah (kg, m, s, A, V, Ω, S, F, H, ...).

Pogosto se zgodi (predvsem pri načrtovanju integriranih vezij), da želimo vezati več enakih elementov vzporedno. Vzemimo na primer tri enake MOS tranzistorje:

```
M10a (10 20 30 0) modn w=10u l=1u
M10b (10 20 30 0) modn w=10u l=1u
M10c (10 20 30 0) modn w=10u l=1u
```

Zgornji opis je potraten tako s stališča opisa vezja, kot tudi trajanja simulacije. Namesto njega lahko uporabimo sledeči zapis

```
M10 (10 20 30 0) modn w=10u l=1u m=3
```

Parameter *m* pove koliko enakih elementov bomo vezali vzporedno. Privzeta vrednost parametra je 1. V nadaljevanju njegovega pomena ne bomo več posebej razlagali.

3.2 Krmiljeni in neodvisni viri

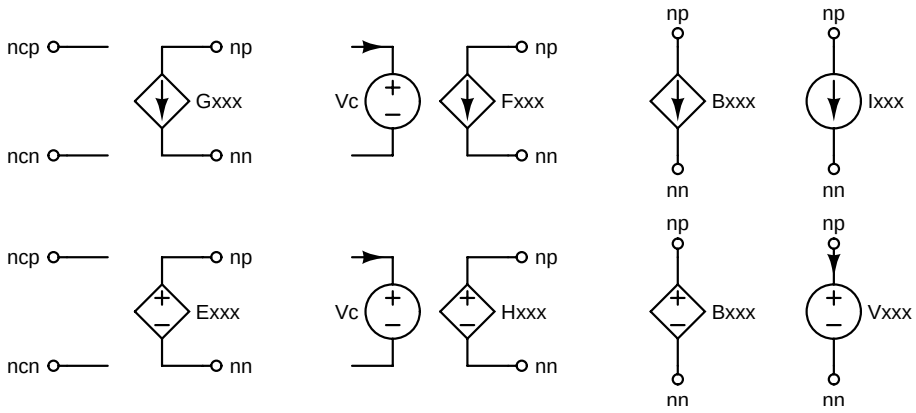
Slika 3.1 prikazuje simbole za tokovne in napetostne vire in pripadajoče oznake vozlišč.

Linearen napetostno krmiljen tokovni vir
`G`*xxx* `(`*np nn ncp ncn*`) gain=`*k*

Pri tem je vir priključen med sponki *np* in *nn*, krmili pa ga napetost med sponkama *ncp* in *ncn*. Faktor krmiljenja je podan z vrednostjo *k* z enoto A/V. Naj samo omenimo, da ime, ki ga predstavlja *xxx*, ni nujno dolgo tri črke, ampak je lahko poljubne dolžine.

Primer:

```
G1 (10 20 31 32) gain=1m
Gout (10 0 20 0) gain=10.0
```



Slika 3.1: Zgoraj: napetostno krmiljen, tokovno krmiljen, nelinearno krmiljen in neodvisen tokovni vir. Spodaj napetostno krmiljen, tokovno krmiljen, nelinearno krmiljen in neodvisen napetostni vir.

Linearen tokovno krmiljen tokovni vir
F*xxx* (*np nn Vc*) gain=*k*

Vir je priključen med sponki *np* in *nn*, krmili pa ga tok, ki teče v pozitivno sponko neodvisnega napetostnega vira *Vc*. Faktor krmiljenja je podan z vrednostjo *k* in nima enote.

Primer:

```
F1 (10 20 Vmeas) gain=1.5
Fd3 (10 0 Vamp) gain=2.0
```

Linearen napetostno krmiljen napetostni vir
E*xxx* (*np nn ncp ncn*) gain=*k*

Priključen je med sponki *np* in *nn*, krmili pa ga napetost med sponkama *ncp* in *ncn*. Faktor krmiljenja je podan z vrednostjo *k* in nima enote.

Primer:

```
E1 (10 20 92 93) gain=1.5
Ecmm (10 0 92 0) gain=2.0
```

Linearen tokovno krmiljen napetostni vir
H*xxx* (*np nn Vc*) gain=*k*

Vir je priključen med sponki *np* in *nn*, krmili pa ga tok, ki teče v pozitivno sponko neodvisnega napetostnega vira *Vc*. Faktor krmiljenja je podan z vrednostjo k z enoto V/A.

Primer:

```
H1 (10 20 92 93) gain=100
Hdif (10 0 92 0) gain=50
```

Nelinearno krmiljen tokovni/napetostni vir
B*xxx* (*np nn*) I=*izraz*
B*xxx* (*np nn*) V=*izraz*

V izrazu lahko nastopajo napetosti vozlišč (npr. `v(nn)`), napetosti med vozlišči (npr. `v(np,nn)`, kar je enako kot `v(np)-v(nn)`) in tokovi v pozitivne sponke neodvisnih napetostnih virov (npr. `i(Vc)`). Uporabljamo lahko aritmetične operatorje `+, -, *, /, ^` (seštevanje, odštevanje, množenje, deljenje, potenciranje), unarni operator negativnega predznaka (−) in funkcije `abs` (absolutna vrednost), `acos`, `asin`, `atan` (inverzni kosinus, sinus in tangens), `acosh`, `asinh`, `atanh` (inverzni hiperbolni kosinus, sinus in tangens), `cos`, `sin`, `tan` (kosinus, sinus in tangens), `cosh`, `sinh`, `tanh` (hiperbolni kosinus, sinus in tangens), `sqrt` (kvadratni koren), `exp` (eksponentna funkcija), `ln` (naravni logaritem), `log` (desetiški logaritem), `sgn` (predznak vrednosti; lahko je 0, 1 ali -1), `u` (enotina stopnica), `uramp` (enotina strmina; integral enotine stopnice). Uporabljamo lahko tudi števila s plavajočo vejico v vseh načinih zapisa, ki jih dovoljuje SPICE.

Primer:

```
Bvin (10 20) V=sqrt(v(2,3)+v(5,6)^3)+1e2
Binn (10 0) I=v(10)*i(vamp)+0.1m
```

Neodvisen napetostni vir
`V`*xxx* `(`*np nn*`)`
`+ dc=`*dc*
`+ acmag=`*acmag* `acphase=`*acphase*
`+` *transrc*`=(`*p1 p2 ...*`)`
`+ m=`*m*

Neodvisen tokovni vir
`I`*xxx* `(`*np nn*`)`
`+ dc=`*dc*
`+ acmag=`*acmag* `acphase=`*acphase*
`+` *transrc*`=(`*p1 p2 ...*`)`
`+ m=`*m*

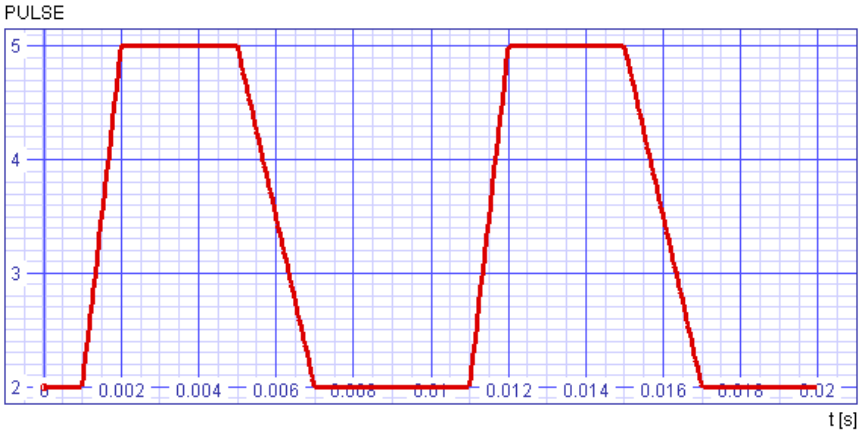
Pri tem *np* in *nn* podajata priključni vozlišči vira. Parameter *dc* določa enosmerno vrednost napetosti oziroma toka vira, katere privzeta vrednost je 0. Parametra *acmag* in *acphase* določata amplitudo in fazo (v stopinjah) kompleksorja $acmag \cdot \mathrm{e}^{\mathrm{i}\pi \cdot acphase/180}$, ki se uporablja v malosignalni (`ac`) in šumni (noise) analizi linearizirane vezja. Če opustimo parameter *acmag*, je amplituda kompleksorja enaka 0. Tudi parameter *acphase* lahko opustimo in s tem fazo nastavimo na 0°. Kot parameter *transrc* lahko navedemo eno od naslednjih vrednosti: `pulse`, `sin`, `exp`, `pwl` ali `sffm` in tako dobimo enega od petih vgrajenih časovnih potekov napetosti oziroma toka. Na primer sinusni vir z offsetom 1V, amplitudo 2V in frekvenco 1kHz podamo kot

```
V1 (10 20) sin=(1.0 2.0 1k)
```

Če ne podamo parametra *dc*, se pri enosmerni analizi delovne točke (`op` in `dc`) uporabi začetna vrednost (ob času 0) časovnega poteka, ki ga določa parameter *transrc*. V tem primeru dobimo pred simulacijo opozorilo. Če parameter *transrc* ni podan, ima vir konstantno vrednost v tranzientni analizi, ki jo določa parameter *dc*. V nasprotnem primeru se uporabi časovni potek določen s parametrom *transrc*. Parameter *m* pove koliko enakih virov želimo vezati vzporedno. Parameter nima kakega posebnega smisla za napetostne vire, je pa uporaben za tokovne vire.

V nadaljnjem bomo z Δ in t_{stop} označili začetni korak in končni čas tranzientne analize. Pri vseh petih časovnih potekih je obvezno podati le prvi dve vrednosti (*p1* in *p2*). Parameter *transrc* skupaj s koeficienti *p1*, *p2*, ... podaja enega od naslednjih petih časovnih potekov.

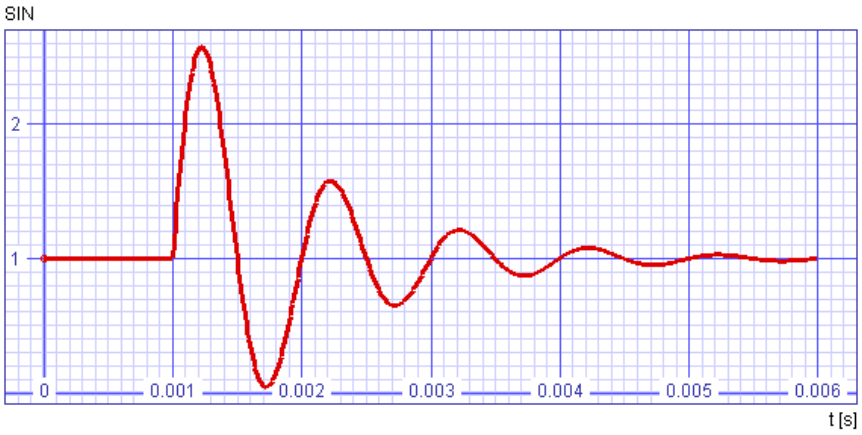
Pulzni potek (`pulse`)
`pulse=(`*v1 v2 td tr tf pw per ph*`)`



Slika 3.2: Pulzni vir, ki ga dobimo s parametrom `pulse=(2.0 5.0 1m 1m 2m 3m 10m)`.

Primer časovnega poeka za pulzni vir je na sliki 3.2. Parametra *v1* in *v2* določata začetno in končno vrednost stopnice, ki tvori prvo fronto verige enakih impulzov. Veriga se začne ob času *td*. Dvižni in upadni čas pulza sta podana s parametroma *tr* in *tf*. Širina pulza od trenutka, ko signal doseže vrednost *v2* do začetka updanja nazaj proti *v1* določa parameter *pw*. Parameter *per* določa periodo pulzov. Začetno fazo signala določimo s parametrom *ph*. Privzete vrednosti parametrov *td*, *tr*, *tf*, *pw*, *per* in *ph* so po vrsti 0, Δ, Δ, t_{stop}, t_{stop} in 0. Tako dobimo enkratno stopnico ob času nič z dvižnim časom Δ, če podamo le prva dva parametra (*v1* in *v2*). S tremi podanimi parametri se stopnica zamakne za čas *td*. Štirje podani parametri omogočajo, da tej stopnici nastavimo še dvižni čas. Šest parametrov določa enkraten pulz širine *pw*. Za verigo pulzov moramo podati sedem parametrov. Če podamo še neničelno začetno fazo *ph*, ima to za vrednosti po času *td* enak učinek, kot zakasnitev $-per \cdot ph/(2\pi)$. Vrednosti pred časom *td* so take, da ob času *td* ni nezveznosti.

Sinusni potek (`sin`)
`sin=(`*v0 va f td k ph*`)`

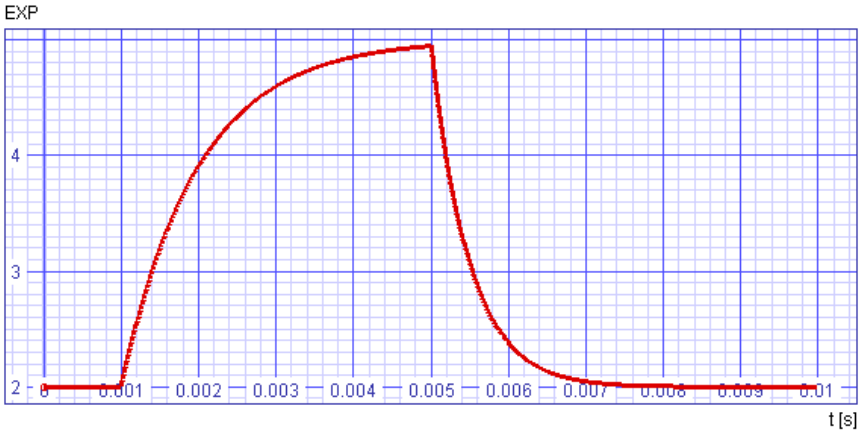


Slika 3.3: Sinusni vir, ki ga dobimo s parametrom `sin=(1.0 2.0 1k 1m 1e3)`.

Matematično lahko časovni potek tega vira zapišemo kot $v0 + va \cdot \sin(2\pi \cdot f \cdot (t - td) + ph)\mathrm{e}^{-\mathrm{k}(\mathrm{t}-\mathrm{td})}$ za čase večje od *td* in $v0 + va \cdot \sin(ph)$ za čase manjše od *td*. Parametra *v0* in *va* določata offset in amplitudo vira. Frekvenca vira je podana s parametrom *f*, zakasnitev začetka sinusnega nihanja in dušenja pa s parametrom *td*. Parameter *k* podaja obratno vrednost časovne konstante dušenja vira. Začetno fazo signala določimo s parametrom *ph*. Privzete vrednosti za *f*, *td*, *k* in *ph* so $1/\Delta$, 0, 0 in 0. Tako *v0* in *va* določata nedušeno sinusno nihanje s periodo, ki je enaka začetnemu koraku tranzientne analize. S tremi parametri lahko določimo tudi frevenco

nedušenega nihanja. Štirje parametri nam omogočajo, da podamo zakasnjen nedušen sinusni vir. Za dušen vir moramo podati pet parametrov. Če želimo vir, katerega amplituda eksponentno raste, podamo negativno vrednost za parameter *k*. Primer časovnega poeka za sinusni vir je na sliki 3.3.

Eksponentni potek (`exp`)
`exp=(`*v1 v2 td1 t1 td2 t2*`)`



Slika 3.4: Eksponentni vir, ki ga dobimo s parametrom `exp=(2.0 5.0 1m 1m 5m 0.5m)`.

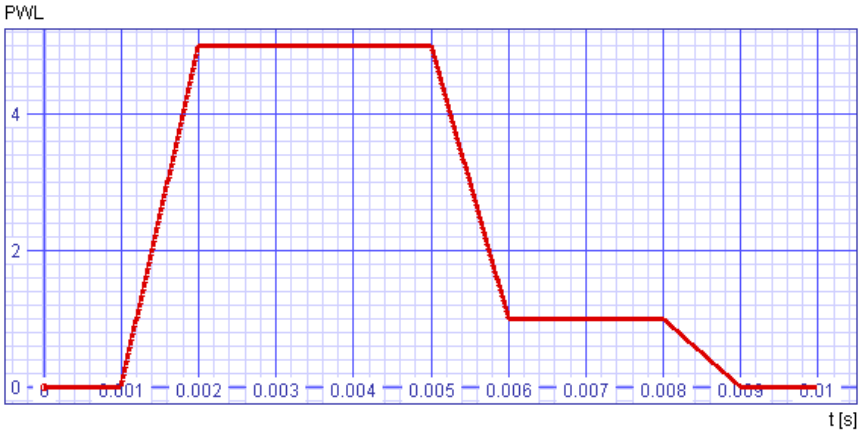
Časovni potek tega vira opišemo kot $v1+u(t-td1)(v2-v1)(1-\mathrm{e}^{(td1-t)/t1})+u(t-td2)(v1-v2)(1-\mathrm{e}^{(td2-t)/t2})$. Pri tem je $u(t)$ enotina stopnica. Parametra *v1* in *v2* določata prvi in drugi nivo vira. Dvig od prvega k drugemu nivoju se začne ob času *td1* in poteka s časovno konstanto *t1*. Upad nazaj na prvi nivo se začne ob času *td2* in poteka s časovno konstanto *t2*. Privzete vrednosti parametrov *td1*, *t1*, *td2* in *t2* so 0, Δ, $t_{d1}+\Delta$ in Δ. Tako s samo dvema parametroma dobimo konico, z vrhom pri $v1+(v2-v1)(1-\mathrm{e}^{-1})$. Vendar pa te vrednosti konica pogosto ne doseže, če ne postavimo strogih omejitev za zgornjo mejo časovnega koraka. Trije oziroma štirje parametri omogočajo, da to konico zakasnimo in spremenimo njeno strmino. Uporabno možnost dobimo šele, če podamo pet parametrov. Sedaj je začetni del konice popolnoma definiran, vrnitev na prvi nivo pa je še zmeraj zelo hitra (časovna konstanta Δ). Vseh šest parametrov nam omogoča, da določimo tudi časovno konstanto vračanja na prvi nivo. Primer časovnega poteka je na sliki 3.4.

Odsekoma linearen potek (`pwl`)
`pwl=(`*t1 v1 t2 v2 ...*`)`

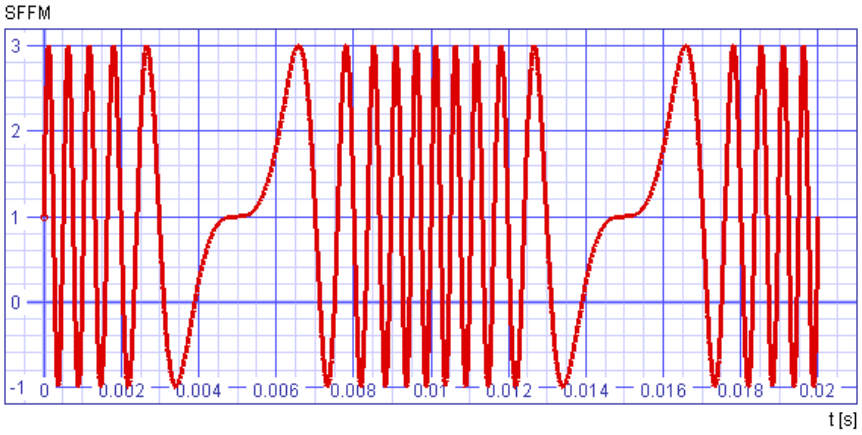
Pri odsekoma linearnem poteku (slika 3.5) podajamo časovni potek po točkah. Časovni potek tega vira je linearen med zaporednimi časi 0, *t1*, *t2*, ... Veljati mora $0 \leq t1 < t2 < \ldots$. Obvezno moramo podati vrednosti *t1* in *v1*. Ne glede na to, kaj izberemo za *t1*, je vrednost vira pri $t = 0$ enaka *v1*. Torej vir povezuje točke $(0, v1), (t1, v1), (t2, v2), \ldots$. Če je zadnji podan čas *tn*, je vrednost vira za čase $t > tn$ enaka *vn*.

Fazno moduliran potek (`sffm`)
`sffm=(`*v0 va f m fs phc phs*`)`

Časovni potek je določen z $v0+va\cdot\sin(2\pi\cdot f\cdot t+phc+m\cdot\sin(2\pi\cdot fs\cdot t+phs))$. Če podamo do tri parametre, se vir obnaša enako, kot sinusni vir s temi parametri. Privzeta vrednost modulacijskega indeksa *m* je torej 0. Privzeta vrednost frekvencem, s katero moduliramo nosilec `f`, pa je enaka $1/\Delta$. Smiselno je torej podati bodisi tri ali pa vsaj pet parametrov. Zadnja dva parametra (*phc* in *phs*) predstavljata začetno fazo nosilnega in modulacijskega signala. Njuna privzeta vrednost je 0. Primer časovnega poteka za fazno moduliran vir je na sliki 3.6.



Slika 3.5: Odsekoma linearen vir, ki ga dobimo s parametrom `pwl=(1m 0.0 2m 5.0 5m 5.0 6m 1.0 8m 1.0 9m 0.0)`.



