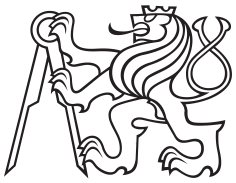
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra radioelektroniky

Program pro symbolickou analýzu lineárních elektrických obvodů

Matyáš Vašek

Školitel: doc. Dr. Ing. Jiří Hospodka
Obor: Elektronika a komunikace
Prosinec 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vašek** Jméno: **Matyáš** Osobní číslo: **491894**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra radioelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Program pro symbolickou analýzu lineárních elektrických obvodů

Název bakalářské práce anglicky:

Software for the Symbolic Analysis of Linear Electrical Circuits

Pokyny pro vypracování:

Proveďte rešerši vlastností symbolických simulátorů elektrických a elektronických obvodů a zaměřte se na možnosti analýz. Navrhněte program pro základní symbolické analýzy (DC, AC, tran) lineárních, spojitě pracujících elektrických obvodů. Program koncipujte pro využití v aplikaci GEEC [1,3] a realizujte ho s využitím volně dostupných SW prostředků. Postup konzultujte s vedoucím práce.

Seznam doporučené literatury:

[1] Paulů F.: Grafický editor elektrotechnických obvodů, diplomová práce, FEL, ČVUT v Praze 2015.
[2] Hospodka J., Bičák J.: PraCAn - Maple Package for Symbolic Circuit Analysis, Digital Technologies 2008. Žilina: Žilinská universita, Elektrotechnická fakulta, 2008. ISBN 978-80-8070-953-2.
[3] Bukovský O.: Analýzy v grafickém editoru elektrotechnických obvodů, diplomová práce, FEL, ČVUT v Praze 2021.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Dr. Ing. Jiří Hospodka katedra teorie obvodů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **17.08.2023** Termín odevzdání bakalářské práce: **09.01.2024**

Platnost zadání bakalářské práce: **16.02.2025**

doc. Dr. Ing. Jiří Hospodka	doc. Ing. Stanislav Vítek, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

V první řadě bych chtěl poděkovat vedoucímu této práce, doc. Dr. Ing. Jiřímu Hospodkovi. Především za časté a kvalitní konzultace, vstřícnost, podporu a hlavně za to, že ve mě vzbudil hlubší zájem o problematiku návrhu obvodů. Dále děkuji Ing. Ondřeji Bukovskému, který mi pomohl vstoupit do projektu a zařídil napojení mého balíčku na simulátor GEEC.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 7. prosince 2023

Abstrakt

První část této práce shrnuje dostupné CAS (Computer Algebra System) využitelné pro symbolickou analýzu lineárních obvodů a představuje přehled teorie, vhodných metod a nástrojů pro realizaci symbolického simulátoru obvodů. V praktické části jsou popsány postupy, kterými jsem naprogramoval SymCircPy, Pythonový balíček pro symbolickou analýzu lineárních obvodů. Práce porovnává dvě různé maticové metody analýzy obvodů - dvojgrafovou modifikovanou metodu uzlových napětí (DMMUN) a tableau metodu. Pro malé obvody (menší než 8 prvků) se ukázala být výhodnější tableau metoda. Pro větší obvody je výhodnější metoda DMMUN. Praktickou limitací čistě symbolické simulace se ukázaly být obvody s 11 až 12 proměnnými. Pro semisymbolickou simulaci s jednou proměnnou je na základě simulací výhodnější metoda DMMUN. Na konci práce je uvedeno statictické srovnání DC analýzy SymCircPy a balíčku Lcapy. SymCircPy dosahuje pomocí metody DMMUN lepšího simulačního času než Lcapy.

Klíčová slova: GEEC, python, symbolická analýza obvodů, lineární obvody, sympy, semisymbolická analýza obvodů

Školitel: doc. Dr. Ing. Jiří Hospodka
Katedra teorie obvodů,
Technická 2,
Praha 6

Abstract

The first part of this thesis summarizes available CAS (Computer Algebra System) usable for symbolic analysis of linear circuits and provides an overview of theory, suitable methods, and tools for implementing a symbolic circuit simulator. In the practical section, I describe the procedures through which I programmed SymCircPy, a Python package for symbolic analysis of linear circuits. This thesis compares two different matrix methods for circuit analysis: the two-graph modified nodal method and the tableau method. For small circuits (less than 8 elements), the tableau method proved to be advantageous. For larger circuits, the DMMUN method is more favorable. The practical limitation of purely symbolic simulation was observed to be circuits with 11 to 12 variables. For semi-symbolic simulation with one variable, the two-graph modified nodal method is preferable based on simulations. Towards the end of the work, I compared the DC analysis performance of SymCircPy with the Lcapy package. Using the two-graph modified nodal method, SymCircPy achieves a better simulation time than Lcapy.