Slika 3.6: Fazno moduliran vir, ki ga dobimo s parametrom `sffm=(1.0 2.0 1k 10 0.1k)`. Nosilec je torej 1kHz sinusni signal, modulacijski signal je 100Hz sinus.

3.3 Pasivni elementi

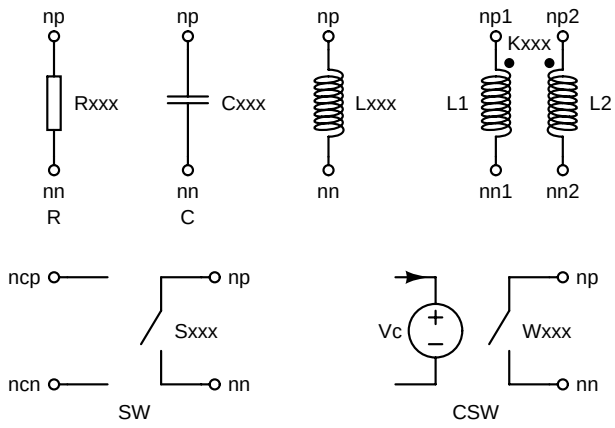
Simboli za osnovne pasivne elemente v SPICE-u so na sliki 3.7.

Upor

R*xxx* (*np nn*) `r=`*r* `tc1=`*tc1* `tc2=`*tc2* `m=`*m*

SPICE pozna tudi polprevodniški upor, ki ga podamo kot
R*xxx* (*np nn*) *ime_modela* `w=`*w* `l=`*l* `r=`*r* `tc1=`*tc1* `tc2=`*tc2* `m=`*m*
`.model` *ime_modela* `R (rsh=`*rsh* `tc1=`*tc1* `tc2=`*tc2* `...)`

w in *l* predstavljata širino in dolžino upora. Pri tem je *rsh* plastna upornost, *tc1* in *tc2* pa linearni in kvadratni temperaturni koeficient v Ω/K oziroma Ω/K^2. Njuna privzeta vrednost je 0. Če kateri od temperaturnih koeficientov ni podan z elementom, se uporabi vrednost iz modela. V stavku `.model` lahko podamo še veliko drugih parametrov, ki jih tukaj ne bomo omenjali. Več različnih uporov lahko uporablja en in isti model, tako da je v večini primerov dovolj že en sam stavek `.model`. Če uporu podamo konkretno upornost (parameter *r*), bo uporabljena podana vrednost upornosti namesto tiste, ki se izračuna iz plastne upornosti in dimenzij upora po formuli $rsh \cdot l/w$. V tem primeru tudi ni potrebno podajat širine in dolžine upora. Na ta način lahko dobimo temperaturno odvisne



Slika 3.7: Pasivni elementi: upor, kondenzator, tuljava, sklop med tuljavami, napetostno krmiljeno stikalo in tokovno krmiljeno stikalo. Elementi, za katere lahko podamo stavek `.model` imajo pod sabo naveden tip modela.

upore, ki so podani na klasičen način (z upornostjo).

Primer:

```
R1 (10 20) r=1k tc1=0.2m
R2 (50 51) rsemi w=1u l=10u
R3 (55 56) rsemi r=5k
.model rsemi r (rsh=1k tc1=0.1m tc2=5u)
```

Upornosti uporov `R1`, `R2` in `R3` so enake 1k, 10k in 5k. Pri tem izkazujejo vsi trije upori temperaturno odvisnost. Pri `R1` in `R3` jo določata vrednosti parametrov *tc1* in *tc2*, ki sta podani z elementom, za `R2` pa je temperaturna odvisnost določena z modelom `rsemi`.

Kondenzator

C*xxx* (*np nn*) c=*c* ic=*ic* m=*m*

SPICE pozna tudi polprevodniški kondenzator, ki je podan kot
C*xxx* (*np nn*) *ime_modela* w=*w* l=*l* c=*r* ic=*ic* m=*m*
.model *ime_modela* C (cj=*cj* cjsw=*cjsw* ...)

w in *l* predstavljata širino in dolžino kondenzatorja. Več polprevodniških kondenzatorjev lahko uporablja isti model. Če za polprevodniški kondenzator vrednost kapacitivnosti ni podana s parametrom *c*, se izračuna po formuli $cj \cdot w \cdot l + 2 \cdot cjsw \cdot (w + l)$. Obstajajo tudi drugi parametri modela polprevodniškega kondenzatorja, ki pa jih tu ne bomo omenjali. Parameter *ic* določa začetno vrednost napetosti na kondenzatorju, ki se uporabi, če pri tranzientni analizi podamo opcijo `uic`. Njegova privzeta vrednost je 0.

Primer:

```
C1 (10 20) c=1u ic=5
C2 (50 51) csemi w=1u l=10u m=2
C3 (55 56) csemi c=5p w=1u l=10u
.model csemi c (cj=1)
```

Kapacitivnosti kondenzatorjev `C1`, `C2` in `C3`, ki jih čuti vezje, so po vrsti 1u, 20p in 5p. Pri `C2` je kapacitivnost, ki jo čuti vezje podvojena (po formuli dobimo 10p) zaradi parametra `m`. Pri `C3` je uporabljena kapacitivnost, ki je podana s parametrom *c*, ne glede na to, da smo podali dimenzije kondenzatorja.

Tuljava
L*xxx* (*np nn*) l=*l* ic=*ic* m=*m*

Vrednost induktivnosti podamo s parametrom *l*. Parameter *ic* podaja začetno vrednost toka skozi induktivnost, ki se uporabi, če pri tranzientni analizi podamo opcijo uic. Njegova privzeta vrednost je 0. Vrednosti parametra m, ki so večje od 1, povzročijo sorazmerno zmanjšanje induktivnosti, ki jo čuti vezje.

Primer:

```
L1 (10 20) l=1m
```

Magnetni sklop
K*xxx* (*La Lb*) k=*k*

Sklop med induktivnostima *La* in *Lb* podamo s sklopnim faktorjem *k*. Pozitivne vrednosti sklopnega faktorja imajo za posledico, da sta piki, ki označujeta polariteto sklopa, pri obeh tuljavah na prvem vozlišču. Negativne vrednosti imajo za učinek, da je pika pri eni tuljavi na prvem, pri drugi pa na drugem vozlišču.

Primer (skoraj idealni transformator 1:1, primarno navitje je med vozliščema 1 in 2, sekundarno pa med 3 in 4):

```
L1 (1 2) l=1m
L2 (3 4) l=1m
K1 (L1 L2) k=0.999
```

Napetostno krmiljeno stikalo
S*xxx* (*np nn ncp ncn*) *ime_modela* m=*m*
.model *ime_modela* SW (vt=*vt* vh=*vh* ron=*ron* roff=*roff*)

Krmilna napetost je med vozliščema *ncp* in *ncn*. Razklenjeno stikalo ima upornost *roff*, sklenjeno pa *ron*. Stikalo preide iz razklenjenega stanja v sklenjeno stanje pri krmilni napetosti $vt + vh$. Prehod nazaj v razklenjeno stanje se zgodi pri $vt - vh$. Če je *vh* različen od 0, ima stikalo torej histerezo. Vrednosti *ron* in *roff* morata biti večji od 0, ker SPICE ne dovoljuje idealnih stikal. Stikala povzročajo konvergenčne probleme pri simulaciji, še posebej, če sta *ron* in *roff* zelo različna.

Primer:

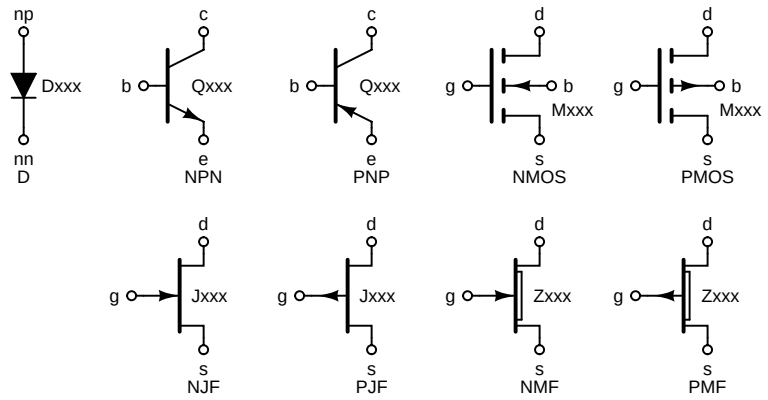
```
S1 (10 20 c1 c2) vcsw1
.model vcsw1 SW (vt=2.5 vh=1.0 ron=1m roff=1Meg)
```

Tokovno krmiljeno stikalo
W*xxx* (*np nn Vc*) *ime_modela* m=*m*
.model *ime_modela* CSW (it=*it* ih=*ih* ron=*ron* roff=*roff*)

Krmilni tok teče v pozitivno sponko neodvisnega napetostnega vira *Vc*. Pomen parametrov *it* in *ih* je enak, kot pomen parametrov *vt* in *vh* pri napetostno krmiljenem stikalu.

Primer:

```
W1 (10 20 Vin) ccsw1
.model ccsw1 CSW (it=1m ih=0.5m ron=1m roff=1Meg)
```



Slika 3.8: Polprevodniški elementi: dioda, bipolarni tranzistor, MOS tranzistor, spojni FET (JFET) in MESFET. MOS tranzistor z vgrajenim kanalom ni posebej omenjen, ker ga modeliramo z ustrezno prilagoditvijo pragovne napetosti.

3.4 Polprevodniški elementi

Pri modelih polprevodniških elementov (slika 3.8) bomo omenili samo tiste najpomembnejše. Ostali parametri za navadnega uporabnika, ki modele dobi od proizvajalca, niso zanimivi. Posebej velja omeniti parameter *level*, ki pove varianto modela. Različne variante modela imajo različne enačbe, ki opisujejo obnašanje elementa, in posledično tudi različna imena ostalih parametrov modela.

Dioda

`D`*xxx* `(`*np nn*`)` *ime_modela* `area=`*area* `m=`*m*
`.model` *ime_modela* `D (level=`*level* `is=`*is* `...)`

Parameter *is* predstavlja tok nasičenja diode. Več podrobnosti o modelu lahko zainteresirani bralec najde v [11]. Za skaliranje površine diode je na voljo parameter *area*, ki nima enote, njegova privzeta vrednost pa je 1. Za ostale vrednosti se parametri modela, kot je tok nasičenja, skalirajo z *area*. Dioda, katere parameter *area* je enak 2, ima površino dvakrat večjo od tiste, za katero je naveden tok nasičenja v modelu. Parameter *level* izbere nivo modela diode. Za diodo sta tako na voljo modela nivojev 1 in 3.

Primer:

```
D1 (10 20) dmod
.model dmod D (is=1e-12 bv=30)
```

Bipolarni tranzistor

`Q`*xxx* `(`*c b e*`)` *ime_modela* `area=`*area* `m=`*m*
`.model` *ime_modela tip* `(bf=`*bf* `...)`

Tip modela je lahko `NPN` ali `PNP`. Parameter *bf* predstavlja tokovno ojačenje. Ostale podrobnosti o modelu so na razpolago v [11]. Podobno, kot pri diodi, je tudi tukaj na voljo brezdimenzijski parameter *area* (privzeta vrednost je 1). S tem parametrom se skalirajo med drugim tudi tokovi nasičenja podani v modelu tranzistorja. V določenih primerih (pri simulaciji integriranih vezij) podajamo štiri priključna vozlišča (četrto vozlišče predstavlja substrat).
`Q`*xxx* `(`*c b e sub*`)` *ime_modela* `area=`*area* `m=`*m*

Primer:

```
Q1 (10 20 0) 2n2222
.model 2n2222 NPN (bf=150)
```

MOS tranzistor

`M`*xxx* `(`*d g s b*`)` *ime_modela* `w=`*w* `l=`*l* `m=`*m*
`+ ad=`*ad* `as=`*as* `pd=`*pd* `ps=`*ps* `nrd=`*nrs* `nrs=`*nrs*
`.model` *ime_modela tip* `(level=`*level* `...)`

Nivo	Model	Nivo	Model
1	Schichman-Hodges	47	BSIM3 v2
2	Grove-Frohman	53	BSIM3 v3
3	Empiričen model	55	BSIM3 SOI v1
4	BSIM1	56	BSIM3 SOI v2
5	BSIM2	57	US SOI3
6	MOS6	58	UFS
7	UF SOI	60	BSIM4
44	EKV		

Tabela 3.2: Nivoji MOS modela v SPICEu.

Tip modela je lahko `NMOS` ali `PMOS`. Parameter *level* določa nivo modela. Zaradi velike popularnosti CMOS tehnologije izdelave integriranih vezij se je tekom let nabralo veliko število modelov različnih nivojev, ki so našteti v tabeli 3.2. Več podrobnosti o enačbah modelov je na razpolago v [7, 11, 12, 13, 14].

Parametra *w* in *l* določata širino in dolžino kanala MOS tranzistorja, *ad* in *as* površini ponora in izvora, *pd* in *ps* obsega ponora in izvora ter *nrd* in *nrs* razmerji dolžine in širine (število kvadratkov) ponora in izvora. Zadnjih šest parametrov služi za določitev številnih parazitskih elementov MOS tranzistorja. Njihova privzeta vrednost je 0. Za osnovno simulacijo zadošča, če podamo parametra *w* in *l*. Omenimo še, da so MOSi simetrični elementi, kar pomeni, da lahko zamenjamo sponki izvora in ponora.

Primer (CMOS inverter):

```
M1 (out in vdd vdd) modp w=10u l=1u
M2 (out in vss vss) modn w=5u l=1u
.model modp PMOS (level=3 kp=0.001 vto=2.2)
.model modn NMOS (level=3 kp=0.002 vto=2.1)
```

Spojni FET (JFET)

`J`*xxx* `(`*d g s*`)` *ime_modela* `area=`*area* `m=`*m*
`.model` *ime_modela tip* `(level=`*level* `...)`

Tudi tukaj ima parameter *area* podoben pomen, kot pri diodi in bipolarnem tranzistorju. Tip modela je `NJF` ali `PJF`. Na voljo sta varianti modela 1 [11] in 2 [8].

Primer:

```
J1 (20 10 40) jfetn
.model jfetn NJF (level=1 ...)
```

MESFET

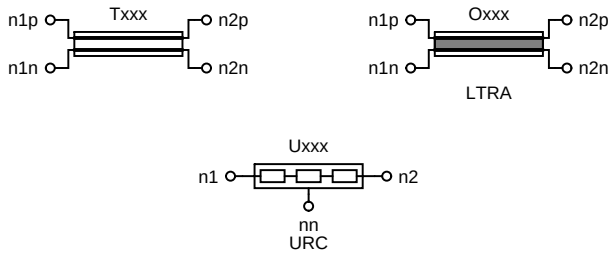
`Z`*xxx* `(`*d g s*`)` *ime_modela* `area=`*area* `m=`*m*
`.model` *ime_modela tip* `( ...)`

Enako kot JFET ima tudi MESFET parameter *area*. Tip modela je `NMF` ali `PMF`. Več o modelu lahko zainteresirani bralec najde v [11, 9].

Primer:

```
Z1 (20 10 40) mesp
.model mesp PMF ...
```

3.5 Prenosna linija



Slika 3.9: Linije. Zgoraj idealna in izgubna linija. Spodaj: Porazdeljena RC linija.

SPICE pozna tri vrste linij, katerih simboli so na sliki 3.9.

idealna linija
`T`*xxx* `(`*n1p n1n n2p n2n*`)` `zo=`*zo* `td=`*td* `f=`*f* `nl=`*nl* `m=`*m*

En vhod predstavljata sponki `n1p` in `n1n`, drugega pa sponki `n2p` in `n2n`. Idealna linija prenese signale, ki vstopajo na enem vhodu, nepopačene in zakasnjene na drugi vhod. Karakteristično impedanco linije podamo s parametrom *zo*. Dolžino linije lahko podamo bodisi s parametrom *td* kot zakasnitev v sekundah, ali pa s parametroma *f* in *nl* kot število valovnih dolžin (*nl*) valovanja s frekvenco *f*, ki potuje po liniji. Omenimo še, da kar se tiče simulacije, predstavlja idealna linija galvansko ločitev obeh vhodov. To pomeni, da moramo na obeh straneh linije definirati maso, če želimo, da bo simulacija sploh mogoča.

Primer:

```
T1 (5 0 8 0) zo=50 td=1u
```

Izgubna linija
`O`*xxx* `(`*n1p n1n n2p n2n*`)` *ime_modela* `m=`*m*
`.model` *ime_modela* `LTRA (r=`*r* `l=`*l* `c=`*c* `...)`

Pri modelu izgubne linije podajamo vzdolžno upornost (*r*), induktivnost (*l*) in kapacitivnost (*c*) na meter. Pri modelu lahko navedemo še celo vrsto parametrov, ki določajo obnašanje simulatorja. Teh parametrov tukaj ne navajamo. SPICE tovrstne linije simulira s pomočjo konvolucije med impulznim odzivom in vhodnim signalom. Tudi za izgubno linijo velja, da sta oba vhoda (kar se tiče simulacije) galvansko ločena, zato moramo na obeh straneh linije definirati maso. Več podrobnosti o modelu in simulaciji izgubnih linij je na razpolago v [10].

Primer:

```
O1 (5 1 8 2) lossy1
R1 (1 0) r=100Meg
```

```
R2 (2 0) r=100Meg
.model lossy1 LTRA (r=0.1 l=1u c=10n)
```

Enakomerno porazdeljena RC linija

U*xxx* (*n1 n2 nn*) *ime_modela* l=*l* n=*n* m=*m*
.model *ime_modela* URC (k=*k* fmax=*fmax* rperl=*rperl* cperl=*cperl*
+ isperl=*isperl* rsperl=*rsperl*)

Ta linija predstavlja poseben primer izgubne linije, ki nima induktivnosti in idealno izolacijo med žilama. Dolžino linije podaja parameter *l*. Pri simulaciji SPICE aproksimira tovrstno linijo z *n* RC segmenti. Če *n* ni podan, ga SPICE izračuna po formuli $\max(3, \log_k(2\pi \cdot fmax \cdot rperl \cdot cperl \cdot (k-1)^2/k^2))$. Konstanta širjenja je podana s parametrom *k*, upornost in kapacitivnost na meter pa s parametroma *rperl* in *cperl*. Če je podan parameter *isperl*, se kondenzatorji nadomestijo z diodami, katerih tok nasičenja in serijska upornost na meter sta podana s parametroma *isperl* in *rsperl*.

3.6 Avtomatska izbira modela (binning)

Pri CMOS tehnologijah se pogosto zgodi, da tudi kompleksen model, kot je BSIM3, ne zadostuje za dovolj natančen opis MOS tranzistorja za vse kombinacije širine in dolžine kanala. Zaradi tega razdelimo celotno območje parametrov *w* in *l* na podobmočja in podamo model tranzistorja kot zbirko parametrov za vsako od teh podobmočij. Območje parametrov *w* in *l* za katero je dan model veljaven, opišemo s štirimi parametri modela: (*wmin*, *wmax*, *lmin* in *lmax*). Tako bi na primer lahko za območje $w = 1\mu\mathrm{m}...100\mu\mathrm{m}$ in $l = 1\mu\mathrm{m}...10\mu\mathrm{m}$ podali štiri podobmočja:

```
.model modn_1 NMOS (level=53 wmin=1u  wmax=10u
+                           lmin=1u  lmax=2u ...)
.model modn_2 NMOS (level=53 wmin=1u  wmax=10u
+                           lmin=2u  lmax=10u ...)
.model modn_3 NMOS (level=53 wmin=10u wmax=100u
+                           lmin=1u  lmax=2u ...)
.model modn_4 NMOS (level=53 wmin=10u wmax=100u
+                           lmin=2u  lmax=10u ...)
```

Zaporedno številko podobmočja podamo za podčrtajem v imenu modela. Če sedaj dodelimo konkretnemu MOS tranzistorju enega od teh modelov, bo SPICE avtomatsko izbral pravega za dano širino in dolžino kanala.

```
M1 (1 2 5 5) modn_1 w=8u l=2.5u
```

Tranzistor M1 bo tako uporabljal model modn_2, čeprav smo ga uvrstili v model modn_1. Še več, ime modela tranzistorja lahko podamo celo brez podčrtaja in številke, pa bo SPICE še zmeraj sam poskrbel, da se bo uporabil primeren model.

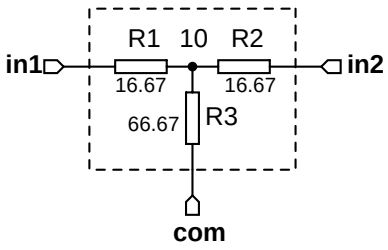
```
M1 (1 2 5 5) modn w=8u l=2.5u
```

Če bomo kdaj pozneje, ko bo vezje že naloženo, spremenili katero od dimenzij kanala, bo SPICE spet samodejno izbral ustrezen model.