Keywords: GEEC, python, symbolic circuit analysis, linear circuits, sympy, semisymbolic circuit analysis

Title translation: Software for the Symbolic Analysis of Linear Electrical Circuits

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V teoretické části této práce se zabývám rešerší možností symbolické simulace elektrických obvodů pomocí volně dostupných a bezplatných nástrojů. Pokračuji rešerší metod a postupů nezbytných k návrhu a realizaci softwarového balíčku pro symbolickou analýzu elektrických obvodů. Výsledkem praktické části práce je balíček pro symbolickou analýzu obvodů napsaný v programovacím jazyce Python. Tento balíček jsem vyvinul pro grafický editor a simulátor obvodů GEEC. GEEC je veřejně dostupná online aplikace, která bez nutnosti instalace disponuje intuitivním a jednoduchým ovládáním. Nabízí uživateli mnoho nástrojů z oblasti numerické analýzy obvodů, ale také nástroje symbolické analýzy momentálně poskytované balíčkem PraCAn. Další velkou výhodou GEECu je možnost exportovat obvod do formátu pdf, této funkce moje práce využívá a všechna schémata obvodů v této práci jsou exportovaná z GEECu. Hlavní motivací pro náhradu PraCanu je licence, pod kterou byl PraCAn vytvořen. PraCAn byl napsaný v jazyce Maple, který má restriktivní licenci. SymCircPy proto používá výhradně knihovny s licencí, která umožňuje open-source vývoj.

Část I

Teoretická část

Kapitola 2

Rešerše dostupných simulátorů elektrických obvodů

Nástroje určené k simulaci elektronických obvodů jsou dnes naprosto nezbytnou součástí výbavy každého návrháře obvodů. Většina dobře známých simulátorů se specializuje na numerické řešení. Většinou se jedná o SPICE-like simulátory. Numerická simulace je mnohem méně výpočetně náročná než symbolická. Z tohoto důvodu dnes používá většina simulátorů převážně numerické metody. Pomocí numerické simulace není snadné získat obecný vztah. Z výsledků lze různými metodami odhadnout obecný tvar funkce, ale ani tento přístup neposkytuje takový vhled do chování obvodu jako symbolická simulace [1]. Symbolické simulátory naopak poskytují jako výsledek obecné vztahy. To je při návrhu obvodů velmi užitečné. Jsou také vhodné jako didaktická pomůcka při vyučování. Umožní studentům lépe pochopit chování obvodů a ověřit si ručně vypočítané výsledky. Jejich nevýhodou je, že jsou pomalejší než numerické simulátory.

Protože symbolická simulace obvodů je založená na symbolických výpočtech, bylo prvním krokem rešerše vybrat vhodný CAS (computer algebra system), ve kterém bych mohl symbolický simulátor implementovat. CAS (někdy také SAS, symbolic algebra system) je matematický systém určený k symbolickým výpočtům. Zvolil jsem tento postup rešerše, protože symbolická simulace obvodů se dá v principu dělat přímo v některých CAS. Symbolické simulátory obvodů jsou většinou nástavbou takového systému. Nejprve jsem vyhledal aktuální seznam takových systémů [2] a z něj jsem vybíral podle těchto kritérií:

- Musí být bezplatný a open-source.
- Měl by být aktivně udržovaný/vyvíjený.
- Umí řešit soustavy rovnic.
- Umí počítat integrální transformace (kvůli tranzientní analýze).

Výsledný výběr jsem shrnul v tabulce 2.1. Mathics [3], SageMath [4] i Cadabra [5] jsou závislé na balíčku SymPy [6], který sám o sobě splňuje všechny požadavky pro tuto aplikaci. Mezi FriCAS [7], Maxima [8] a SymPy jsem zvolil Sympy, protože považuji programovací jazyk Python [9] za perspektivnější a lépe přístupný než Common Lisp [10].