3.7 Podvezja

Pogosto se zgodi, da se kak del vezja ponavlja. Spet drugič bi radi, da so elementi, ki tvorijo nek zaključen del vezja, združeni skupaj. V takih primerih uporabimo podvezja. Podvezje definiramo s sledečo sintakso.
`.subckt` *ime_modela_podvezja* `(`*n1 n2* `...)`
`...`
`.ends`

Vozlišča *n1*, *n2*, ... predstavljajo zunanje sponke podvezja. Med vrstici `.subckt` in `.ends` zložimo elemente, ki tvorijo podvezje. Ti elementi se povezujejo z zunanjimi sponkami podvezja preko vozlišč *n1*, *n2*, ... Uporabimo lahko tudi notranja vozlišča, katerih imena morajo biti drugačno od imen vozlišč zunanjih sponk. Izjema je vozlišče *0*, ki tudi znotraj definicije podvezja predstavlja maso.



Slika 3.10: Shema 6dB atenuatorja.

Tako bi na primer 6dB atenuator s karakteristično impedanco 50Ω (slika 3.10), katerega model bomo poimenovali `attn6`, definirali kot

```
.subckt attn6 (in1 in2 com)
R1 (in1 10) r=16.67
R2 (10 in2) r=16.67
R3 (10 com) r=66.67
.ends
```

Definicija podvezja še ne ustvari konkretnega podvezja, ki ga SPICE simulira. To naredim z naslednjo sintakso:
`X`*xxx* `(`*nc1 nc2* `...)` *ime_modela_podvezja*

S tem ustvarimo konkretno podvezje, ki vsebuje elemente naštete v definiciji podvezja, in ga priključimo med vozlišča *nc1*, *nc2*, ... Konkretno podvezje se tako imenuje `X`*xxx*, uporablja pa model *ime_modela_podvezja*. Naš atenuator lahko tako priklopimo v vezje z naslednjo vrstico

```
X1 (50 60 0) attn6
```

Tako bosta vhoda priključena na vozlišči `50` in `60`, skupna sponka atenuatorja pa na maso.

Masa, globalna vozlišča in modeli
Če bi znotraj definicije podvezja `attn6` uporabili vozlišče `50`, bi to ne imelo nič skupnega z vozliščem `50`, kamor priključimo naš atenuator `X1`. Pravimo, da so vozlišča znotraj definicije podvezja lokalna. Poznamo tudi globalna vozlišča. Takšno je na primer vozlišče `0`, ki predstavlja maso. Če bi definicijo atenuatorja napisali kot

```
.subckt attn6 (in1 in2)
R1 (in1 10) r=16.67
R2 (10 in2) r=16.67
R3 (10 0) r=66.67
.ends
```

bi bila skupna sponka atenuatorja zmeraj povezana na maso. Tak atenuator bi priključili v vezje z vrstico

```
X1 (50 60) attn6
```

Poleg vozlišča `0` lahko sami definiramo svoja globalna vozlišča. To naredimo z naslednjo sintakso
`.global` *n1 n2 ...*

Globalna vozlišča pridejo prav v digitalnih vezjih, kjer že vnaprej vemo, da bomo vsa vrata napajali preko vozlišč `vdd` in `vss`.

```
.global vdd vss

.subckt cmosinv (in out)
M1 (out in vdd vdd) modp w=10u l=1u
M2 (out in vss vss) modn w=5u  l=1u
.ends

X1 (src sink) cmosinv
V1 (vdd 0) dc=5
V2 (vss 0) dc=0
```

V zgornjem vezju je inverter `X1` priključen na napajanje, ki vozlišče `vdd` postavi na potencial 5V, vozlišče `vss` pa na 0V. MOS tranzistorja `M1` in `M2` potrebujeta modela `modn` in `modp`. Ta dva modela lahko navedemo znotraj definicije podvezja. V tem primeru sta modela lokalna in veljata samo znotraj podvezja. Lahko pa ju navedemo v glavnem vezju. Tedaj sta modela globalna in veljata za celotno vezje, razen znotraj definicij podvezij, ki vsebujejo lokalne modele z enakimi imeni.

Gnezdena podvezja, sploščeno vezje
Definicij podvezij ne moremo gnezditi (torej ne moremo ustvariti lokalne definicije podvezja, kakor to lahko naredimo z modeli elementov v SPICEu). Lahko pa znotraj podvezja uporabimo drugo podvezje. Za zgled bomo prikazali 6dB atenuator iz temperaturno odvisnih uporov.

```
.subckt attn6 (in1 in2 com)
R1 (in1 10) rmod r=16.67
R2 (10 in2) rmod r=16.67
R3 (10 com)   rmod r=66.67
.model rmod R (tc1=1e-4)
.ends
```

Tako bi lahko 12dB atenuator sestavili iz dveh verižno vezanih 6dB atenuatorjev.

```
.subckt attn12 (in1 in2 com)
X1 (in1 10 com) attn6
X2 (10 in2 com) attn6
.ends
```

12dB atenuator uporabimo v vezju.

```
X12 (50 60 0) attn12
```

Pravimo da smo podvezji `X1` in `X2` vgnezdili v podvezje `X12`. Kadar SPICE simulira, ne dela z gnezdenim vezjem, ampak pred simulacijo vezje splošči. Sploščeno vezje, ki nastane iz atenuatorja `X12`, je

```
R1:X1:X12 (50 10:X1:X12) rmod:X1:X12 r=16.67
R2:X1:X12 (10:X1:X12 10:X12) rmod:X1:X12 r=16.67
R3:X1:X12 (10:X1:X12 0) rmod:X1:X12 r=66.67
.model rmod:X1:X12 R (tc1=1e-4)

R1:X2:X12 (10:X12 10:X2:X12) rmod:X2:X12 r=16.67
R2:X2:X12 (10:X2:X12 60) rmod:X2:X12 r=16.67
R3:X2:X12 (10:X2:X12 0) rmod:X2:X12 r=66.67
.model rmod:X2:X12 R (tc1=1e-4)
```

Vidimo, da se imena lokalnih vozlišč, imena lokalnih modelov in imena elementov znotraj podvezij dopolnijo s potjo ali pa preimenujejo. Pot vsebuje zaporedje konkretnih podvezij, preko katerih pridemo do nekega lokalnega vozlišča, modela ali elementa. Tako se lokalno vozlišče `10` znotraj podvezja `X12` imenuje `10:X12`. Lokalno vozlišče `10` znotraj podvezja `X2`, ki je vgnezdeno v podvezju `X12`, se imenuje `10:X2:X12`. Podobno je tudi z modeli in elementi. Model `rmod` iz `X1`, vgnezdenem v `X12` se imenuje `rmod:X1:X12`. Tudi upor `R3` iz `X1`, vgnezdenem v `X12`, se pokorava enakim pravilom (v sploščenem vezju se imenuje `R3:X1:X12`). Do lokalnih vozlišč, modelov in elementov v podvezjih pridemo preko imen, ki jih imajo pripadajoči elementi v sploščenem vezju.

Parametrizirana podvezja

Pogosto si želimo, da bi lahko imeli eno definicijo podvezja za vsa podvezja z enako zgradbo in različnimi vrednostmi parametrov elementov. To dosežemo s parametriziranimi podvezji. Parametrizirano podvezje definiramo z naslednjo sintakso.

`.subckt` *ime_modela_podvezja* `(`*n1 n2* `...)`
`+ param:` *p1*`[=`*vd1*`]` *p2*`[=`*vd2*`]` `...` `...`
`.ends`

Tako definirano podvezje ima parametre *p1*, *p2*, ... Ti parametri imajo privzete vrednosti *vd1*, *vd2*, ... S parametri lahko računamo. Na voljo so vsi operatorji in funkcije, ki jih ponuja jezik NUTMEG[1] za gradnjo izrazov. Prav tako so na voljo vse konstante iz skupine vektorjev `const` (npr. `pi`, `e`, ...). Poglejmo si definicijo parametriziranega atenuatorja, ki mu lahko poljubno izberemo karakteristično impedanco in slabljenje.

```
.subckt attn (in1 in2 com) param: z0=50 k
R1 (in1 10) r={z0*(k-1)/(k+1)}
R2 (10 in2) r={z0*(k-1)/(k+1)}
R3 (10 com) r={z0*2*k/(k^2-1)}
.ends
```

Pri definiciji smo podali privzeto vrednost parametra `z0`, ki je enaka 50. Parameter `k` nima privzete vrednosti. Vidimo, da je treba izraze pisati v zavite oklepaje. Konkretno podvezje ustvarimo z naslednjo sintakso.

`X`*xxx* `(`*nc1 nc2* `...)` *ime_modela_podvezja*
`+ param:` *p1*`[=`*v1*`]` *p2*`[=`*v2*`]` `...`

S tem ustvarimo podvezje `X`*xxx* in za parametre *p1*, *p2*, ... uporabimo vrednosti *v1*, *v2*, ... Če vrednost parametra ni podana, se uporabi privzeta vrednost. Za parametre, ki nimajo privzete vrednosti, moramo vrednost podati najkasneje sedaj. Postavimo sedaj 6dB atenuator s karakteristično impedanco 50Ω med vozlišča `50`, `60` in `0`.

```
X1 (50 60 0) attn param: k=2
```

Naredimo še podobno za 12dB atenuator, ki pa naj ima tokrat karakteristično impedanco 75Ω.

[1] Jeziku NUTMEG je namenjeno celo poglavje v knjigi.

```
X2 (5 6 0) attn param: z0=75 k=4
```

Pogosto se dogaja, da se kak izraz večkrat pojavlja. V ta namen lahko ustvarimo nov lokalni parameter, ki ga nato uporabljamo v izrazih.
`.param` *ime=izraz*

Kot primer navedimo še enkrat definicijo parametriziranega atenuatorja, tokrat malo drugače.

```
.subckt attn (in1 in2 com) param: z0=50 k
.param z1=z0/(k+1)
R1 (in1 10) r={z1*(k-1)}
R2 (10 in2) r={z1*(k-1)}
R3 (10 com) r={z1*2*k/(k-1)}
.ends
```

`.param` vrstice lahko postavimo tudi izven definicije podvezja. V tem primeru so to globalni parametri, ki so dostopni v vseh definicijah podvezij.

3.8 Vključitev datotek v opis vezja

Pogosto so modeli elementov podani v ločenih datotekah. Takšno datoteko vključimo v vezje na sledeč način.
`.include` *ime_datoteke*

ime_datoteke lahko vsebuje pot do datoteke. Če je pot relativna, SPICE išče datoteko v poti relativni na mapo, kjer se nahaja datoteka s stavkom `.include`.

Včasih združimo več modelov (odsekov opisa vezja) v knjižnice. Zgradba datoteke s knjižnico je sledeča
`.lib` *ime_odseka_1*
vsebina_odseka_1
`.endl`

`.lib` *ime_odseka_2*
vsebina_odseka_2
`.endl`

`...`

Nek konkreten odsek iz take datoteke vključimo z vrstico
`.lib '`*ime_datoteke*`'` *ime_odseka*

3.9 Temperatura elementov in vezja

SPICE simulira celotno vezje pri temperaturi $27°$C. Če želimo vezje simulirati za katero drugo temperaturo, napišemo v opis vezja
`.options temp=`*vrednost*

Temperaturo podajamo v stopinjah Celzija.

Večina polprevodniških elementov in upor nam nudi parameter `temp` s katerim lahko nastavimo temperaturo za vsak konkreten element neodvisno od temperature vezja. Tako bi na primer temperaturo bipolarnega tranzistorja `Q1` lahko nastavili na $40°$C

```
Q1 (5 9 0) nbjt temp=40
```

Za elemente, katerih temperatura ni podana s parametrom `temp`, velja temperatura vezja.

3.10 Vključitev ukazov v opis vezja

V opis vezja lahko vključimo zaporedje ukazov, ki se izvršijo, kadar vezje naložimo v simulator. Te ukaze pišemo med vrstici `.control` in `.endc`. Ukazom, ki so med tema vrsticama, pravimo tudi ukazni blok. Uporabo ukaznega bloka si oglejmo kar na primeru.

```
Preprosto vezje

R1 (1 0) r=1k
V1 (1 0) dc=5

.control
op
print -i(v1)
.endc
.end
```

Zgornji opis vezja pozroči, da SPICE po nalaganju opisa izračuna delovno točko vezja, in izpiše tok, ki teče iz pozitivne sponke napetostnega vira `V1`.

3.11 Izbor delovne točke in začetnega stanja

Nekatera vezja imajo lahko več enosmernih delovnih točk. V katero od teh točk se postavi vezje je odvisno od začetnih pogojev. Analizi enosmerne delovne točke lahko pomagamo s pomočjo vrstice `.nodeset` s katero podamo začetni poskus za izračun enosmerne delovne točke.
`.nodeset v(`*ime1*`)=`*v1* `v(`*ime2*`)=`*v2* `...`

Če podamo začetni poskus dovolj blizu končne rešitve, potrebuje analiza delovne točke malo iteracij, da najde rešitev vezja.

Podoben stavek je na razpolago za določanje začetnega stanja vezja za analizo v časovnem prostoru.
`.ic v(`*ime1*`)=`*v1* `v(`*ime2*`)=`*v2* `...`

V vezju imamo lahko več `.nodeset` in `.ic` vrstic.

Primeri:

```
.nodeset v(1)=10 v(2)=5
.ic v(1)=0 v(2)=0.5
```

3.12 Parametri simulatorja

Simulator ponuja veliko število nastavitev ki do potankosti določajo njegovo delovanje. Te parametre nastavimo z vrstico
`.options` *ime1*`[=`*vrednost1*`]` *ime2*`[=`*vrednost2*`]` `...`

Več podrobnosti o parametrih simulatorja je na voljo v poglavju o analizi vezij.

Primeri:

```
* Relativna natancnost na 0.1%, tokovna natancnost na 1nA
* napetostna napancnost na 100uV
.options reltol=1e-3 abstol=1e-9 vntol=1e-4
* Izklopi avtomatsko izbiro konvergencnih pomagal
.options noautoconv
```

4

Analize

Ukaze, ki so opisani v tem poglavju (razen ukaza `options`), pišemo v ukazno okno oziroma med vrstici `.control` in `.endc` vhodne datoteke (ukazni blok). Vsaka analiza ustvari eno ali več skupin rezultatov. Vsaka skupina vsebuje enega ali več vektorjev. Vektorji predstavljajo izračunane veličine v vezju (kot so vozliščne napetosi in tokovi skozi napetostne vire). Vektor lahko vsebuje eno vrednost (npr. kadar je skupina rezultatov ustvarjena z analizo delovne točke) ali pa več vrednosti (npr. pri tranzientni analizi, kjer vsaka vrednost predstavlja izračunano veličino v vezju ob nekem času). Pri določenih analizah ima poseben pomen privzeti vektor v skupini rezultatov.

Tako na primer pri tranzientni analizi privzeti vektor (z imenom `time`) vsebuje vrednosti časa ob katerih je bilo vezje simulirano. Če zapišemo v en stolpec tabele vrednosti iz privzetega vektorja `time`, v drug stolpec pa vrednosti iz vektorja, ki opisuje napetost vozlišča `10`, dobimo tabeliran časovni potek napetosti v vozlišču `10`. Če na x-os nanašamo vrednosti iz vektorja `time`, na y-os pa vrednosti iz vektorja vozliščne napetosti, dobimo graf časovnega poteka napetosti v vozlišču.

4.1 Analiza delovne točke (`op`)

Pri tej analizi so vse kapacitivnosti odstranjene iz vezja in vse induktivnosti kratko sklenjene. Izvršimo jo z ukazom `op`

Primer	Pomen	NUTMEG ime
`V(10)`	Napetost vozlišča `10`	`v(10)`
`abc`	Napetost vozlišča `abc`	`v(abc)` ali `abc`
`v1#branch`	Tok skozi vir `V1`	`v1#branch` ali `i(v1)`
`e1#branch`	Tok skozi krmiljen vir `E1`	`e1#branch` ali `i(e1)`
`h1#branch`	Tok skozi krmiljen vir `H1`	`h1#branch` ali `i(h1)`
`b1#branch`	Tok skozi krmiljen vir `B1`	`b1#branch` ali `i(b1)`
`l1#branch`	Tok skozi induktivnost `L1`	`l1#branch` ali `i(l1)`

Tabela 4.1: Imena vektorjev, ki so rezultat `op` analize v SPICEu.

Pri analizi delovne točke se izračunajo vozliščne napetosti in tokovi skozi napetostne vire in induktivnosti. Imena vektorjev, ki so rezultat analize, so našteta v tabeli 4.1.

Analiza delovne točke je kljub svoji enostavnosti pogosto največji problem za konvergenco. Osnovni pogoj za konvergenco je, da za vsako vozlišče obstaja pot za enosmeren signal, ki vodi do mase. Brez tega dobimo v opisu

vezja (matriki) vozlišča, katerih potencial je nedoločen (plavajoč). Zapomniti si velja, da idealna in izgubna linija (kar se tiče simulacije) ne predstavljata poti za enosmerne signale med obema vhodoma. Tako je treba na obeh straneh linije definirati maso.

Pri analizi delovne točke imajo neodvisni viri vrednosti, ki jih podajajo parametri *dc*. Če za neodvisen vir parameter *dc* ni podan, parameter *transrc* pa je, se uporabi vrednost ustreznega časovnega poteka ob času 0.

4.2 Analiza enosmerne karakteristike (`dc`)

Pri tej analizi spreminjamo nek parameter vezja in opazujemo kako se spreminja delovna točka vezja. Za vsako vrednost parametra dobimo eno delovno točko vezja. Vektorji, ki so rezultat te analize, vsebujejo v splošnem več točk. Privzeti vektor se imenuje `sweep` in vsebuje vrednosti parametra, ki jim pripadajo izračunane delovne točke. `dc` analiza, ki izračuna 10 delovnih točk je hitrejša, kot 10 zaporednih `op` analiz, ker za vsako naslednjo delovno točko uporabi rešitev prejšnje kot začetni poskus. Na razpolago sta dve obliki analize.
`dc` *ime_parametra start stop step*
`dc` *ime_parametra start stop tip nstep*

Prva oblika preleti vse vrednosti parametra od *start* do *stop* v korakih *step*. Druga oblika ima dodaten paramerer *tip*, ki pove način preleta vrednosti med *start* in *stop*. Na razpolago so trije načini: `dec`, `oct` in `lin`. V načinu `dec` se vrednosti preletijo v logaritemski skali. Parameter *nstep* pri tem pove koliko točk bo preiskanih na dekado (razpon od 1 do 10). Če je $start = 10$ $stop = 1000.1$ in $nstep = 10$, vsak vektor vsebuje 21 točk. Način `oct` je podoben, le da tokrat parameter *nstep* pove število točk na oktavo (razpon od 1 do 2). Če je $start = 5$ $stop = 10.1$ in $nstep = 8$, so vektorji dolgi 9 točk. Način `lin` pomeni linearen prelet od *start* do *stop*. Za $nstep = 10$ se izračuna 11 točk med *start* in *stop*.

Ime parametra	Pomen
v*xxx*	Parameter `dc` vira `V`*xxx*
i*xxx*	Parameter `dc` vira `I`*xxx*
`@`*element*`[`*par*`]`	Parameter *par* elementa *element*
`@`*element*`[`*par*`][`*n*`]`	*n*-ta komponenta vektorskega parametra *par* elementa *element*
`@@`*model*`[`*par*`]`	Parameter *par* modela *model*
`@@`*model*`[`*par*`][`*n*`]`	*n*-ta komponenta vektorskega parametra *par* modela *model*
`@@@temp`	Temperatura vezja v stopinjah Celzija
`@@@tempk`	Temperatura vezja v stopinjah Kelvina

Tabela 4.2: Imena parametrov za `dc` analizo v SPICEu.

Možnosti, ki jih imamo pri navajanju imena parametra so naštete v tabeli 4.2.

Primeri:

```
* Parameter dc vira v1, korak 0.001 linearno
dc v1 -2.0 2.0 0.001
* Parameter r upora r1, 10 tock/dekado
dc @r1[r] 100 100k dec 10
* Parameter tc modela rmod, 200 tock linearno
dc @@rmod[tc1] -1m 1m lin 200
* Temperatura vezja, korak 10.0 linearno
dc @@@temp -50.0 150.0 10.0
```

4.3 Malosignalna enosmerna analiza (tf)

Analiza nam daje vpogled v vhodno in izhodno upornost (oziroma prevodnost) in prenosno funkcijo dvovhodnega vezja, ki je v delovni točki linearizirano. Tudi pri tej analizi so kapacitivnosti odstranjene iz vezja in induktivnosti kratko sklenjene. Najprej se izračuna delovna točka in vezje linearizira. Vsi nadaljnji izračuni se naredijo na lineariziranem vezju. Sintaksa analize je
`tf` *izhodna_velicina vhodni_vir*

Izhodna veličina je lahko bodisi napetost vozlišča (npr. `v(10)`), napetost med vozliščema (npr. `v(10,20)`) ali tok neodvisnega napetostnega vira (npr. `i(vamp)`). Vhodni vir je ime neodvisnega napetostnega oziroma tokovnega vira (`V`*xxx* oziroma `I`*xxx*), ki je priključen na vhod vezja. Možne so torej štiri kombinacije toka in napetosti na vhodu oziroma izhodu. V vseh štirih primerih dobimo kot rezultat vhodno in izhodno upornost v vektorjih `input_impedance` in `output_impedance`. Izhodna upornost je izračunana pravilno le, če je izhodna veličina napetost. Poleg upornosti dobimo še prevajalno funkcijo (vektor `transfer_function`), katere enota pa je odvisna od kombinacije vhodne in izhodne veličine:

- napetostni vir na vhodu in napetost kot izhodna veličina nam data napetostno ojačenje,
- tokovni vir na vhodu in napetost kot izhodna veličina nam data transimpedanco,
- napetostni vir na vhodu in tok kot izhodna veličina nam data transkonduktanco in
- tokovni vir na vhodu in tok kot izhodna veličina nam data tokovno ojačenje.