CAS	Implementovaný v jazyce	Uživatelský programovací jazyk
FriCAS	Common Lisp	SPAD
SymPy	Python	Python
Mathics	Python	Python
SageMath	Python	Python
Cadabra	Python	Python
Maxima	Common Lisp	Maxima/Lisp

Tabulka 2.1: Přehled vhodných CAS

2.1 Vybraný CAS

Pro realizaci simulátoru jsem zvolil SymPy. Je to knihovna pro symbolickou matematiku napsaná v Pythonu. Jejím cílem je stát se plně funkčním počítačovým algebraickým systémem (CAS) a přesto si udržet co nejjednodušší zdrojový kód, aby byla pochopitelná a snadno rozšiřitelná. SymPy je napsaná plně v Pythonu. Knihovna SymPy splňuje všechny požadavky této práce. Je bezplatná a open-source, má velmi rozsáhlou a kvalitní dokumentaci a obsahuje všechny základní funkce nezbytné pro symbolické výpočty, které potřebuje symbolický simulátor obvodů. Na základě balíčku SymPy již existuje několik symbolických simulátorů, nejrozšířenější jsou:

- Lcapy [11] – čistě symbolický simulátor. Je v aktivním vývoji a měl by mít do budoucna vlastní uživatelské rozhraní. Používá k simulaci modifikaci metody uzlových napětí.
- Ahkab [12] – primárně zaměřený na numerickou simulaci. Nabízí i možnost symbolické AC a DC analýzy. Také používá k simulaci modifikaci metody uzlových napětí.

V dalším kroku rešerše jsem proto prozkoumal možnosti uvedených simulátorů. Protože PraCan [13] je původní knihovna využívaná simulátorem GEEC, použil jsem její výsledky jako benchmark.

Simulátor	DC a AC	Tran	Vyvíjený	Podobně rychlý jako PraCAn
Lcapy	Ano	Ano	Ano	Ne
Ahkab	Ano	Ne	Ne	Ne

Tabulka 2.2: Porovnání simulátorů založených na SymPy

Ani jeden ze simulátorů nedosahuje rychlosti PraCAnu (viz tabulku 2.2) a u obou by bylo nutné provést velké množství úprav pro kompatibilitu s GEECem. Po domluvě s vedoucím práce jsem se proto rozhodl implementovat vlastní symbolický simulátor na základě knihovny SymPy. Protože PraCAn

rychlostí simulace předčí oba zmíněné simulátory, rozhodl jsem se inspirovat jeho metodikou řešení. Většina metod uvedených v této práci vychází z vynikající publikace Computer Methods For Circuit Analysis [14]. Z té čerpal i autor simulátoru PraCAn.

2.2 GEEC

Grafický editor eletronických obvodů (GEEC) [15] je webový simulátor, který poskytuje nástroje numerické i symbolické analýzy. Má webové grafické rozhraní, ve kterém může uživatel pohodlně sestavovat obvody a simulovat jejich chování. Cílem mojí práce je vytvořit nástroj, který by GEEC mohl v budoucnu používat k symbolické simulaci. V tuto chvíli používá GEEC k symbolické simulaci balíček PraCAn.

2.3 PraCAn

PraCAn [13] byl naprogramován v jazyce Maple [16] a podléhá tudíž jeho licenci. Uživatelé simulátoru GEEC, kteří nemají licenci Maple, nemohou tento balíček využít. Je to velmi rychlý a efektivní simulátor. Dlouhodobým cílem mého simulátoru je dosáhnout dostatečné kvality, aby mohl PraCAn nahradit. PraCAn mi umožnil proniknout do problematiky obvodové simulace a silně inspiroval metodiku řešení.

Kapitola 3

Základy teorie potřebné k algoritmizaci simulace obvodů

Tato práce se zabývá analýzou pomocí teorie obvodů. Zákony teorie obvodů vycházejí ze zjednodušení maxwellových rovnic. V celé teoretické části budu využívat převážně Kirchhoffovy zákony a Ohmův zákon. Abych mohl považovat Kirchhoffovy zákony ve tvaru

$$\sum_{k=1}^{n} I_k = 0 \tag{3.1}$$

$$\sum_{k=1}^{n} U_k = 0 \tag{3.2}$$

a Ohmův zákon ve tvaru

$$V = IR \tag{3.3}$$

za platné, musím zavést určité předpoklady a omezení. Uvedená omezení budou platit pro výsledný simulátor. Hlavními z nich jsou následující (viz [17]):

1. Prvky a obvody, které analyzuji, musí být rozměrově zanedbatelné v porovnání s nejkratší vlnovou délkou signálu. Čím menší je rozdíl rozměrů, tím méně bude simulace přesná. Toto omezení vychází ze zanedbání fyzických rozměrů prvků, simulují se pouze jejich idealizované varianty.