Vhodna upornost se izračuna kot obratna vrednost odvoda toka po enosmerni napetosti vhodnega napetostnega vira, oziroma kot odvod napetosti na vhodnem tokovnem viru po njegovem enosmernem toku. Izhodna upornost se računa tako, da se na izhod priklopi neodvisen tokovni vir in nato izračuna odvod napetosti na viru po toku vira. Prenosna funckija se določi kot odvod izhodne veličine po enosmerni vrednosti vhodnega vira.

Tok neodvisnega napetostnega vira je pozitiven, če teče v plus sponko vira (prvo vozlišče vira). Napetost na tokovnem viru je pozitivna, če je sponka, ki vleče tok v vir (prvo vozlišče vira), na višjem potencialu, kot sponka, ki daje tok v vezje.

Primeri:

```
* Napetostno ojacenje v delovni tocki
tf v(20,10) vinput
* Transkonduktanca
tf i(voutamp) vinput
* Transimpedanca
tf v(out) iinput
```

4.4 Malosignalna kompleksna analiza (ac)

Pri `ac` analizi se najprej naredi analiza delovne točke (enako, kot pri `tf` analizi). Nato se vezje v tej delovni točki linearizira. Tokrat se upoštevajo tudi kapacitivnoti in induktivnosti. `ac` analiza poteka v frekvenčnem prostoru, zato namesto prevodnosti povsod tekom analize nastopajo admitance. Te so kompleksne in frekvenčno odvisne zaradi kapacitivnosti in induktivnosti v vezju. Neodvisni viri predstavljajo vzbujanja, katerih kompleksorji so določeni s parametroma *acmag* in *acphase* kot $acmag \cdot \mathrm{e}^{\mathrm{i}\pi \cdot acphase/180}$. Neodvisni viri, ki nimajo podanega parametra *acmag*, predstavljajo bodisi odprte sponke (tokovni viri) oziroma kratek stik (napetostni viri). Analiza izračuna vse veličine, ki jih izračuna tudi `op` analiza, le da so te sedaj kompleksne in predstavljajo odziv vezja

na kompleksno vzbujanje. Običajno imamo v vezju samo en neodvisen vir s podanim parametrom *acmag*, ki je nastavljen na 1. Ta vir je priključen na vhod vezja. V tem primeru so odzivi vezja (vozliščne napetosti in tokovi) kar kompleksne vrednosti prevajalne funkcije iz vhoda vezja na posamezne vozliščne napetosti oziroma tokove neodvisnih napetostnih virov.

`ac` analiza preleti območje frekvenc in za vsako frekvenco izračuna kompleksen odziv vezja na malosignalno vzbujanje. Dobljeni vektorji vsebujejo toliko točk, kot je frekvenc. Vsaka točka prestavlja odziv vezja pri eni frekvenci. Privzeti vektor se imenuje `frequency`, je kompleksen in vsebuje frekvence, pri katerih je bilo vezje analizirano. Imaginarni del vsake komponente tega vektorja je 0. `ac` analizo poženemo z ukazom
`ac` *tip nstep fstart fstop*

Vrednost parametra *tip* je lahko `dec`, `oct` ali `lin` in ima enak pomen, kot pri `dc` analizi. Parameter *nstep* določa število točk na dekado, oktavo oziroma linearno območje med *fstart* in *fstop*. Parametra *fstart* in *fstop* določata območje frekvenc, za katere se bo vezje analiziralo.

Primeri:

```
* 10 tock na dekado med 10Hz in 100MHz
ac dec 10 10.0 100Meg
* 50 tock med 100Hz in 110Hz
ac lin 50 100.0 110.0
```

4.5 Iskanje polov in ničel (`pz`)

Prevajalno funkcijo vezja lahko podamo v zapisu s poli in ničlami

$$H(2\pi\mathrm{i}\cdot f) = A\frac{(2\pi\mathrm{i}\cdot f - z_1)\cdot(2\pi\mathrm{i}\cdot f - z_2)\cdot ...}{(2\pi\mathrm{i}\cdot f - p_1)\cdot(2\pi\mathrm{i}\cdot f - p_2)\cdot ...}$$

Števila $z_1, z_2, ...$ predstavljajo ničle, števila $p_1, p_2, ...$ pa pole prevajalne funkcije. Velja, da je sistem stabilen, če ležijo poli levo od imaginarne osi. SPICE zna poiskati pole in ničle vezja. V ta namen ima vgrajeno `pz` analizo, katere sintaksa je
`pz` *in1 in2 out1 out2 tip_funkcije vrsta_analize*

Tudi ta analiza najprej izračuna enosmerno delovno točko vezja (enako, kot pri `op` analizi), kjer ga nato linearizira. Če je parameter *tip_funkcije* enak `vol`, se analiza ukvarja s prevajalno funkcijo, ki jo dobimo, če vezje vzbujamo z neodvisnim napetostnim virom priklopljenim med sponki *in1* in *in2*, katerega parameter *acmag* je enak 1, in opazujemo napetost med sponkama *out1* in *out2*. Če pa je parameter *tip_funkcije* nastavljen na `cur`, se analiza ukvarja s prevajalno funkcijo, ki jo dobimo, če vezje vzbujamo s tokovnim virom, katerega parameter *acmag* je enak 1, med sponkama *in1* in *in2* in spet opazujemo napetost med sponkama *out1* in *out2*. V obeh primerih je prevajalna funkcija številčno enaka $v(out1) - v(out2)$.

Parameter *vrsta_analize* ima lahko vrednosti `pol`, `zer` ali `pz`, odvisno od tega ali želimo samo pole, samo ničle ali pa kar oboje. Vektor `pole` vsebuje kompleksne pole prevajalne funkcije. Pri tem vektor `pole_valid` pove, kateri od teh polov so veljavni (enica pomeni, da je pol veljaven). Neveljavni poli iz vektorja `pole` ne predstavljajo polov vezja. Vektorja `zero` in `zero_valid` podajata ničle vezja. `pz` analiza ne zna poiskati polov in ničel vezja, ki vsebuje idealne ali izgubne linije.

Primeri:

```
pz 1 2 10 21 vol pz
pz 1 2 10 0  cur zer
pz 1 0 10 0  cur pol
```

4.6 Šumna analiza (noise)

Tudi šumna analiza računa z malosignalnim lineariziranim modelom vezja v delovni točki (kot tf, ac in pz analize). Predpostavljamo, da je amplituda šuma dovolj majhna, da lahko vezje obravnavamo kot linearno. Pri šumni analizi SPICE najprej izračuna delovno točko vezja, nato pa vezje linearizira v delovni točki. Za vsako frekvenco, za katero nas zanima gostota šumnega spektra, SPICE izračuna doprinos vseh šumnih virov k izhodnemu šumu.

Vsak šumni vir v vezju je predstavljen z enim neodvisnim tokovnim virom priključenim na sponke elementa, ki je vir šuma. Za vsak tak šumni vir SPICE izračuna njegov prispevek na izhodu vezja. Pri tem so vsi neodvisni viri, ki jih navede uporabnik v opisu vezja, izklopljeni ($acmag = 0$). Kvadrat prispevka se za vsak šumni vir hrani v svojem vektorju. Ker so ti prispevki nekorelirani, nam da vsota kvadratov prispevkov posameznih šumnih virov gostoto izhodnega šumnega spektra. Ker je izhod kar napetost med vozliščema, je enota za gostoto izhodnega šumnega spektra V^2/Hz.

Šumno analizo zaženemo z ukazom
noise *izhod vhodni_vir tip npts fstart fstop* [*ptssum*]

Izhod podamo kot napetost vozlišča (v(n1)), ali pa kot napetost med vozliščema (npr. v(n1,n2)). Vhodni vir je lahko katerikoli neodvisen napetostni ali tokovni vir. Parametri *tip*, *npts*, *fstart* in *fstop* podajajo frekvenčno območje v katerem se bo opravila šumna analiza. Zadnji parameter (*ptssum*) ima privzeto vrednost 1 in pove na vsake koliko izračunanih frekvenčnih točk med *fstart* in *fstop* se bo zapisala ena točka v vektorje šumnega spektra.

Analiza ustvari dve skupini rezultatov. Prva skupina vsebuje gostote šumnega spektra v odvisnosti od frekvence. Drugo skupino tvorijo vektorji katerih komponente predstavljajo integrale gostote šumnega spektra na območju od *fstart* do *fstop*.

Privzeti vektor prve skupine se imenuje frequency in ima enak pomen, kot v AC analizi. Prispevki posameznih šumnih virov k izhodnemu šumnemu spektru se hranijo v vektorjih oblike onoise_*element_vir*. Te prispevke dobimo samo, če podamo parameter *ptssum*. Gostota izhodnega šumnega spektra se hrani v vektorju onoise_spectrum. Poleg izračuna prispevkov šumnih virov SPICE izračuna tudi prevajalno funkcijo od vhodnega vira na izhod, ki je številčno enaka kar napetosti med izhodnima sponkama.

Gostota izhodnega šumnega spektra deljena s kvadratom absolutne vrednosti te prevajalne funkcije se shrani v vektor inoise_spectrum. Če smo parameter *acmag* vhodnega vira nastavili na 1, nam ta vektor predstavlja izhodni šumni spekter preračunan na vhodni vir. Če bi v brezšumnem vezju imel vhodni vir tak šumni spekter, bi bil izhodni šumni spekter enak tistemu, ki ga na izhodu daje šumno vezje z bezšumnim vhodnim virom. Enote vektorja *inoise_spectrum* so V^2/Hz, če je vhodni vir neodvisen napetostni vir, in A^2/Hz, če je vhodni vir neodvisen tokovni vir.

Druga skupina rezultatv vsebuje integrale šumnega spektra in prispevkov šumnih virov. Vsi vektorji v tej skupini rezultatov imajo samo eno komponento. Tudi tukaj velja, da dobimo prispevke posameznih šumnih virov samo, če podamo parameter *ptssum*. Integral vhodnega in izhodnega šumnega spektra se hrani v vektorjih inoise_total in onoise_total. Prispevki posameznih šumnih virov k integralu izhodnega šuma so v vektorjih onoise_total_*element_vir*. Poleg prispevkov posameznih šumnih virov k izhodnemu šumu hrani ta skupina rezultatov tudi njihove prispevke k vhodnemu šumu v vektorjih inoise_total_*element_vir*. Enote vektorjev inoise_total in inoise_total_*element_vir* so V^2, če je vhodni vir neodvisen napetostni vir, oziroma A^2, če je vhodni vir neodvisen tokovni vir.

Primeri:

```
* Shrani prispevke posameznih sumnih virov
noise v(10) vin dec 10 10.0 100Meg 1
* Izracunaj samo izhodni in ekvivalenten vhodni sum
noise v(10,9) i1 dec 10 10.0 100Meg
```

4.7 Tranzientna analiza (`tran`)

Ta analiza poteka v časovnem prostoru. Tudi tukaj se upoštevajo induktivnosti in kapacitivnosti v vezju. Sintaksa ukaza, ki izvrši tranzientno analizo je
`tran` *tstep tstop tstart tsm* `[uic]`

Obvezno je treba podati parametra *tstep* in *tstop*, ki podajata izpisni korak in končni čas analize. Privzeta vrednost za parameter *tstart* je 0. Kadar je podan parameter *tstart*, se vezje še zmeraj analizira od časa 0 do *tstop*, izračunane vrednosti pa se shranjujejo od časa *tstart* naprej. Če parametra *tsm* ne podamo, ga določi SPICE po formuli $tsm = \min(rmax \cdot tstep, tstop)$. Če je v vezju še kak neodvisen vir s časovnim potekom `sin` ali `sffm`, se postavi še dodatna zgornja meja za časovni korak, ki je enaka $fs/(2 \cdot fmax)$, kjer je *fmax* najvišja frekvenca neodvisnih virov. Privzeta vrednost parametra *rmax* simulatorja je 5. Pri tranzientni analizi časovni korak ni konstanten, ampak se prilagaja vezju. Kadar se vezje odziva manj živahno, se korak poveča. Ob hitrih spremembah se korak samodejno zmanjša.

Analiza v časovnem prostoru ustvari skupino rezultatov, ki vsebuje enak nabor vektorjev, kot pri `op` analizi. Privzeti vektor skupine je `time` in vsebuje časovne točke, ob katerih so izračunane veličine v vezju.

Začetno stanje vezja se določi z analizo delovne točke. Ta se razlikuje od navadne `op` analize po tem, da se za vrednosti neodvisnih virov uporabijo začetne vrednosti (ob času 0) časovnih potekov (*pulse*, *sin*, *exp*, *pwm* ali *sffm*), če so ti podani. Če kak časovni potek ni podan, se uporabi vrednost parametra *dc*. Iz rezultatov analize delovne točke se določijo začetna stanja kondenzatorjev in tuljav v vezju.

Če želimo uporabiti drugo začetno točko, podamo parameter `uic`. Začetna stanja reaktanc se tedaj določijo na osnovi vrednosti *ic* parametrov elementov. Če *ic* parameter ni podan za kak element, se začetno stanje določi na osnovi vrednosti podanih v `.ic` vrsticah. Za vozlišča, za katera ni podanih vrednosti v `.ic` vrsticah, se privzame vrednost 0. Izračunane vrednosti za prvo točka take tranzientne analize (ob času 0) so napačne. Da bi bile vrednosti pravilne tudi za prvo točko, moramo podati parameter *icstep* simulatorja. Tako se najprej izračuna odziv vezja za čas $tstep/(rmax \cdot icstep)$. Rešitev se nato proglasi za rešitev ob času 0. Ta pristop daje pravilne rezultate za velike vrednosti parametra *icstep*. Tipično nastavimo $icstep = 10^9$. Prevelike vrednosti parametra *icstep* lahko povzročijo težave s konvergenco.

Primeri:

```
* Od 0ms do 10ms
tran 0.01m 10m
* Od 5ms do 10ms
tran 0.01m 10m 5m
* Od 0ms do 10ms, uporabi informacijo o zacetnem stanju
tran 0.01m 10m 0m 0.01m uic
```

4.8 Algoritmi za izboljšanje konvergence simulatorja

Predvsem analiza enosmerne delovne točke zna biti problematična za simulator. Za primer, kadar simulator ne uspe najti rešitve nelinearnega sistema enačb, oziroma kadar časovni korak tranzientne analize postane pretirano majhen, je na razpolago cela vrsta konvergenčnih pomagal. Vsa konvergenčna pomagala so namenjena analizi delovne točke, razen modifikacije vezja s kapacitivnostmi, ki je namenjena tranzientni analizi.

Modifikacija vezja s prevodnostmi in kapacitivnostmi
Simulator nudi možnost priklopa uporov (v primeru težav pri analizi delovne točke) in kapacitivnosti (pri težavah z analizo v časovnem prostoru) med vsa vozlišča in maso. Velikost uporov nastavimo z opcijo simulatorja *rshunt*,

velikost kapacitivnosti pa z opcijo *cshunt*. Tako modificirano vezje ima manj težav s konvergenco, vendar pa se rezultat lahko precej razlikuje od pravilnega če je vrednost *cshunt* prevelika oziroma vrednost *rshunt* premajhna.

Prevodnostno korakanje
Pri prevodnostnem (GMIN) korakanju se med vsa vozlišča in maso najprej priklopijo upori, katerih prevodnost je velika v primerjavi z prevodnostmi vezju. Nato simulator naredi analizo delovne točke. Sledi povečanje upornosti uporov in ponovna analiza delovne točke, tokrat z začetnim poskusom, ki je enak rezultatu prejšnje analize delovne točke. S postopnim povečevanjem uporov postaja vezje čedalje bolj podobno prvotnemu vezju. Ko prevodnost pade pod vrednost parametra *gmindc* se algoritem konča. Zgornjo mejo za število korakov določa parameter *gminsteps*. Vrednost 0 izklopi prevodnostno korakanje. Dosežena rešitev se uporabi kot začetni poskus za navadno analizo delovne točke.

Korakanje enosmerne vrednosti neodvisnih virov
Pri tej vrsti korakanja se vrednosti neodvisnih virov postopoma spreminjajo od 0 do vrednosti, ki se uporablja v analizi delovne točke. Zgornjo mejo za število korakov določa parameter *srcsteps*. Vrednost 0 izklopi korakanje enosmerne vrednosti neodvisnih virov. Dosežena rešitev se uporabi kot začetni poskus za navadno analizo delovne točke.

Dušena Newton-Raphsonova metoda
Označimo z x_i približek rešitve enačbe $g(x) = 0$, z $G(x) = dg(x)/dx$ pa matriko prevodnosti. Pri Newton-Raphsonovi iteracijski metodi se približek rešitve izračuna po formuli $x_{i+1} = x_i - G^{-1}(x_i)g(x_i)$. Pri dušeni Newton-Raphsonovi metodi je iteracijska formula spremenjena v $x_{i+1} = x_i - \beta G^{-1}(x_i)g(x_i)$, kjer je $0 < \beta \leq 1$. β predstavlja faktor dušenja, ki za $\beta = 1$ da osnovno Newton-Raphsonovo iteracijsko formulo. Opcija *sollimiter* določa zgornjo mejo za število dušenih iteracij, ki je enaka $sollimiter \cdot itl1$. Pri tem parameter *itl1* določa zgornjo mejo za število navadnih Newton-Raphsonovih iteracij.

Faktor dušenja se samodejno prilagaja tako, da se v enem koraku nobena veličina iz vektorja x ne spremeni za več kot $tol/sollim$. Parameter *sollim* je parameter simulatorja. Vrednost tol se izračuna iz relativne natančnosti, ki jo določa parameter *reltol* in absolutne tokovne in napetostne natančnosti, ki jo določata parametra *abstol* in *vntol*, po formuli $tol = \max(x_{i+1}^j, x_i^j) \cdot reltol + abstol$ (če je j-ta komponenta vektorja x tok), oziroma $tol = \max(x_{i+1}^j, x_i^j) \cdot reltol + vntol$ (če je j-ta komponenta vektorja x napetost).

Dušena Newton-Raphsonova metoda se lahko uporabi, če je rešitev že blizu končne rešitve, Newton-Raphsonov postopek pa kljub temu ne konvergira. SPICE preveri konvergenco z dodatnimi iteracijami Newton-Raphsonovega postopka, ki sledijo iteraciji v kateri simulator izpolni konvergenčni kriterij. Opcija *noconviter* izklopi to preverjanje.

Tranzientna analiza s postopnim vklopom virov
Pri tem pomagalu se postavijo vsi neodvisni viri in začetno stanje vezja na 0. Sledi analiza v časovnem prostoru ki postopoma dviga vrednosti neodvisnih virov. Končna točka te analize se uporabi kot začetni poskus za navadno analizo delovne točke.

Opcija *srclmaxiter* določa zgornjo mejo za število izračunanih časovnih točk. Če se v *srclconviter* zaporednih časovnih točkah vrednosti izračunanih veličin ne razlikujejo za več kot tol, SPICE predpostavi, da je dosegel stacionarno stanje. Parameter *srclriseiter* pove koliko časovnih točk traja, da se vrednosti neodvisnih virov postavijo na vrednost, ki se uporablja v analizi delovne točke. Če je podana opcija *srclrisetime*, neodvisni viri dosežejo vrednosti, ki se uporabljajo v analizi delovne točke, ob času *srclrisetime*. Celotno trajanje tranzientne analize je omejeno s časom *srclmaxtime*, korak pa je navzdol omejen s parametrom *srclminstep*.

Kapacitivnostno korakanje

Če začetni poskus, ki ga da tranzientna analiza s postopnim vklopom virov, ne vodi do konvergence analize delovne točke, SPICE ponuja še eno možnost. Med vsako vozlišče in maso se dodajo kapacitivnosti, ki jih določa parameter *cmin*. Nato se ponovi tranzientna analiza s postopnim vklopom virov. Če tudi ta ne da primernega začetnega poskusa za analizo delovne točke, se vrednost kapacitivnosti poveča za *cmin*. Postopku pravimo kapacitivnostno korakanje. Postopek se zaključi po *cminsteps* korakih. Vrednost 0 parametra *cminsteps* izklopi kapacitivnostno korakanje.

Izbira konvergenčnih pomagal
Brez dodatnih opcij so vsa konvergenčna pomagala vklopljena in se uporabijo v primeru težav s konvergenco. Prevodnostno korakanje in korakanje enosmerne vrednosti neodvisnih virov izklopimo tako, da postavimo parametra *gminstep* in *srcsteps* na 0. Tranzientno analizo s postopnim vklopom virov izklopimo tako, da podamo opcijo *nosrclift*. Če pri tranzientni analizi s postopnim vklopom virov želimo takoj začeti s kapacitivnostnim korakanjem, podamo opcijo *noinitsrcl*. Opcija *noopiter* preskoči navadno analizo delovne točke in kar takoj začne s konvergenčnimi pomagali. Vrstni red uporabe konvergenčnih pomagal določajo opcije *srcspriority*, *gminpriority* in *srclpriority*. Njihova vrednost je lahko 1, 2 ali 3 in določa vrstni red, v katerem se bodo ta pomagala zvrstila pri konvergenčnih težavah.

Avtomatsko nastavljanje konvergenčnih opcij
Pri iskanju pravih vrednosti za nastavitve konvergenčnih pomagal lahko vklopimo opciji *opdebug* in *sollimdebug*, ki izpisujeta vrsto sporočil o tam, kaj se dogaja tekom analize delovne točke, oziroma tekom postopka dušene Newton-Raphsonove iteracijske metode.

SPICE samodejno nastavi konvergenčne opcije, kadar naleti na težave s konvergenco. Postopek zna biti dolgotrajen, zato ga lahko izklopimo z opcijo *noautoconv*.

Preverjanje vezja
Zaradi zaokrožitev in prelivov pri računanju s števili s plavajočo vejico se v matriki prevodnosti in na desni strani sistema enačb lahko pojavijo neskončno velike vrednosti ($+\infty$, $-\infty$) oziroma nedoločene vrednosti (NaN). Če podamo opcijo *matrixcheck* SPICE sproti preverja matriko in desno stran in zamenja $+\infty$, $-\infty$, in NaN z 10^{100}, -10^{100} in 0. Opcija predstavlja rešitev v skrajni sili.

SPICE pred vsako analizo preveri ali imajo vsa vozlišča enosmerno pot do mase. Če kako vozlišče nima enosmerne poti do mase, SPICE izpiše opozorilo in ne zažene zahtevane analize. Opcija *nofloatnodescheck* izklopi preverjanje poti enosmernih signalov do mase.

4.9 Parametri simulatorja (`.options`)

Nastavitve, ki so označene z zvezdico (*) podajamo brez vrednosti

```
.options noautoconv
```

Ostale nastavitve podajamo z vrednostmi, kot na primer

```
.options reltol=1e-6 abstol=1e-10
```

V tabelah 4.3, 4.4, 4.5, 4.6, 4.7 in 4.8 so naštete in na kratko opisani parametri simulatorja.

Ime	Privzeto	Pomen
reltol	10^{-4}	relativna natančnost
abstol	10^{-12}	absolutna tokovna natančnost
vntol	10^{-6}	absolutna napetostna natančnost
pivtol	10^{-13}	absolutna meja pivota
pivrel	10^{-3}	relativna meja pivota
nsfactor	10^{10}	vsiljevalni faktor začetne rešitve delovne točke
nssteps	1	število iteracij v katerih vsiljevalni faktor začetne rešitve delovne točke pade na 0

Tabela 4.3: Osnovne nastavitve natančnosti simulatorja.

Ime	Privzeto	Pomen
itl1	100	zgornja meja za število Newton-Raphsonovih iteracij pri analizi delovne točke (OP)
itl2	50	zgornja meja za število Newton-Raphsonovih iteracij na točko pri analizi odvisnosti delovne točke (DC)
itl3, *imin*	3	zgornja meja za število Newton-Raphsonovih iteracij do katere še povečamo časovni korak pri tranzientni analizi (velja za določanje časovnega koraka s štetjem iteracij)
itl4, *imax*	8	zgornja meja za število Newton-Raphsonovih iteracij do katere pri tranzientni analizi še ne zmanjšamo časovnega koraka (velja za določanje časovnega koraka s štetjem iteracij)
itl5	-	nima pomena
itl6, *srcsteps*	10	glej opcijo *srcsteps* pri konvergenčnih pomagalih
pziter	200	število iteracij v analizi polov in ničel

Tabela 4.4: Omejitev števila iteracij.

Ime	Privzeto	Pomen
trtol	7	varnostni faktor lokalne integracijske napake
chgtol	10^{-14}	abs. natančnost naboja
relq	0.01	rel. toleranca naboja
ltereltol	0.01	rel. toleranca za tokovni vir v časovno diskretiziranem modelu kapacitivnosti
lteabstol	0.01	abs. toleranca za tokovni vir v časovno diskretiziranem modelu kapacitivnosti
newtrunc	*	izklopi postopek prediktor-korektor za določanje lokalne integracijske napake, izklopi opcijo *nopredictor*
nopredictor	*	izklopi postopek prediktor-korektor za predikcijo vozliščnih napetosti, izklopi opcijo *newtrunc*
method	trap	integraciski postopek *trap* ... trapezni postopek *gear* ... Gearov postopek
maxord	2	zgornja meja za red integracijskega postopka, 1...2 za trapezni in 1...6 za Gearov postopek
bypass	1	preskoči elemente, ki se ne spreminjajo

Tabela 4.5: Tranzientna analiza - integracija in lokalna integracijska napaka.

Ime	Privzeto	Pomen
lvltim	1	način določanja časovnega koraka 0 ... štetje iteracij 1 ... dvdt algoritem 2 ... prvotni algoritem iz Berkeleyevega SPICE3
icstep	0	faktor zmanjšanja začetnega časovnega koraka za izračun pravilne rešitve vezja ob času t=0, vrednost 0 izklopi iskanje pravilne rešitve za čas t=0
rmax	5	faktor med izpisnim korakom in zgornjo mejo časovnega koraka
rmin	10^{-9}	faktor med izpisnim korakom in spodnjo mejo časovnega koraka
fs	0.25	faktor med izpisnim korakom in prvim časovnim korakom
ft	0.25	faktor spremembe časovnega koraka, če Newton-Raphsonov postopek ne skonvergira v *itl4* iteracijah

Tabela 4.6: Tranzientna analiza - določanje časovnega koraka

Ime	Privzeto	Pomen
temp	27	temperatura vezja v $^\circ$C oziroma privzeta vrednost parametrs *temp* elementov
tnom	27	privzeta vrednost parametra *tnom* elementov v $^\circ$C
scale	1	enota za dolžino (npr. za 10^{-6} vnašamo vse dimnezije elementov v μm)
dcap	1	izbira enačbe za kapacitivnost diode (trenutno na voljo samo 1)
defl	10^{-4}	privzeta dolžina kanala MOS tranzistorja
defw	10^{-4}	privzeta širina kanala MOS tranzistorja
defad	0	privzeta površina ponora MOS tranzistorja
defas	0	privzeta površina izvora MOS tranzistorja
defpd	0	privzeti obseg ponora MOS tranzistorja
defps	0	privzeti obseg izvora MOS tranzistorja
defnrd	0	privzeto število kvadratkov ponora MOS tranzistorja
defnrs	0	privzeto število kvadratkov izvora MOS tranzistorja
defmodcheck	0	privzeta vrednost parametra *check* modelov BSIM3v3 MOS tranzistorjev, 0 izklopi preverjanje parametrov modela
definstcheck	0	privzeta vrednost parametra *check* BSIM3v3 MOS tranzistorjev, 0 izklopi preverjanje parametrov elementa
oldlimit	*	vklopi stari način omejevanja napetosti MOS tranzistorjev
badmos3	*	vklopi stari model MOS $level = 3$ tranzistorja
trytocompact	*	poenostavi računanje izgubnih linij
keepopinfo	*	shrani rezultate analize delovne točke, ki se izvrši pred malosignalno analizo (`ac` in `pz`)

Tabela 4.7: Temperatura, modeli in shranjevanje rezultatov.

Ime	Privzeto	Pomen
opdebug	*	vklopi obveščanje uporabnika o rabi konvergenčnih pomagal
sollimdebug	*	vklopi obveščanje uporabnika o korakih dušenega Newton-Raphsonovega postopka
matrixcheck	*	vklopi preverjanje in korekcijo enačb v primeru neskončnih oziroma nedefiniranih vrednosti
nofloatnodescheck	*	izklopi preverjanje plavajočih vozlišč
noautoconv	*	izklopi algoritem za samodejno nastavitev konvergenčnih pomagal
voltagelimit	10^{30}	omejitev napetosti na pn spojih
cshunt	0	kapacitivnost, ki se veže med vozlišča in maso
rshunt	∞	upornost, ki se veže med vozlišča in maso
gminpriority	1	prioriteta za algoritem prevodnostnega korakanja
srcspriority	2	prioriteta za algoritem korakanja neodvisnih virov
srclpriority	3	prioriteta za tranzientno analizo s postopnim vklopom virov
noopiter	*	preskoči navadno `op` analizo in takoj uporabi konvergenčna pomagala
nosrclift	*	izklopi tranzientno analizo s postopnim vklopom virov
noinitsrcl	*	preskoči korak brez kapacitivnosti na vozliščih pri tranzientni analizi s postopnim vklopom virov
gmin	10^{-12}	najmanjša prevodnost v vezju
gmindc	10^{-12}	najmanjša prevodnost v vezju za enosmerno analizo (`op`, `dc`, `tf`)
cmin	10^{-12}	korak pri kapacitivnostnem korakanju
sollim	10	delež tolerance, ki omejuje korak dušene Newton-Raphsonove iteracije
sollimiter	10	zgornja meja za število korakov dušene Newton-Raphsonove iteracije
noconviter	0	izklop preverjanja konvergence Newton-Raphsonovega postopka (neničelna vrednost izklopi preverjanje)
gminsteps	10	zgornja meja za število korakov algoritma prevodnostnega korakanja
srcsteps, *itl6*	10	zgornja meja za število korakov algoritma korakanja neodvisnih virov
cminsteps	2	zgornja meja za število korakov kapacitivnostnega korakanja
srclmaxiter	5000	zgornja meja za število časovnih korakov tranzientne analize s postopnim vklopom virov
srclconviter	50	število časovnih korakov znotraj katerih se mora rešitev ustaliti, da se tranzientna analiza s postopnim vklopom virov zaključi
srclriseiter	50	število časovnih korakov znotraj katerih neodvisni viri dosežejo svoje polne vrednosti pri tranzientni analizi s postopnim vklopom virov
srclrisetime	0	čas po katerem neodvisni viri dosežejo svoje polne vrednosti pri tranzientni analizi s postopnim vklopom virov, vrednost 0 vklopi dviganje na osnovi števila časovnih korakov in opcije *srclriseiter*
srclmaxtime	10	čas po katerem se konča tranzientna analiza s postopnim vklopom virov

Tabela 4.8: Konvergenčna pomagala.

5

Jezik NUTMEG

NUTMEG je jezik, v katerem pišemo ukaze v ukazno okno oziroma v ukazni blok. Izrazi jezika NUTMEG se med drugim uporabljajo pri parametrizaciji podvezij. NUTMEG je namenjen opisu zahtevnejših analiz vezja. Jezik tudi omogoča, da avtomatiziramo ponavljajoča se opravila, kot naprimer analizo vezja za različne vrednosti parametrov.

5.1 Zagon, substitucija in izpis v datoteko

Ob zagonu SPICE OPUS prebere vsebino datoteke `spinit` in izvede ukaze jezika NUTMEG, ki se nahajajo v njej. Datoteka se nahaja v mapi *INSTALL/lib/scripts* (*INSTALL/lib/spiceopus/scripts* pod sistemi LINUX). `INSTALL` pomeni ime mape, kamor smo namestili SPICE OPUS.

Opis vezja ni občutljiv na velike in male črke, jezik NUTMEG pa je, zato je priporočljivo vse ukaze in njihove argumente pisati z malimi črkami.

Izrazno substitucijo si poglejmo kar na primeru.

```
echo 4
echo {2*2}
```

Oba ukaza izpišeta 4. Vendar prvi izpiše 4 dobesedno kot podani niz, drugi primer pa najprej izračuna vrednost izraza v zavitih oklepajih, ga pretvori v niz in nato izpiše. Izrazno substitucijo uporabljamo pri ukazih, ki zahtevajo konkretne vrednosti in ne dovoljujejo podajanje izrazov.

```
* To ne gre, ker ac analiza zahteva konkretne frekvence
ac dec 10 fstart fstart*1e4
* To pa gre
ac dec 10 {fstart} {fstart*1e4}
```

Če je rezultat izraza vektor, se v niz pretvori po naslednjem vzorcu: `(`*v1*`;`*v2*`;`*v3*`;...)`. Tako se vektor s komponentama 5 in 9 pretvori v `(5;9)`. Taka pretvorba omogoča, da dobljeni niz uporabimo v nekem izrazu. Tako na primer `1+(0;1)` da enak rezultat, kot `1+{(0;1)}`.

Substitucija spremenljivk temelji na podobni ideji, kot izrazna substitucija (pretvorba vrednosti v niz). Spremenljivke med drugim tudi določajo obnašanje NUTMEGA. V znakovnih nizih jih uporabimo tako, da pred njihovo ime postavimo znak `$`, ime spremenljivke pa damo v oklepaje. Tako na primer ukaz

`echo $(prompt)`
izpiše pozivnik programa SPICE OPUS, ki ga določa spremenljivka `prompt`. Če znak `$` nastopa znotraj enojnih narekovajev, se substitucija ne opravi.

Izpis v datoteko dosežemo z znakom > oziroma nizom >>, ki mu sledi ime datoteke. Oba v datoteko zapišeta vse, kar bi vrstica izpisala v ukazno okno. Prvi pri tem ustvari podano datoteko na novo, drugi pa ji izpis le pripne na koncu datoteke.

Primer:

```
* Ustvari datoteko moja.txt
echo Pozdravljeni > moja.txt
* Dodaj rezultat izraza 2+2 v datoteko moja.txt
echo {2+2} >> moja.txt
```

5.2 Osnovni ukazi

Ti ukazi služijo osnovni komunikaciji med okoljem SPICE OPUS in uporabnikom.

Nalaganje vezja in zagon ukazov v ukaznem bloku (source)

`source` *ime_datoteke*

Pogosto uporabljamo skrajšano obliko ukaza kjer ne navedemo besedice `source`. Pot, v kateri SPICE išče datoteke, je podana v spremenljivki `sourcepath`, katere privzeta vrednost je `(. INSTALL/lib/scripts)` oziroma `(. INSTALL/lib/spiceopus/scripts)` pod sistemi LINUX. Pot spremenimo z ukazom `set sourcepath=( $(sourcepath)` *dir1 dir2* `... )`.
Za oklepajem in pred zaklepajem mora biti obvezno presledek. Če ne želimo vključiti prejšnje vrednosti poti v novo, izpustimo `$(sourcepath)`.

Podrobnosti v zvezi z nalaganjem vezja in ukazom `source` se nahajajo v podpoglavju o vezjih.

Primer:

```
* Nalozi vezje mojevezje.cir.
source mojevezje.cir
* Tudi tako gre.
mojevezje.cir
```

Izhod iz okolja SPICE OPUS (quit)

Ob izhodu SPICE v nekaterih primerih vpraša, če smo prepričani, da želimo zapustiti okolje SPICE OPUS. Nadležno vprašanje izklopimo s spremenljivko `noaskquit`.

Izpis verzije programa (version)

Ukaz izpiše verzijo programa in datum, kdaj je bila ustvarjena.

Izpis podatkov o simulatorju (siminfo)

`siminfo [analyses] [devices] [nodes] [all]`

Besede `analyses`, `devices` in `nodes` povzročijo izpis seznama analiz, elementov in tipov vozlišč, ki jih ponuja SPICE OPUS. `siminfo all` ima enak učinek, kot `siminfo analyses devices nodes`.

Sprememba in izpis delovne mape (cd)

`cd [`*ime_mape*`]`

Če opustimo ime mape, nas SPICE OPUS pod LINUXom premakne v našo domačo mapo. V sistemu WINDOWS ukaz izpiše opozorilo, da ne najde domače mape uporabnika. Za izpis imena mape, kjer se trenutno nahajamo, služi ukaz `cd .`

Pod sistemom WINDOWS je v poteh, ki jih podajamo okolju SPICE OPUS, potrebno uporabljati znak `/` namesto znaka `\`.

Kopiranje datoteke (copy)

Ukaz naredi kopijo datoteke. Izvorna datoteka ostane nespremenjena.

`copy` *izvorna_datoteka nova_datoteka*

Premikanje datoteke (move)

Ukaz prenese oziroma preimenuje datoteko. Izvorna datoteka pri tem preneha obstajat.

`copy` *izvorna_datoteka nova_datoteka*

Brisanje datoteke (remove)

Ukaz izbriše datoteko.

`remove` *ime_datoteke*

Dostop do sistema pomoči (help)

Ukaz odpre sistem pomoči.

`help`

Po sistemu pomoči se premikamo tako, da vtipkamo zaporedno številko teme, ki nam jo ponuja SPICE OPUS in pritisnemo tipko **Enter**. En nivo više se vrnemo s pritiskom na tipko **Enter**.

Preimenovanje ukazov (alias)

Ustvari drugo ime za uporabniški ukaz.

`alias [`*ime_ukaza* `[`*NUTMEG_ukaz*`]]`

Če navedemo samo parameter *ime_ukaza*, se izpiše ukaz jezika NUTMEG, ki pripada navedenemu uporabniškemu ukazu. Če ne navedemo ničesar, se izpiše seznam vseh uporabniških ukazov in pripadajočih ukazov jezika NUTMEG.

Primeri:

```
* Definira ukaz exit
alias exit quit
* Izpiše definicijo ukaza exit
alias exit
* Izpiše definicije vseh ukazov
```

```
alias
```

Brisanje preimenovanega ukaza (unalias)

Izbriše definicijo uporabniškega ukaza.

`unalias` *ime_ukaza*

Primer:

```
* Izbrise definicijo ukaza exit
unalias exit
```

Izpis seznama vtipkanih ukazov (history)

`history`

Zgodovina ukazov je oštevilčena. Če natipkamo `!`*n*, se izvrši ukaz s številko *n* v zgodovini ukazov. Globino zgodovine ukazov določa spremenljivka `history`.

Izpis porabe pomnilnika, procesorskega časa in statistike simulatorja (rusage)

`rusage [`*kategorija1*`] [`*kategorija2*`] ...`

Na voljo so naslednje kategorije: `space` (poraba pomnilnika), `time` (poraba procesorskega časa), `temp`, `tnom` (vrednosti parametrov *temp* in *tnom* simulatorja), `equations` (število enačb vezja), `totiter` (skupno število iteracij med simulacijo), `accept`, `rejected` (število sprejetih in zavrnjenih časovnih točk med simulacijo), `loadtime`, `reordertime`, `lutime` in `solvetime` (čas porabljen za nalaganje, pivotiranje, LU-razcep in reševanje matrike vezja), `trantime`, `tranpoints`, `traniter`, `trancuriters`, `tranlutime` in `transolvetime` (čas, število točk, število iteracij, število iteracij za zadnjo časovno točko, čas porabljen za LU-razcep in čas porabljen za reševanje sistema enačb pri tranzientni analizi).

Če navedemo besedico `all` je učinek enak, kot če bi navedli vse zgoraj omenjene besede. Če ne navedemo nobene, se izpišejo čas od zadnjega klica `rusage`, čas od zagona programa in poraba pomnilnika.

Primer:

```
* Izpise casa od zagona in zadnjega klica rusage
*  ter porabo pomnilnika
rusage
* Izpise vse informacije
rusage all
* Izpise vrednosti parametrov simulatorja temp in tnom
rusage temp tnom
```

Izpisovanje besedila v ukazno okno (echo)

`echo [`*niz*`]`

Pred izpisovanjem SPICE opravi izrazno substitucijo, substitucijo spremenljivk in odstrani vejice in večkratne presledke. Če tega ne želimo, dele niza, za katere nočemo, da jih SPICE spremeni postavimo v enojne narekovaje. Če parameter *niz* izpustimo, ukaz izpiše prazno vrstico.

Primer:

```
* Izpise: Uporov kondenzatorjev: 4
echo Uporov, kondenzatorjev: {2+2}
* Izpise: Uporov, kondenzatorjev: 4
echo 'Uporov, kondenzatorjev:' {2+2}
* Izpise: Uporov, kondenzatorjev: {2+2}
echo 'Uporov, kondenzatorjev: {2+2}'
```

5.3 Skupine vektorjev

Vsaka analiza ustvari eno ali več skupin vektorjev. Ena od skupin vektorjev v pomnilniku je trenutna skupina. Skupina `const` je zmeraj prisotna in vsebuje konstante kot na primer π (`pi`) in e (`e`). Če nekega vektorja SPICE ne najde v trenutni skupini, ga gre iskat v skupino `const`. Če ga tudi tam ni, dobimo sporočilo o napaki. Vsaka neprazna skupina ima privzeti vektor, ki predstavlja skalo za ostale vektorje. Pri prazni skupini postane prvi ustvarjeni vektor v skupini privzeti vektor. Skupine vektorjev upravljamo z naslednjimi ukazi.

Izbira trenutne skupine vektorjev (**setplot**)

`setplot [`*ime_skupine*`]`

Če imena skupine ne navedemo, se izpiše seznam vseh skupin. Če za *ime_skupine* podamo besedo `new`, se ustvari nova prazna skupina vektorjev. Po končanem ukazu se nahajamo v tej skupini. Ime skupine `previous` nas pomakne do skupine, ki je bila ustvarjena pred trenutno skupino vektorjev. Ime skupine `next` nas pomakne do skupine, ki je bila ustvarjena za trenutno skupino vektorjev.

Primer:

```
* Izbere skupino const
setplot const
* Izpise seznam skupin
setplot
* Ustvari novo prazno skupino vektorjev
setplot new
```

Preimenovanje trenutne skupine vektorjev (**nameplot**)

`nameplot` *ime_skupine*

Skupine `const` ne moremo preimenovati.