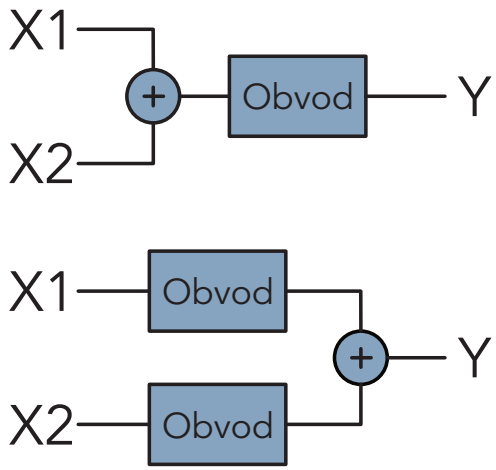
2. Předpokládám, že přenos napětí a proudu mezi zdroji a zbytkem obvodu je okamžitý.

3. Zanedbávám ztráty energie způsobené radiací, tyto ztráty začínají být významné až u vysokých frekvencí. Tento faktor je závislý na materiálu součástky.

Hlavním předpokladem jsou tedy omezené maximální frekvence vstupního signálu. Řádově jde o limitaci na úrovni stovek MHz až jednotek GHz. Tato limitace je přímo závislá na rozměru obvodu. Konkrétně se dále zabývám jen analýzou LTI (linear time-invariant) systémů. Jsou to takové systémy, které splňují dva základní předpoklady.

1. Systém je lineární. Tato podmínka je splněna, pokud systém splňuje princip superpozice. Princip superpozice je ilustrovaný na obrázku 3.1.

2. Systém je časově invariantní (nezávislý). Je-li systém vybuzen z klidového stavu stejnou funkcí v různých časech, budou výsledky identické.



Obrázek 3.1: Ilustrační diagram linearity systému

3.1 Ideální obvodové prvky

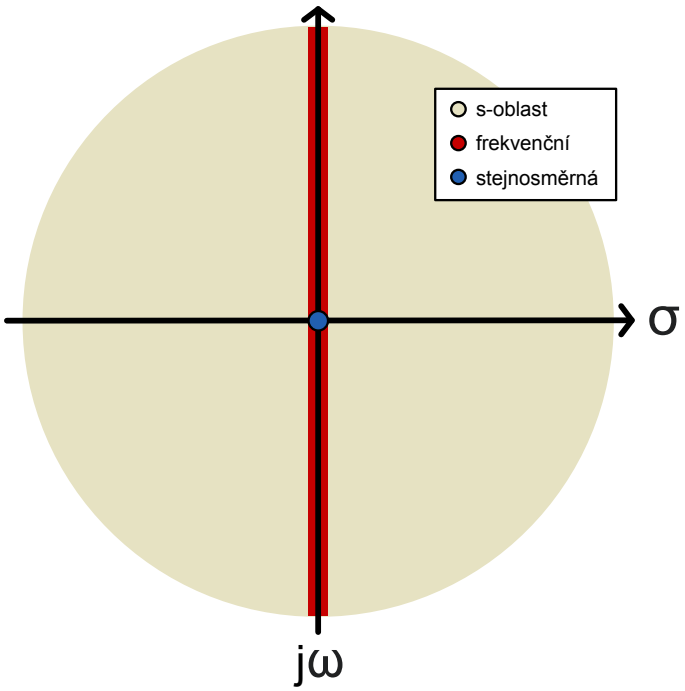
V této kapitole popíšu elementární obvodové prvky, se kterými bude můj simulátor pracovat. Z prvků uvedených v této kapitole je možné dále stavět složitější modely, které již blíže odpovídají reálným součástkám.

Z předpokladů uvedených v předchozí sekci vyplývá, že všechny obvodové prvky musí být lineární. Zároveň se také jedná jen o prvky se soustředěnými parametry. U prvků se soustředěnými parametry zanedbávám fyzické rozměry a předpokládám, že jsou mezi nimi spoje z ideálních vodičů.