Primer:

```
* Ustvari novo prazno skupino vektorjev in jo poimenuje vs1
setplot new
nameplot vs1
```

Brisanje skupine vektorjev (**destroy**)

`destroy [`*ime_skupine*`]`

Če za *ime_skupine* podamo `all`, se izbrišejo vse skupine vetorjev, razen skupine `const`. Če *ime_skupine* izpustimo, se izbriše trenutna skupina vektorjev, skupina pred njo pa postane trenutna skupina.

Primer:

```
* Izbrisi skupino vs1
destroy vs1
* Izbrisi trenutno skupino
destroy
* Izbrisi vse skupine razen const
destroy all
```

Izpis podrobnosti o vektorjih (display)

`display [`*ime_vektorja1*`] [`*ime_vektorja2*`] ...`

Če ne navedemo nobenega vektorja, se izpišejo podrobnosti o vseh vektorjev v trenutni skupini. Privzeti vektor je označen z besedo `default scale`.

Primer:

```
* Izpisi podrobnosti o vseh vektorjih v trenutni skupini
display
* Izpisi podrobnosti o vektorjih time in v(1)
display time v(1)
* Izpisi podrobnosti o vektorjih pi in e iz skupine const
display const.pi const.e
```

Prepisovanje vektorjev iz ene skupine v drugo (copyplot)

`copyplot` *izvorna_skupina ciljna_skupina*

Če ciljna skupina ne obstaja, se ustvari. Če kateri od vektorjev že obstaja v ciljni skupini, se zamenja z vektorjem iz izvorne skupine.

Izbira privzetega vektorja skupine (setscale)

`setscale [`*ime_vektorja*`]`

Če izpustimo ime vektorja se izpiše ime privzetega vektorja v skupini.

Shranjevanje vektorjev v datoteko (write)

Shrani skupino v datoteko.
`write [[`*ime_datoteke*`] [`*izraz1*`] [`*izraz2*`] ...]`

Shrani izraze in privzeto skalo trenutne skupine v datoteko. Če izraze izpustimo, se shranijo vsi vektorji iz trenutne skupine. Če ne navedemo imena datoteke, se uporabi ime določeno s spremenljivko `rawfile`. Privzeta vrednost te spremenljivke je `rawspice.raw`. Spremenimo jo lahko z zagonsko opcijo `-r`. Format datoteke je lahko binaren ali ASCII in ga določa spremenljivka `filetype`. Njena vrednost je lahko `binary` ali `ascii`. Privzeti format je ASCII. Če datoteka že obstaja, jo SPICE nadomesti z novo. Spremenljivka `appendwrite` spremeni obnašanje ukaza `write` tako, da se vsebina dodaja na konec datoteke, če ta že obstaja.

```
* Nastavi binaren format
set filetype=binary
* Shrani vse vektorje trenutne skupine v datoteko podatki.raw
write podatki.raw
```

Nalaganje vektorjev iz datoteke (load)

`load [`*ime_datoteke*`]`

Ustvari novo skupino vektorjev in jo napolni z vektorji iz datoteke. Če ne navedemo imena datoteke, se uporabi ime določeno s spremenljivko `rawfile`, katere privzeta vrednost je `rawspice.raw` (če ni bila podana zagonaska opcija `-r`).

```
* Nalozi vektorje iz datoteke podatki.raw
load podatki.raw
* Poimenuj nastalo skupino vektorjev ns1
nameplot ns1
```

5.4 Vektorji

Vektorji hranijo zaporedje realnih ali kompleksnih vrednosti in predstavljajo rezultate analiz, ki jih opravi simulator.

Izpisovanje vrednosti vektorjev (print)

`print [col|line]` *izraz1* `[`*izraz2*`] ...`

Ukaz izpiše vertikalno tabelo, če podamo besedo `col`. V primeru, da podamo `line`, se vrednosti izrazov izpišejo po vrsticah. Če ne podamo nobene od besed `col` ali `line`, se vektorji izpisujejo v vertikalno tabelo, če ima vrednost kateregakoli od izrazov več kot eno komponento. Pri vertikalnem izpisu se poleg vrednosti izrazov izpišejo še indeksi in skala trenutne skupine vektorjev (kar lahko izklopimo s postavitvijo spremenljivk `noprintindex` in `noprintscale`). Izpis je razdeljen na strani. Če tega nočemo, postavimo ospremenljivko `nobreak`. Velikost strani določata spremenljivki `width` in `length`. Na vrhu vertikalnega izpisa je glava, katere izpis izklopimo s postavitvijo spremenljivke `noprintheader`.

Za kompleksne vektorje se izpiše po en stolpec za realen in imaginaren del. Če zahtevamo vrstični izpis oziroma, če so vrednosti vseh izrazov dolžine 1, se izpišeta realen in imaginaren del ločena z vejico.

Primer:

```
* Izklopi izpisovanje indeksov in delitev na strani
set noprintindex
set nobreak
* Sirina izpisa naj bo 1000 znakov
set width=1000
* Izpise vozliscni napetosti v(1) in v(10)
print v(1) v(10)
```

Ustvarjanje in spreminjanje vektorjev (let)

`let` *vektor=izraz*
`let` *skupina*`.`*vektor=izraz*
`let` *vektor*`[`*komponenta*`]=`*izraz*
`let` *skupina*`.`*vektor*`[`*komponenta*`]=`*izraz*

Ukaz nam omogoča, da v poljubni skupini (oziroma trenutni skupini, če je ne navedemo posebej) spremenimo oziroma ustvarimo vektor. Spremenimo lahko tudi eno samo komponento vektorja. Prva komponenta vektorja ima indeks 0. Spreminjanje komponente vektorja je mogoče le, če vektor že obstaja. Če ne navedemo komponente, vektor pa ne obstaja, se ustvari nov vektor v trenutni oziroma navedeni skupini vektorjev. Če vektor že obstaja, ga novoustvarjeni vektor nadomesti.

Primeri:

```
* Ustvari vektor a in ga nastavi na 10
let a=10
* Spremeni komponento z indeksom 8 vektorja b na 99
let b[8]=99
* Ustvari vektor c v skupini ac1
let ac1.c=10
```

Brisanje vektorjev (unlet)

`unlet` *ime_vektorja1* [*ime_vektorja2*] ...

Primer:

```
* Izbrise vektorja a in b
unlet a b
```

Iskanje interpoliranega indeksa za dano vrednost v vektorju (merjenje) (cursor)

Odčitavanje vrednosti vektorja pri interpoliranem indeksu

`cursor` *ime_kurzorja smer* [*izraz1 izraz2* [*izraz3* [*odvod*]]]

Kurzor je vektor z eno samo komponento in predstavlja realni indeks v nekem vektorju (indeks je običajno celo število, tu pa je realno število). Parameter *smer* določa smer premikanja kurzorja in je `left` (pomikanje k nižjim indeksom) ali `right` (pomikanje k višjim indeksom). S samo tremi parametri ukaz `cursor` postavi vektor *ime_kurzorja* na vrednost 0 oziroma indeks zadnjega elementa v vrednosti izraza *izraz1*, odvisno od tega ali je podana *smer* `left` ali `right`.

S petimi parametri ukaz najprej ovrednosti izraz *izraz1*, da dobi preiskovani vektor in izraz *izraz2*, da dobi vrednost iskanega nivoja. Nato v preiskovanem vektorju išče vrednost, ki je enaka nivoju. Pri iskanju se pomika od indeksa shranjenega v vektorju *ime_kurzorja* proti nižjim (`left`) ali višjim (`right`) indeksom. Vrednost, ki se shrani v vektorju *ime_kurzorja* je realni indeks, ki ob linearni interpolaciji preiskovanega vektorja da vrednost enako iskanemu nivoju.

Linearno interpolacijo z realnimi indeksi dobimo s pomočjo operatorja `[%`.

```
* Shrani vrednost vektorja a pri indeksu 9.8 v vektor b
let b=a[%9.8]
```

Primer (Poišči čas točke, ko časovni potek signala `out` zadnjič doseže vrednost 2.5.):

```
* Ustvari kurzor c
let c=0
* Pomakni ga do najvisjega indeksa v vektorju out
cursor c right out
* Pomikaj se proti nizjim indeksom v vektorju out,
*   dokler ne najdes nivoja 2.5
cursor c left out 2.5
* Odcitaj pripadajoci cas iz vektorja time
let cas=time[%c]
```

Pogosto si želimo najti ne prvo mesto v nekem vektorju, ampak šele n-to. Včasih si tudi želimo, da bi pri štetju presečišč v vektorju upoštevali samo tista, kjer vrednost raste ali pa pada z naraščajočim indeksom. Prvo možnost

dobimo tako, da podamo *izraz3*, katerega vrednost pove katero presečišče po vrsti nas zanima. Drugo možnost nam ponuja parameter *odvod*, katerega vrednost je lahko `rising` (upošteva samo presečišča z naraščujočimi vrednostmi), `falling` (upošteva samo presečišča s padajočimi vrednostmi) ali `any` (upošteva vsa presečišča). Če izpustimo parameter *odvod*, SPICE privzame vrednost `any`. Če izpustimo *izraz3*, SPICE predpostavi, da iščemo prvo presecišče.

Primer (Poišči čas, ko signal `out` petič prečka nivo 2.5. Upoštevaj samo tista presečišča, kjer signal narašča):

```
* Ustvari kurzor c,
*   postavi ga na zacetek poteka signala out
let c=0
* Pomikaj se proti visjim indeksom v vektorju out,
*   dokler ne najdes 5. presecisca z nivojem 2.5.
*   Stej samo presecisca z narascujocim signalom out.
cursor c right out 2.5 5 rising
* Odcitaj pripadajoci cas iz vektorja time
let cas=time[%c]
```

Če SPICE zahtevanega presečišča v vektorju ne najde, nastavi vektor *ime_vektorja* na negativno vrednost.

5.5 Izrazi

Konstante v izrazih so lahko v navadnem formatu (npr. `2`, `2.5`, `25e-6`) ali pa v formatu s SI predponami (npr. `10k`, `9u`, `5Meg`). Vektorje dobimo na več različnih načinov. Če je ime vektorja številka (npr. vozliščna napetost vozlišča `10`), dobimo pripadajoči vektor z `v(`*ime*`)` (npr. `v(10)`). Če se ime vozlišča ne začne s cifro, dobimo pripadajoči vektor z `v(`*ime*`)` ali kar *ime*. Razliko dveh vektorjev lahko krajše zapišemo kot `v(`*ime1*`,`*ime2*`)` (npr. `v(10,20)`). Tokove skozi napetostne vire dobimo preko vektorjev *ime_elementa*`#branch` oziroma z `i(`*ime_elementa*`)` (npr. `v10#branch` ali `i(v10)`).

SPICE OPUS išče vektor v trenutni skupini. Če ga v tej skupini ni, pogleda v skupino, `const`. Če ga tudi tam ni, javi napako. Pred imenom vektorja lahko navedemo tudi ime skupine, v kateri bo SPICE OPUS iskal vektor. To naredimo z naslednjo sintakso: *ime_skupine*`.`*vektor* (npr. `ac1.v(1,2)`, `dcres.i(v1)`).

Operatorji, ki jih pozna SPICE OPUS so našteti v tabeli 5.1.

Aritmetični, logični in primerjalni operatorji zahtevajo, da sta vektorja enako dolga, ali pa da je dolžina enega od vektorjev enaka 1. Operacija se izvaja nad istoležnimi elementi vektorjev oziroma med skalarjem in vsemi elementi vektorja. Operand, ki predstavlja indeks, mora biti skalar. Če je vektor dolg n elementov, je indeks prvega elementa enak 0, zadnjega pa $n - 1$.

Poleg operatorjev nam NUTMEG ponuja tudi funkcije, ki so naštete v tabeli 5.2.

Vse funkcije delajo enako z vsemi komponentami vektorjev, razen sledečih izjem.

Funkcija `unwrap(`a`)` popravi vektor, ki vsebuje potek faze tako, da ni skokov za 2π. Uporabljamo jo v kombinaciji s funkcijo `ph(`a`)`.

Funkcija `rnd(`a`)` vrne vektor z naključnimi komponentami, ki so med 0 in pripadajočimi komponentami a zmanjšanimi za 1. Za kompleksen vektor a sta realni in imaginarni del komponent vrnjenega vektorja med 0 in realnim in imaginarnim delom pripadajočih komponent vektorja a zmanjšanih za 1.

Funkciji `mean(`a`)` in `sum(`a`)` vrneta skalar. Funkciji `min(`a`)` in `max(`a`)` za kompleksne vektorje vrneta absolutno najmanjšo oziroma največjo komponento.

Funkcije zaokroževanja `floor(`a`)`, `ceil(`a`)` in `round(`a`)` delujejo neodvisno na realnem in imaginarnem delu

Sintaksa	Primer	Pomen
a+*b*	`v(10)+1`	vsota vektorjev *a* in *b*
a−*b*	`v(5)-v(6)`	razlika vektorjev *a* in *b*
a⋆*b*	`v(2)*i(v1)`	produkt vektorjev *a* in *b*
a/*b*	`v(2)/v(1)`	kvocient vektorjev *a* in *b*
a%*b*	`8%3`	ostanek pri deljenju vektorjev *a* in *b*
a^*b*	`v(2)^2`	vektor *a* potenciran na vektor *b*
−*a*	`-v(5)`	operator negativnega predznaka
a `eq` *b*	`v(10) eq v(20)`	je enako
a `ne` *b*	`v(10) ne v(20)`	ni enako
a `gt` *b*	`v(10) gt v(20)`	večje
a `lt` *b*	`v(10) lt v(20)`	manjše
a `ge` *b*	`v(10) ge v(20)`	večje ali enako
a `le` *b*	`v(10) le v(20)`	manjše ali enako
a `and` *b*	`m1 and m2`	logični in
a `or` *b*	`m1 or m2`	logični ali
`not` *a*	`not m1`	logična negacija
a,*b*	`-4,9`	sestavi kompleksen vektor iz realnega dela *a* in imaginarnega dela *b*
a;*b*	`2;8;8`	združi vektorja *a* in *b*
a[*b*]	`v(10)[0]`	komponenta z indeksom *b* vektorja *a*
a[*b*,*c*]	`v(10)[0,5]`	podvektor vektorja *a* z indeksi od *b* do *c*
a[%*b*]	`v(10)[%2.8]`	odčita vektor *a* z uporabo linearne interpolacije pri realnem indeksu *b*

Tabela 5.1: Operatorji v NUTMEGU.

vektorja. Funkcija `round(`*a*`)` prišteje k argumentu 0.5 in rezultat zaokroži navzdol.

Funkciji `vector(`*a*`)` in `unitvec(`*a*`)` vrneta vektor podane dolžine *a*. Dolžina mora biti realen skalar. Funkcija `length(`*a*`)` vrne realen skalar, ki predstavlja število komponent vektorja *a*.

Funkcija `area(`*a*`)` izračuna površino vezja. Trenutno upošteva samo MOS tranzistorje po formuli $m \cdot (w \cdot l + ad + as)$. Podamo ji celoštevilsko vrednost, ki predstavlja številko vezja, za katerega nas zanima površina. Površino trenutnega vezja dobimo tako, da podamo 0.

Funkciji `timer(`*a*`)` in `clock(`*a*`)` vrneta procesorski čas (čas, ki ga računalnik porabi znotraj programa SPICE OPUS) in čas ure (čas merjen od nekega fiksnega trenutka) v sekundah zmanjšana za vrednost argumenta.

Funkcija `interpolate(`*a*`)` interpolira vektor *a* iz njegove skale (privzetega vektorja skupine iz katere prihaja *a*) na skalo (privzeti vektor) trenutne skupine vektorjev. Stopnjo interpolacijskega polinoma določa spremenljivka `polydegree`.

Funkciji `deriv(`*a*`)` in `integrate(`*a*`)` vrneta odvod vektorja *a* po njegovi skali oziroma določeni integral vektorja po njegovi skali od začetka skale naprej.

Definicija uporabniške funkcije (define)

`define [`*ime*`(`*p1*`, `*p2*`, ...)=(`*izraz*`)]`

Definira novo funkcijo z imenom *ime* in argumenti *p1*, *p2*, ... Funkcijo opisuje *izraz*. Če izpustimo *ime*, parametre in *izraz*, se izpišejo vse definirane funkcije.

Nekatere funkcije so že vgrajene v SPICE OPUS. Definirali bi jih z naslednjimi ukazi

```
define vr(x,y)  re((v(x))-(v(y))) \\
define vr(x)    re(v(x)) \\
define vp(x,y)  ph((v(x))-(v(y))) \\
define vp(x)    ph(v(x)) \\
define vm(x,y)  mag ((v(x))-(v(y))) \\
define vm(x)    mag (v(x)) \\
define vi(x,y)  im ((v(x))-(v(y))) \\
define vi(x)    im (v(x)) \\
define vdb(x,y) db ((v(x))-(v(y))) \\
define vdb(x)   db (v(x))
```

Brisanje definicije uporabniške funkcije (undefine)

`undefine` *ime*

Izbriše definicijo funkcije.

Primer:

```
* Brisi definicijo funkcije rampa
undefine rampa
```

Sintaksa	Primer	Pomen
`abs(`*a*`)`	`abs(v(1))`	absolutna vrednost kompleksnega vektorja *a*
`mag(`*a*`)`	`mag(v(1))`	absolutna vrednost kompleksnega vektorja *a*
`magnitude(`*a*`)`	`magnitude(v(1))`	absolutna vrednost kompleksnega vektorja *a*
`ph(`*a*`)`	`ph(v(1))`	faza kompleksnega vektorja *a*
`phase(`*a*`)`	`phase(v(1))`	faza kompleksnega vektorja *a*
`unwrap(`*a*`)`	`unwrap(p1)`	popravi potek faze tako, da je brez prelomov
`j(`*a*`)`	`j(v(1))`	produkt kompleksnega vektorja *a* in imag. enote
`real(`*a*`)`	`real(v(1))`	realen del kompleksnega vektorja *a*
`re(`*a*`)`	`re(v(1))`	realen del kompleksnega vektorja *a*
`imag(`*a*`)`	`imag(v(1))`	imaginaren del kompleksnega vektorja *a*
`im(`*a*`)`	`im(v(1))`	imaginaren del kompleksnega vektorja *a*
`db(`*a*`)`	`db(v(1))`	vrednost izraza `20 log10(abs(`*a*`))`
`log(`*a*`)`	`log(2.5)`	desetiški logaritem kompleksnega vektorja *a*
`log10(`*a*`)`	`log10(2.5)`	desetiški logaritem kompleksnega vektorja *a*
`ln(`*a*`)`	`ln(2.5)`	naravni logaritem kompleksnega vektorja *a*
`exp(`*a*`)`	`exp(9)`	e na kompleksni vektor *a*
`sqrt(`*a*`)`	`sqrt(16)`	kvadratni koren kompleksnega vektorja *a*
`sin(`*a*`)`	`sin(pi/2)`	sinus kompleksnega vektorja *a*
`cos(`*a*`)`	`cos(pi/2)`	kosinus kompleksnega vektorja *a*
`tan(`*a*`)`	`tan(pi/4)`	tangens kompleksnega vektorja *a*
`atan(`*a*`)`	`atan(1.4142)`	arkus tangens kompleksnega vektorja *a*
`norm(`*a*`)`	`norm(v(1))`	vektor *a* deljen z abs. največjo komponento
`rnd(`*a*`)`	`rnd(100)`	naključen celoštevilski vektor
`pos(`*a*`)`	`pos(-9)`	vrednost izraza `re(`*a*`) gt 0`
`mean(`*a*`)`	`mean(v(9))`	srednja vrednost kompleksnega vektorja *a*
`sum(`*a*`)`	`sum(c1)`	vsota komponent kompleksnega vektorja *a*
`min(`*a*`)`	`min(v(10))`	najmanjša komponenta vektorja *a*
`max(`*a*`)`	`max(v(10))`	največja komponenta vektorja *a*
`floor(`*a*`)`	`floor(a/b)`	navzdol zaokrožen vektor *a*
`ceil(`*a*`)`	`ceil(a/b)`	navzgor zaokrožen vektor *a*
`round(`*a*`)`	`round(a/b)`	zaokrožen vektor *a*
`vector(`*a*`)`	`vector(5)`	realen vektor s komponentami od 0 do $a-1$
`unitvec(`*a*`)`	`unitvec(5)`	realen vektor dolžine a iz samih enic
`length(`*a*`)`	`length(time)`	število komponent vektorja *a*
`timer(`*a*`)`	`timer(t1)`	procesorski čas v sekundah zmanjšan za *a*
`clock(`*a*`)`	`clock(t1)`	urin čas v sekundah zmanjšan za *a*
`area(`*a*`)`	`area(0)`	površina vezja s številko *a* (0 za trenutno vezje)
`interpolate(`*a*`)`	`interpolate(dc1.v(1))`	interpolira vektor na skalo trenutne skupine
`deriv(`*a*`)`	`deriv(v(1))`	odvod vektorja po privzetem vektorju
`integrate(`*a*`)`	`integrate(v(1))`	integral vektorja po privzetem vektorju

Tabela 5.2: Funkcije v NUTMEGU.

5.6 Krmilne strukture

Krmilne strukture določajo potek skozi zaporedje ukazov. NUTMEG pozna pogojne stavke in zanke.

Pogojni stavek (if - elseif - else - end)

`if` *pogojni_izraz1*
ukazi1
`elseif` *pogojni_izraz2*
ukazi2
`...`
`else`
ukazin
`end`

Pogoj, ki ga določa *pogojni_izraz*, je resničen, če je vsaj ena komponenta vrednosti tega izraza različna od 0. Če NUTMEG ne uspe ovrednostiti izraza, se smatra, da je njegova vrednost neresnična. Veji `ifelse` in `else` ter pripadajoče ukaze lahko izpustimo.

Stavek `if-elseif-else-end` izvrši zaporedje ukazov *ukazi1*, če je vrednost izraza resnična. V nasprotnem primeru se preveri pogoj pri naslednjem `elseif` stavku. Če je ta resničen, se izvrši zaporedje ukazov *ukazi2*. Postopek se nadaljuje, dokler ne pride na vrsto veja `else`, katere pripadajoče zaporednje ukazov se izvrši, če noben od podanih pogojev ni bil izpolnjen.

Primer:

```
if a gt 0
  echo Res je.
elseif a lt 0
  echo Ni res.
else
  echo Odloci se ze enkrat.
end
```

Zanka while (while - end)

`while` *pogojni_izraz*
ukazi
`end`

Zanka ob vsakem prehodu preveri vrednost pogojnega izraza. Dokler je ta vrednost resnična, se izvršujejo *ukazi*. Pogoj se torej preverja na začetku zanke.

Primer:

```
* Naredi malosignalno analizo za vrednosti upora R1
*   1k, 2k, 5k, 10k
* Za vsako analizo izrise frekvencni potek ojacenja
setplot new
nameplot ctl
let ctl.cnt=0
let ctl.rval=(1k;2k;5k;10k)
while ctl.cnt lt length(ctl.rval)
  * Spremeni vrednost upornosti r1
```

```
  *  (glej podpoglavje o vezju, ukaz let)
  let @r1[resistance]=ctl.rval[ctl.cnt]
  let ctl.cnt=ctl.cnt+1;
  ac dec 10 1.0 100meg
  plot db(v(out)/v(in))
end
```

Zanka dowhile (dowhile - end)

`dowhile` *pogojni_izraz*
ukazi
`end`

Zanka najprej izvrši ukaze *ukazi*. Nato preveri vrednost pogojnega izraza. Dokler je ta resnična, se vrača nazaj in izvršuje ukaze *ukazi*. Pogoj se torej preverja na koncu zanke.

Primer:

```
* Izpise stevilo 6, ker se pogoj preverja na koncu zanke.
setplot new
nameplot ctl
let ctl.cnt=6
dowhile ctl.cnt le 5
  echo {ctl.cnt}
  let ctl.cnt=ctl.cnt+1;
end
```

Prekinitev zanke (break)

Ukaz prekine izvajanje zanke znotraj katere se nahaja.

Primer:

```
* 3x izpise besedo 'ponavljam'
let cnt=0
while cnt le 5
  if cnt ge 3
    break
  end
  echo ponavljam
  let cnt=cnt+1
end
```

5.7 Spremenljivke

Spremenljivke določajo obnašanje NUTMEGA. V znakovnih nizih jih uporabimo tako, da pred njihovo ime postavimo znak `$`, ime spremenljivke pa damo v oklepaje (substitucija spremenljivk). Tako na primer ukaz `echo $(prompt)` izpiše pozivnik programa SPICE OPUS, ki ga določa spremenljivka `prompt`. Če znak `$` nastopa znotraj enojnih narekovajev, se substitucija ne opravi.

Nastavljanje spremenljivke (set)

`set` [*ime* [=*vrednost*]]

Če izpustimo *vrednost*, se spremenljivka postavi brez vrednosti in se obnaša kot vklopljena zastavica. Če ne navedemo tudi imena spremenljivke, se izpišejo vse spremenljivke. Poleg spremenljivk se izpišejo tudi vrednosti opcij simulatorja, nastavljenih z vrsticami `.options`. Označene so z znakom + pred imenom. Posebne spremenljivke jezika NUTMEG so označene z znakom ⋆ pred imenom.

Ime	Pomen
`appendwrite`	vklopi dodajanje k obstoječi datoteki pri ukazu `write`
`badcktstop`	ne izvrši ukaznega bloka ob nalaganju vezja, če vezje vsebuje napake
`badcktfree`	če vezje vsebuje napake, ga sprosti iz pomnilnika
`color0`	barva ozadja grafov pri ukazu `plot`
`color1`	barva mreže grafov pri ukazu `plot`
`color`*n*	barva $(n-1)$-te krivulje na grafih pri ukazu `plot`
`circuits`	seznam imen vezij v pomnilniku računalnika
`curcirc`	ime trenutnega vezja
`curcirctitle`	prva vrstica opisa trenutnega vezja
`curplot`	ime trenutne skupine vektorjev
`curplotdate`	datum nastanka trenutne skupine vektorjev
`curplotname`	dolgo ime trenutne skupine vektorjev
`curplotscale`	ime privzetega vektorja trenutne skupine vektorjev
`dpolydegree`	stopnja interpolacijskega polinoma za funkcijo `deriv`
`filetype`	tip datoteke za ukaz `write` (`ascii` ali `binary`)
`fourgridsize`	število interpoliranih točk pri ukazu `fourier`
`history`	število ukazov v zgodovini ukazov (za ukaz `history`)
`iplottag`	začetni del oznake grafa, ki ga izrisujemo med analizo (ukaz `iplot`)
`length`	višina strani za ukaz `print`
`linewidth`	debelina črte za ukaz `plot` (0, 1, 2, ...)
`magunits`	enote (`normal`, `db20` ali `db10`) za izrisni način `mag` (ukaz `plot`)

Tabela 5.3: Spremenljivke v NUTMEGU (a-m).

Seznam spremenljivk in njihov pomen je naštet v tabelah 5.3 in 5.4.

Spremenljivko `sourcepath` nastavljamo z naslednjo sintakso:
`set sourcepath=( $(sourcepath)` *dir1 dir2* `... )`.
Za oklepajem in pred zaklepajem mora biti obvezno presledek. Če ne želimo vključiti prejšnje vrednosti poti v novo, izpustimo `$(sourcepath)`.

Primeri:

```
* Ne izpisuj privzetega vektorja
set noprintscale
* Stopnja interpolacijskega polinoma naj bo 2
set polydegree=2
* Enote naj bodo stopinje
set units=degrees
* Nov pozivnik. ! pomeni stevilko ukaza.
set prompt='Moj spice ! >'
```

Brisanje spremenljivke (unset)

`unset` *ime*

Ime	Pomen
nfreqs	število harmonskih komponent pri ukazu fourier
noaskquit	izklopi vprasanja pred izhodom iz programa pri ukazu quit
nobreak	izklopi deljenje izpisa na strani v ukazu print
nogrid	izklopi risanje mreže pri ukazu plot
nosort	izklopi sortiranje vektorjev po abecedi pri ukazu display
noprintheader	izklopi izpisovanje glave pri ukazu print
noprintindex	izklopi izpisovanje indeksov pri ukazu print
noprintscale	izklopi izpisovanje privzetega vektorja pri ukazu print
numdgt	število decimalk pri izpisih
plots	seznam vseh skupin vektorjev
plottype	tip grafov za ukaz plot (line, point ali comb)
pointtype	tip točk za ukaz plot (o, x, +, q, d ali t)
pointsize	velikost točk za ukaz plot (0, 1, 2, ...)
plotwinheight	višina grafičnega okna za ukaz plot
plotwinwidth	širina grafičnega okna za ukaz plot
plotwininfo	vklopi merilno okno za grafe izrisane z ukazom plot
plotautoident	vklopi samodejno identifikacijo krivulj za grafe izrisane z ukazom plot
polydegree	stopnja interpolacijskega polinoma za funkciji integrate(*a*) in interpolate(*a*) in pri ukazih fourier in linearize
prompt	pozivnik programa SPICE OPUS
rndinit	negativno seme za generator naključnih števil
rawfile	privzeto ime datoteke za ukaz write
sourcepath	pot do datotek za ukaz source
units	enota za fazo in kotne funkcije v NUTMEGU (degrees ali radians)
width	širina strani za ukaz print
workdir	ime trenutne mape

Tabela 5.4: Spremenljivke v NUTMEGU (n-z).

Primer:

```
* Izpisuj privzeti vektor
unset noprintscale
```

5.8 Vezja

V tej skupini so ukazi, ki omogočajo nalaganje, izbiro, pregledovanje in spreminjanje vezja.

Nalaganje vezja iz datoteke (source)

`source` *ime_datoteke*

Ukaz je bil opisan že v poglavju o osnovnih ukazih. Poglejmo si samo podrobnosti, ki se tičejo nalaganja vezja.

Ob nalaganju vezje dobi ime oblike cktn in postane trenutno vezje simulatorja. Seznam vseh naloženih vezij je v spremenljivki circuits, ime trenutnega vezja pa v spremenljivki curcirc. Prva vrstica trenutnega vezja je v spremenljivki curcirctitle.

Po nalaganju vezja se izvršijo ukazi iz ukaznega bloka v izvorni datoteki. Priporočljivo je najprej popraviti vezje do te mere, da se naloži brez napak in šele nato dodati ukazni blok. Če je nastavljena spremenljivka `badcktstop`, se ukazni blok ne izvrši, če je v opisu vezja napaka. Če je postavljena spremenljivka `badcktfree`, se vezje, ki vsebuje napako sprosti iz pomnilnika takoj po nalaganju.

Izbira trenutnega vezja (**setcirc**)

`setcirc [`*ime_vezja*`]`

Izbere eno od vezij, ki so trenutno naložena. Vezje postane aktivno, kar pomeni, da se vse operacije na vezju od tega trenutka naprej nanašajo nanj. Če ime vezja izpustimo, se izpiše seznam vseh naloženih vezij.

Primer:

```
setcirc ckt9
```

Preimenovanje trenutnega vezja (**namecirc**)

`namecirc` *ime_vezja*

Primer:

```
* Nalozi prvo vezje.
source prvo.cir
* Poimenuj ga vezje1.
namecirc vezje1
* Nalozi drugo vezje.
source drugo.cir
* Nastavi, da bo prvo vezje aktivno.
setcirc vezje1
```

Brisanje vezja iz pomnilnika (**delcirc**)

`delcirc` *ime_vezja*

Če je ime vezja izpuščeno, se izbriše trenutno vezje. Če za ime vezja navedemo `all`, se izbrišejo vsa vezja iz pomnilnika.

Vzpostavitev prvotnega stanja vezja (**reset**)

`reset`

Vezje postavi v stanje, v katerem se je nahajalo neposredno po nalaganju vezja.

Izpis podatkov o trenutnem vezju (**listing**)

```
listing deck
listing physical
listing logical
listing hierarchy
listing sub [ime1] [ime2] ...
listing subdef [ime1] [ime2] ...
listing global
```

Besedi `deck` in `physical` povzročita izpis vsebine datoteke iz katere je bilo prebrano vezje (prva brez številk vrstic, druga pa s številkami vrstic). Beseda `logical` povzroči izpis opisa vezja in nastavitev simulatorja. Vsebina ukaznega bloka se ne izpiše.

`listing hierarchy` izpiše hierarhijo datotek vključenih v vrsticah `.include` in `.lib`.

`listing sub` izpiše seznam vseh podvezij. Če želimo podrobnejše informacije o enem ali večih podvezjih, navedemo njihova imena za besedo `sub`. `listing subdef` izpiše seznam definicij vseh podvezij. Če želimo podrobnejše informacije o konkretnih definicijah, navedemo njihova imena za besedo `subdef`. Glavno vezje ima ime `xtopinst_`, njegova definicija pa `topdef_`.

`listing global` izpiše seznam vseh globalnih vozlišč v vezju.

Izpis podatkov o skupini elementov trenutnega vezja (show)

`show` *ime1* [*ime2*] `...` [`:` *p1* [*p2*] `...`]
`show all` [`:` *p1* [*p2*] `...`]
`show` *ime1* [*ime2*] `...` [`:` `all`]
`show all` [`:` `all`]

Pred dvopičjem naštejemo s presledki ločena imena elementov, ki nas zanimajo. Za dvopičjem naštejemo s presledki ločena imena parametrov, ki nas zanimajo. Seznam imen elementov lahko nadomestimo z besedo `all`, ki povzroči izpis vseh elementov. Podobno lahko tudi seznam imen parametrov nadomestimo z besedo `all`, ki povzroči izpis vseh parametrov.

Če izpustimo seznam parametrov, se izpišejo le najpomembnejši parametri za vsak element.

Elementi imajo poleg parametrov, ki jih lahko podajamo, tudi parametre, ki jih lahko samo beremo. Ti parametri hranijo informacije o delovni točki in lastnostih elementa v delovni točki. Tako lahko izvemo veliko informacij o elementu, če izvršimo ukaz `show` *ime* `:` `all`. Za upor na primer lahko izvemo tok (*i*) in moč (*p*) v delovni točki.

Primeri:

```
* Vrednosti r in p za r1 in r2
show r1 r2 : r p
* Vrednosti vseh parametrov za r5
show r5 : all
* Vrednost parametra m za vse elemente
show all : m
* Vrednosti vseh parametrov za vse elemente
show all : all
* Vrednosti najpomembnejsih parametrov za vse elemente
show all
```

Izpis podatkov o skupini modelov trenutnega vezja (showmod)

`showmod` *ime1* [*ime2*] `...` [`:` *p1* [*p2*] `...`]
`showmod all` [`:` *p1* [*p2*] `...`]
`showmod` *ime1* [*ime2*] `...` [`:` `all`]
`showmod all` [`:` `all`]

Ukaz je podoben ukazu `show`, le da izpisuje parametre modelov. Za ime lahko podamo kar ime elementa (npr. `r1`), kar povzroči izpis parametrov modela, ki ga uporablja element `r1`. Če navedemo samo črko (npr. `q`), se izpišejo vsi modeli bipolarnih tranzistorjev. Če želimo informacije o nekem konkretnem modelu, postavimo pred njegovo ime znak # (tako `#rmod` izpiše informacije o modelu `rmod`).

Primeri:

```
* Vrednosti parametrov rsh in tc1 vseh modelov uporov
showmod r : rsh tc1
* Vrednosti vseh parametrov modela, ki ga uporablja q1
showmod q1 : all
* Vrednost vseh parametrov modela rmod
show #rmod : all
```

Spreminjanje vrednosti parametrov elementov in modelov trenutnega vezja (let)

Uporaba vrednosti parametrov elementov in modelov v izrazih

`let` *ime=izraz*

Za dostop do parametrov elementov in modelov uporabljamo posebno sintakso za *ime* na levi strani enačaja.

`@`*element*`[`*parameter*`]` spremeni vrednost parametra *parameter* elementa *element* v vrednost izraza *izraz*. Če je parameter elementa vektorski, so njegova nova dolžina in nove vrednosti komponent določene s komponentami vrednosti izraza. Na ta način lahko spreminjamo tudi parametre podvezij (ime elementa se v tem primeru začne s črko `x`). Če želimo spremeniti samo eno komponento vektorskega parametra uporabimo sintakso
`@`*element*`[`*parameter*`][`*n*`]`

Primeri:

```
* Nastavi parameter r upora r10 na 100k
let @r10[r]=100k
* Nastavi parameter k podvezja xatn1 na 4
let @xatn1[k]=4
* Nastavi vektorski parameter sin vira vs na (0;5;1k)
let @vs[sin]=(0;5;1k)
* Nastavi amplitudo sinusnega vira vs na 2.5
let @vs[sin][1]=2.5
```

`@@`*model*`[`*parameter*`]` oziroma `@@`*model*`[`*parameter*`][`*n*`]` spremeni vrednost parametra oziroma komponente parametra *parameter* modela *model* v vrednost izraza *izraz*. Če za *model* podamo ime definicije podvezja, se ustrezen parameter spremeni za vsa podvezja, ki uporabljajo to definicijo. Priporočljivo je, da na ta način spreminjamo le tiste parametre, ki imajo privzeto vrednost in ta ni navedena pri nobenem podvezju, ki uporablja definicijo *model*.

Primeri:

```
* Nastavi parameter tc1 modela rmod na 1e-4
let @@rmod[tc1]=1e-4
* Nastavi parameter k na 4 za vsa podvezja,
*   ki uporabljajo definicijo attn
let @@attn[k]=4
```

Vrednosti parametrov elementov in modelov lahko uporabljamo tudi v izrazih. Uporabimo enako sintakso, kot za *ime* na levi strani enačaja.

Primer:

```
* Povecaj upornost upora r1 za 5%
let @r1[r]=@r1[r]*1.05
```

Spreminjanje parametrov simulatorja za trenutno vezje (set in unset)

Uporaba vrednosti parametrov simulatorja v izrazih

`set` *ime*`[=`*vrednost*`]`
`unset` *ime*

Parameter simulatorja *ime* se spremeni na vrednost, ki je navedena desno od enačaja. Če je parameter simulatorja logičen (na primer parameter `noautoconv`), ne navajamo enačaja in vrednosti. V tem primeru z ukazom `set` parameter postavimo. Pobrišemo ga z ukazom `unset`.

Primeri:

```
* Nastavi parameter reltol simulatorja na 1e-6.
set reltol=1e-6
* Nastavi parameter temp simulatorja na 75.
set temp=75
* Izklopi samodejno nastavljanje konvergencnih pomagal.
set noautoconv
* Vklopi samodejno nastavljanje konvergencnih pomagal.
unset noautoconv
```

Da uporabimo vrednosti parametrov simulatorja, ki so navedeni v opisu vezja v eni od vrstic `.options` oziroma so bili nastavljeni z ukazom `set`, uporabimo substitucijo spremenljivk.

```
* Shrani nastavitev parametra reltol v vektor a.
let a=$(reltol)
```

Če želimo vrednost parametra simulatorja nastaviti s pomočjo izraza, uporabimo aritmetično substitucijo.

```
* Nastavi parameter simulatorja reltol na 0.5e-6.
set reltol={1e-6/2}
```

5.9 Analize in obdelava rezultatov

V to skupino spadajo ukazi, ki izvršujejo analize oziroma izbirajo, kaj bo simulator shranil v skupino vektorjev.

Zagon analize vezja (op, dc, tf, ac, noise, pz, tran)

Ukazi, ki izvršijo analize (`op`, `dc`, `tf`, `ac`, `noise`, `pz` in `tran`), so bili že opisani. Omenimo samo, da vsi ti ukazi ustvarijo po eno skupino vektorjev, razen ukaza `noise`, ki izvrši malosignalno šumno analizo, in ustvari dve skupini vektorjev. Ustvarjena skupina ima zmeraj enako ime, kot ukaz za analizo, ki mu je pripeta zaporedna številka. Številčenje se začne pri 1. Če pobrišemo vse skupine vektorjev, se številčenje vrne nazaj na 1. Tako prva `ac` analiza ustvari skupino `ac1`, druga `ac2`, ...

Prva šumna analiza ustvari skupini `noise1` in `noise2`, druga `noise3` in `noise4`, ... Pri tem je prva skupina od obeh tista, ki hrani šumne spektre. Druga skupina hrani integrale šumnih spektrov. Po končani šumni analizi se nahajamo v drugi skupini. Da pridemo v prvo skupino, ki hrani šumne spektre, natipkamo `setplot previous`. Za vrnitev v skupino z integriranim šumom natipkamo `setplot next`.

Izpis imena vozlišča, ki povzroča težave pri simulaciji (where)

`where`

Omeniti velja, da se izpiše samo eno vozlišče, težave pa se lahko pojavijo pri večih vozliščih hkrati.

Vklop sprotnega izpisovanja izračunanih vrednosti tekom analize (trace)

`trace` *ime1* `[`*ime2*`] ...`

Imena vozlišč navedemo v obliki `v(`*ime_vozlisca*`)`. Če želimo opazovati tok napetostnega vira, to povemo z `i(`*ime_vira*`)`. Izpisovanje vrednosti izklopimo s pomočjo ukaza `delete all`.

Primer:

```
* Vklopi zasledovanje napetosti vozlisca 100
*   in toka vira v9
trace v(100) i(v9)
* Pozeni analizo, ki bo izpisovala vrednosti
tran 1n 100n
* Vzpostavi prvotno stanje
delete all
```

Vklop shranjevanja vrednosti parametrov med analizo (save)

`save` *ime1* `[`*ime2*`] ...`

Ime parametra podamo v enaki obliki, kot to počnemo na levi strani enačaja pri ukazu `let` (torej `@`*ime_elementa*`[`*ime_parametra*`]`). Če podamo vsaj en ukaz `save`, se vozliščne napetosti in tokovi napetostnih virov nehajo shranjevavi. Če želimo shraniti tudi te, podamo pri ukazu `save` še besedo `all`. Prvotno stanje simulatorja (shranjevanje vozliščnih napetosti in tokov napetostnih virov) vzpostavimo z ukazom `delete all`.

Seznam parametrov, ki so na voljo za shranjevanje pri nekem konkretnem elementu, dobimo z ukazom
`show` *ime_elementa* `: all`.

Primer:

```
* Vklopi shranjevanje toka upora r1
save @r1[i]
* Vklopi se shranjevanje toka upora r2,
*   vozliscnih napetosti in tokov napetostnih virov
save @r2[i] all
* Pozeni analizo, ki bo shranjevala vrednosti
tran 1n 100n
* Vzpostavi prvotno stanje
delete all
```

Vklop sprotnega izrisovanja napetosti in tokov med analizo (iplot)

Grafi se izrisujejo med analizo v grafičnem načinu. V konzolnem načinu ukaz nima učinka.
`iplot` *ime1* `[`*ime2*`] ...`

Tudi tukaj navajamo veličine podobno, kot pri ukazu `trace`. Veličine, ki jih navedemo ob enem ukazu `iplot` se izrisujejo na en graf. Torej se med analizo izrisuje toliko grafov, kot je bilo podanih ukazov `iplot`. Grafi dobijo oznake, ki so sestavljene iz vsebine spremenljivke `iplottag` in zaporedne številke. Prvotno stanje simulatorja (brez sprotnega risanja grafov) vzpostavimo z ukazom `delete all`.

Primer:

```
* Izrisuj dva grafa
*   1. graf: napetost vozlisc 1 in 2
*   2. graf: tok napetostnega vira vin
iplot v(1) v(2)
iplot i(vin)
* Pozeni analizo, ki bo risala grafe
tran 1n 100n
* Vzpostavi prvotno stanje
delete all
```

Izpis nastavitev izbranih z ukazi `trace`, `save` in `iplot` (**status**)

`status`

Seznam je oštevilčen. Številke uporabljamo pri ukazu `delete`.

Brisanje nastavitev izbranih z ukazi `trace`, `save` in `iplot` (**delete**)

`delete` *n1* `[`*n2*`] ...`
`delete all`

Ukazu podajamo številke, ki jih ukaz `status` izpiše zraven posameznih nastavitev. Če želimo izbrisati vse nastavitve ukazov `trace`, `save` in `iplot`, podamo ukaz `delete all`.

Izrisovanje grafov (**plot**)

Ukaz deluje samo v grafičnem načinu. V konzolnem načinu ne moremo izrisovati grafov.

`plot [mode real|imag|mag|phase|cx|realz|imagz|realy|imagy]`
`+ [xlin] [ylin] [xlog] [ylog] [xygrid] [polar] [smith|smithz] [smithy]`
`+ [xlimit|xl` *x1 x2*`] [ylimit|yl` *y1 y2*`] [aspect]`
`+ [xdelta|xdel` *xd*`] [ydelta|ydel` *yd*`]`
`+ [xlabel` *xime*`] [ylabel` *yime*`] [title` *naslov*`]`
`+` *izraz1a izraz1b* `...` `[vs` *skupina1skala*`]`
`+` *izraz2a izraz2b* `...` `[vs` *skupina2skala*`]`
`+ ...`

Izrišejo se vrednosti izrazov *izraz1a*, *izraz2a*, ..., *izraz1b*, *izraz2b*, ... Za vsako skupino izrazov lahko podamo vrednoti, ki se nanašajo na x-os (skalo) s pomočjo besede `vs`.

Z opcijo `mode` izbiramo način interpretacije kompleksnih vektorjev. Privzeti način je `real`. V tem načinu se izrisuje realni del vektorjev. Načini `imag`, `mag` in `phase` izrisujejo imaginarni del, absolutno vrednost in fazo kompleksnega vektorja. Način `cx` izrisuje realni del na os x, imaginarni del pa na os y. V tem načinu se izbrana skala za os x (opcija `vs`) ne upošteva. Načina `realz` in `imagz` izrisujeta realni oziroma imaginarni del normirane impedance, pri čemer se vektor interpretira kot odbojnost. Načina `realy` in `imagy` sta podobna, le da izrisujeta realni oziroma imaginarni del normirane admitance. Slika 5.1 prikazuje različne načine prikazovanja istega vektorja. Grafe dobimo z naslednjim zaporedjem ukazov.

```
* Nastavi enote na radiane
set units=radians
* Vektor parametra t (0 .. 8 pi)
let t=vector(10001)/10000*8*pi
* Izracunaj x in y
let x=cos(t)*exp(-0.2*t)
```

```
let y=sin(t)*exp(-0.2*t)
* Pripravi kompleksni vektor z=x+iy
let z=(x,y)
* Nastavi stopinje za enote faze (nacin phase)
set units=degrees
* Risi
plot mode real z vs t
plot mode imag z vs t
plot mode cx z vs t
plot mode mag z vs t
plot mode phase z vs t
plot mode realz z vs t
plot mode imagz z vs t
plot mode realy z vs t
plot mode imagy z vs t
```

Opcije `xlin`, `ylin`, `xlog` in `ylog` vklopijo linearno oziroma logaritemsko skalo za osi x in y ter prikažejo pravokotno mrežo (slika 5.2). Za vklop pravokotnme mreže uporabimo opcijo `xygrid`. V tem primeru odloča o logaritemski skali privzeti vektor. Opcije `polar`, `smith` (`smithz`) in `smithy` vklopijo polarno mrežo, Smithovo impedančno mrežo oziroma Smithovo admitančno mrežo. Slika 5.3 ilustrira mreže, ki jih ponuja SPICE. Grafe na sliki dobimo z naslednjim zaporedjem ukazov.

```
* Nastavi enote na radiane
set units=radians
* Vektor parametra t (0 .. 8 pi)
let t=vector(10001)/10000*8*pi
* Izracunaj x in y
let x=cos(t)*exp(-0.2*t)
let y=sin(t)*exp(-0.2*t)
* Pripravi kompleksni vektor z=x+iy
let z=(x,y)
* Narisi na treh mrezah
plot mode cx xygrid aspect z vs t
plot mode cx polar z vs t
plot mode cx smithz z vs t
```

Meje območja, ki ga bo prikazovalo okno, nastavimo z opcijama `xlimit` (`xl`) in `ylimit` (`yl`). Opcija `aspect` ponastavi območje tako, da je prikazovano v razmerju 1:1. Tako so krogi na zaslonu res krogi.

Opciji `xdelta` (`xdel`) in `ydelta` (`ydel`) nastavita korak v katerem sta označeni osi x in y.

Za označevanje osi in grafa uporabimo opcije `xlabel`, `ylabel` in `title`. Oznake osi, ki vsebujejo presledke, damo v narekovaje, da si SPICE posameznih besed ne razlaga kot opcije ukaza `plot`.

Spremenljivki `plotwinwidth` in `plotwinheight` določata širino in višino grafičnega okna. Spremenljivka `plotwininfo` določa ali bo merilno okno za odčitavanje koordinat vklopljeno. Samodejno identifikacijo krivulje, ki je najbližja kurzorju, vklopimo s spremenljivko `plotautoident`.

Način prikazovana krivulj določa `plottype` (`point` za točkoven prikaz, `line` za črte, ki povezujejo točke in `comb` za navpične črte od osi x do vsake točke). Obliko točk v načinu `point` določa spremenljivka `pointtype` (možne so vrednosti `o`, `x`, `+`, `q`, `d`, `t`). Debelino črte na grafu določa `linewidth`. Velikost točk v načinu `point` določa spremenljivka `pointsize`.

Barvo ozadja in mreže določata spremenljivki `color0` in `color1`. Spremenljivke `color2`, `color3`, ... določajo barve krivulj. Barvo opišemo s sintakso r*cr*g*cg*b*cb*. Pri tem *cr*, *cg* in *cb* predstavljajo vrednosti rdeče,

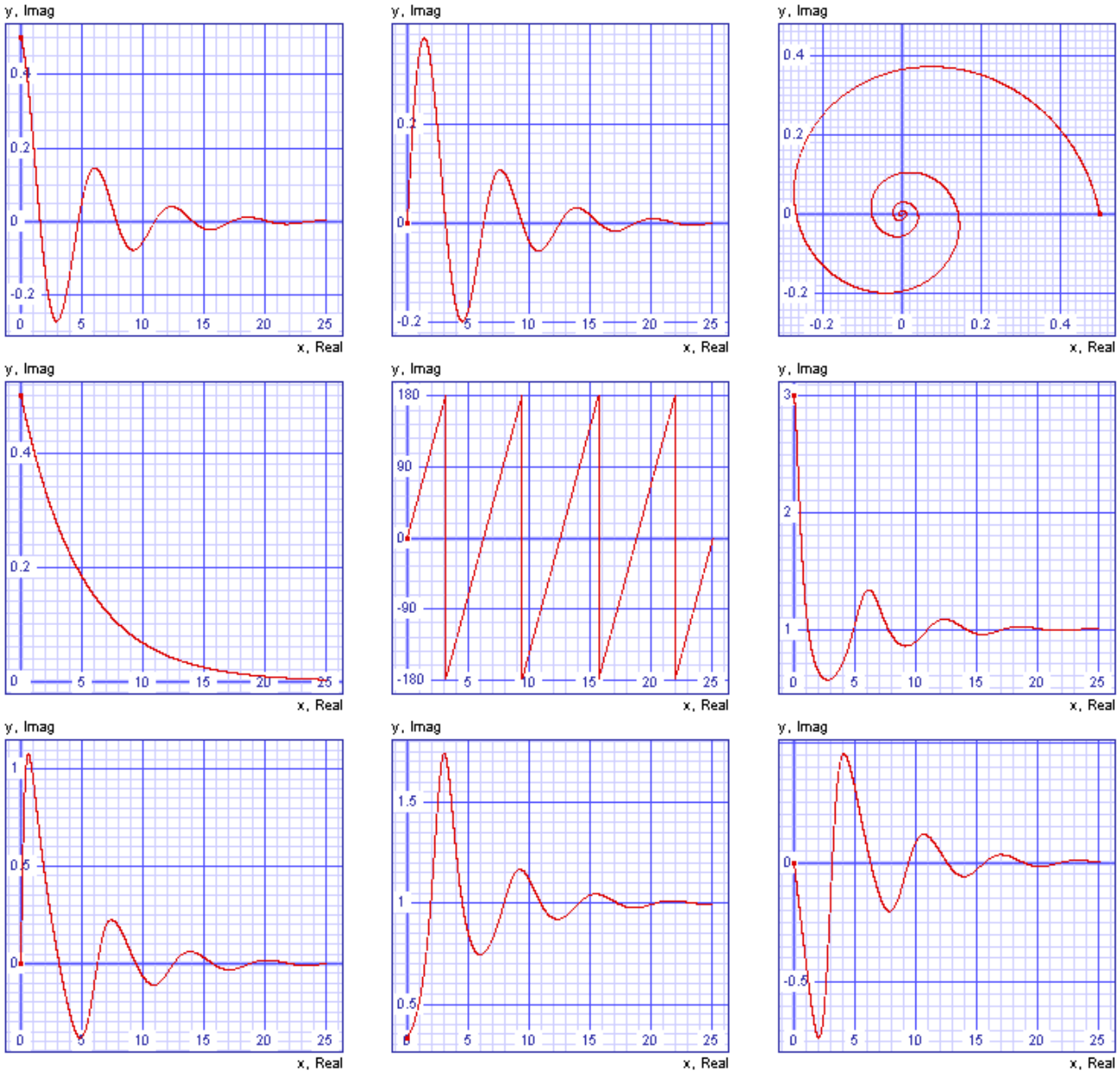
zelene in modre barvne komponente. Njihova vrednost je lahko med 0 in 255. S spremenljivko `nogrid` izklopimo prikazovanje mreže.

Enote v načinu `phase` nastavimo s spremenljivko `units` (`degrees` za stopinje in `radians` za radiane). V načinu `mag` izbiramo način prikazovanja absolutne vrednosti s spremenljivko `magunits` (`absolute` za absolutno vrednost, `db20` za decibele ojačenja napetosti ali toka in `db10` za decibele ojačenja moči).

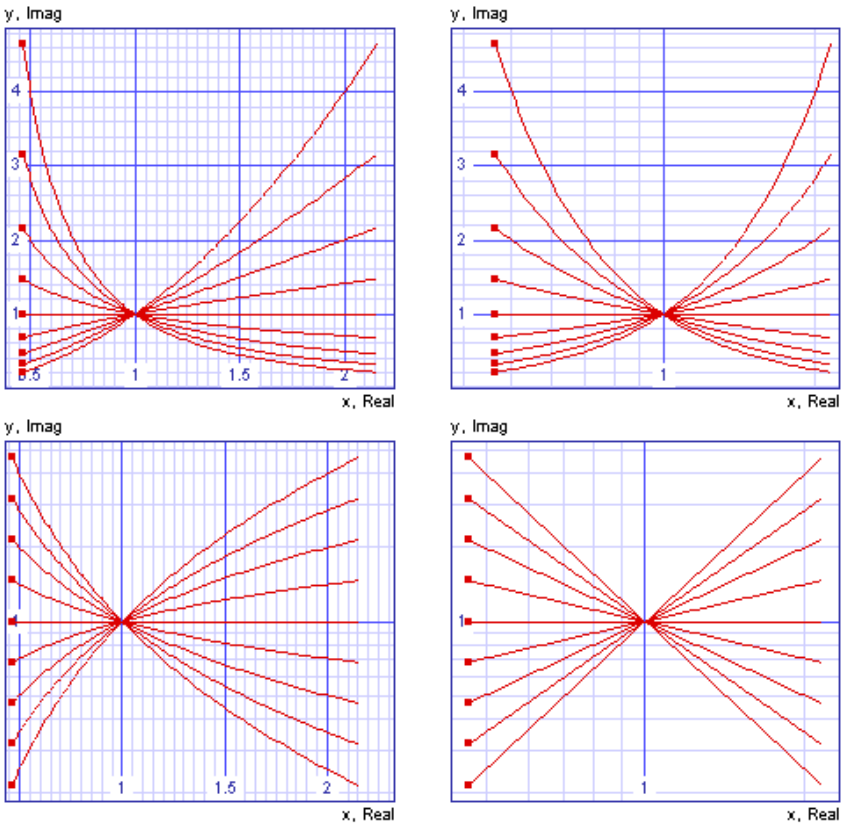
Primer:

```
* Okno 400x400, vklopi merilno okno, rdeca mreza
set plotwinwidth=400
set plotwinheight=400
set plotwininfo
set color1=r200g120b120
* Narisi
plot v(1) v(2) vs v(4) xl -5 5
plot mode cx v(out)/v(in) polar title 'Nyquistov diagram'
plot unwrap(phase(v(out)/v(in))) title Faza
```

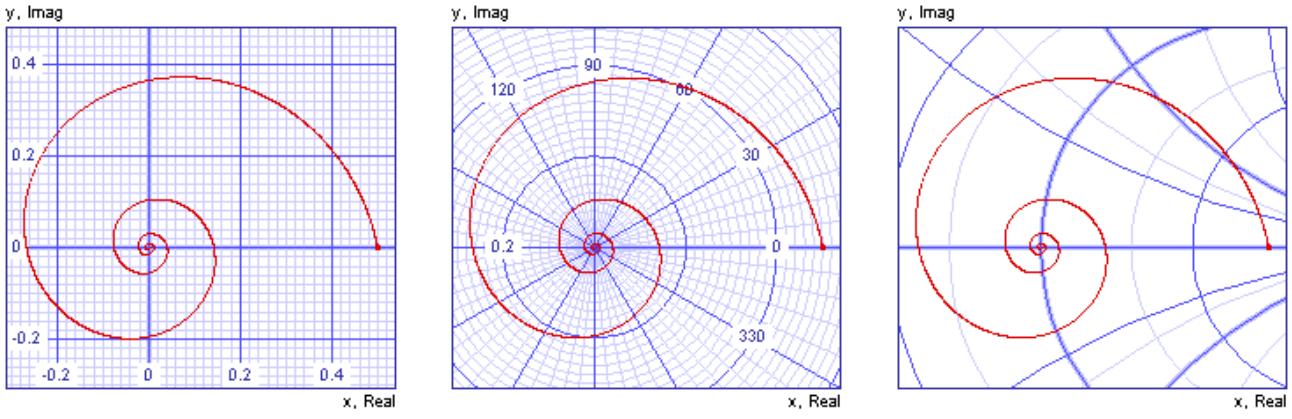
Grafično okno ima svoj meni in omogoča meritve s pomočjo kurzorja. Podrobnosti o meniju in meritvah so bile opisane v poglavjih 2.3 in 2.4.



Slika 5.1: Kompleksna krivulja $0.5\cos(t)e^{-0.2t} + 0.5i\sin(t)e^{-0.2t}$ v načinih prikazovanja `real`, `imag`, `cx`, `mag`, `ph`, `r`, `x`, `g` in `b` (od leve proti desni od vrha navzdol).



Slika 5.2: Funkcija x^n v različnih kombinacijah linearne in logaritemske skale.



Slika 5.3: Kompleksna krivulja $0.5\cos(t)e^{-0.2t} + 0.5i\sin(t)e^{-0.2t}$ na pravokotni (levo), polarni (sredina) in Smithovi (desno) mreži.

Interpolacija na skalo z enakomernim razmikom točk (**linearize**)

Predela skupino vektorjev tako, da imajo točke enake razmike na skali.
`linearize` *korak* `[`*ime1 ime2* `...]`

Za časovni korak lineariziranih vektorjev se vzame vrednost parametra *korak*. Rezultat je nova skupina vektorjev, ki postane trenutna skupina. V novi skupini imamo vektor skale in vektorje, katerih imena smo našteli kor argument ukaza. Če ne naštejemo vektorjev, se linearizirajo vsi vektorji iz trenutne skupine.

Primer:

```
* Analiza
tran 0.1m 10m
* Lineariziraj vse vektorje (korak 0.1m)
linearize 0.1m
```

Fourierova analiza vektorjev (**fourier**)

`fourier` *f1* `[`*ime1 ime2* `...]`

Funkcija samodejno interpolira zadnjih $1/f1$ skale podanih vektorjev na linearno skalo, ki je enakomerno razdeljena na `fourgridsize` točk. Stopnjo interpolacijskega polinoma določimo s spremenljivko `polydegree`. Funkcija uporablja nasledjo obliko Fourierove vrste.

$$y(t) = \sum_{n=0}^{nfreqs-1} A_n \cos(2\pi f_1 nt + \varphi_n) + \epsilon(t)$$

Analiza izpiše amplitude (A_n), faze (φ_n), normirane amplitude (faktor normiranja je A_1) in normirane faze (faza harmonske komponente zmanjšana za φ_1) posameznih harmonskih komponent. Iz amplitud se izračuna popačenje (THD) po formuli

$$THD = \frac{1}{A_1}(\sum_{n=2}^{nfreqs-1} A_n^2)^{1/2}$$

Število frekvenc, ki se upoštevajo pri izračunu popačenja (THD), določa spremenljivka `nfreqs`.

Primer:

```
* Analizirala se bo zadnja perioda (od 10m do 20m)
tran 0.1m 20m
* Osnovna frekvenca 100Hz
* Harmonska analiza napetosti vozlisca 1 in toka skozi vin
fourier 100 v(1) i(vin)
```

Literatura

[1] L. W. Nagel: *A Computer Program to Simulate Semiconductor Circuits*, Memorandum no. ERL-M5520, University of California, Berkeley, 1975.

[2] A. R. Newton, D. O. Pederson, A Sangiovanni-Vincentelli: *SPICE3 Version 3f3 User's Manual*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1993.

[3] *IsSpice4 User's Guide*, Intusoft, San Pedro, California, 1994.

[4] *PSpice User's Guide*, Cadence Design Systems, San Jose, California, 2003.

[5] *HSPICE Simulation and Analysis User's Guide*, Synopsys, Mountain View, California, 2003.

[6] *SPECTRE Circuit Simulator User's Guide*, Cadence Design Systems, San Jose, California, 2002.

[7] H. Shichman and D. Hodges: *Modeling and simulation of insulated- gate field-effect transistor switching circuits*, IEEE J. Solid-State Circuits, vol. SC-3, pp. 285-289, 1968.

[8] A. E. Parker, D. J. Skellern: *A Realistic Large-signal MESFET Model for SPICE*, IEEE Transactions on Microwave Theory and Techniques, vol. 45, no. 9, pp 1563-1571, 1997.

[9] H. Stanz et. al.: *GaAs FET Device and Circuit Simulation in SPICE*, IEEE Transactions on Electron Devices, vo. 34, no. 2, pp 160-169, 1987.

[10] J. S. Roychowdhury, D. O. Pederson: *Efficient Transient Simulation of Lossy Interconnect*, Proc. 28th ACM/IEEE Design Automation Conference, San Francisco, 1991.

[11] G. Massobrio, P. Antognetti: *Semiconductor Device Modeling with SPICE*, McGraw-Hill, 1998.

[12] D. P. Foty: *MOSFET Modeling with SPICE - Principles and Practice*, Prentice Hall, 1997.

[13] C. C. Enz, E. A. Vittoz: *Charge-Based MOS Transistor Modeling: The EKV Model for Low-Power and RF IC Design*, Wiley, 2006.

[14] *BSIM Homepage - Official Site*, `http://www-device.eecs.berkeley.edu/~bsim3`, oktober 2006.

[15] *Spice OPUS homepage*, `http://www.fe.uni-lj.si/spice`, oktober 2006.

Stvarno kazalo