Některé ideální prvky se v časové oblasti definují pomocí diferenciálních rovnic. Z tohoto důvodu probíhá analýza obvodů většinou v s-oblasti (v literatuře také Laplaceova doména). V s-oblasti se, zjednodušeně řečeno, diferenciální operátory mění na kladné mocniny proměnné s a integrální operátory na záporné mocniny s. Z integro-diferenciálních rovnic časové oblasti se tedy v s-oblasti stávají polynomiální rovnice. Jedná se o naprosto zásadní zjednodušení pro řešení soustav obvodových rovnic. Pro další kapitoly této práce je nezbytné stručně uvést rozdíl mezi s-oblastí a frekvenční oblastí. Podrobnější rozbor je v kapitole 3.3. Funkce obvodových veličin v s-oblasti jsou závislé na proměnné

$$s = \sigma + j\omega, \qquad \omega = 2\pi f, \tag{3.4}$$

kde ω reprezentuje úhlovou frekvenci (radial frequency), σ reprezentuje útlum funkce a v praxi určuje rychlost přechodového děje. Ve frekvenční oblasti jsou



Obrázek 3.2: Ilustrační diagram vztahu různých oblastí analýzy

funkce veličin závislé na proměnné $j\omega$. Je tedy zřejmé, že frekvenční oblast je podoblastí s-oblasti, kde $\sigma = 0$. Tento stav nastane, pokud se obvod ustálí v HUS (harmonický ustálený stav) nebo SUS (stejnosměrný ustálený stav). Dále tedy budu psát převážně o s-oblasti, protože se jedná o obecnější oblast (viz obrázek 3.2).

Pokud je požadovaným výsledkem analýza v časové oblasti, provedeme na konci simulace zpětnou integrální transformaci. Standardně se používá Laplaceova transformace (viz sekci 3.3). Obvodové rovnice budu proto stavět v s-oblasti.

Chování obvodových prvků je možné obecně definovat pomocí maticové rovnice

$$\begin{bmatrix}\mathbf{Y}_1\\ \mathbf{K}_2\end{bmatrix}\mathbf{V}_b + \begin{bmatrix}\mathbf{K}_1\\ \mathbf{Z}_2\end{bmatrix}\mathbf{I}_b = \begin{bmatrix}\mathbf{W}_{b_1}\\ \mathbf{W}_{b_2}\end{bmatrix}. \tag{3.5}$$

$\mathbf{Y}_1$ reprezentuje admitance a $\mathbf{Z}_2$ reprezentuje impedance. $\mathbf{K}_1$ je matice bezrozměrných konstant korespondujících k $\mathbf{Y}_1$. $\mathbf{K}_2$ je matice bezrozměrných konstant korespondujících k $\mathbf{Z}_2$. $\mathbf{W}_{b_1}$ reprezentuje zdrojovost prvků s levou stranou $\mathbf{Y}_1\mathbf{V}_b + \mathbf{K}_1\mathbf{I}_b$, tento parametr je nenulový jen pro zdroje a prvky s nenulovými počátečními podmínkami. Podobně je $\mathbf{W}_{b_2}$ zdrojovost prvků s levou stranou $\mathbf{K}_2\mathbf{V}_b + \mathbf{Z}_2\mathbf{I}_b$. Maticovou rovnici 3.5 je možné zjednodušit pro jednoportový prvek do tvaru

$$Y_bV_b + Z_bI_b = W_b. \tag{3.6}$$

V tabulce 3.1 jsou uvedené příklady odvozených rovnic základních prvků.

Prvek	Stavební rovnice	Y_b	Z_b	W_b
Rezistor	$V_b - R_b I_b = 0$	1	$-R_b$	0
Kapacitor	$sC_bV_b - I_b = C_bV_0$	sC_b	-1	C_bV_0
Induktor	$V_b - sL_bI_b = -L_bI_0$	1	$-sL_b$	$-L_bI_0$
Zdroj napětí	$V_b = U_b$	1	0	U_b
Zdroj proudu	$I_b = J_b$	0	1	J_b

Tabulka 3.1: Příklady stavebních rovnic prvků (převzato z publikace [14])

Pro kapacitor a induktor platí uvedené vztahy pouze v případě, že je cílem tranzientní analýza. Pokud je cílem TF, AC nebo DC analýza, je třeba dosadit

$$V_0 = 0, \qquad I_0 = 0. \tag{3.7}$$

Rovnice 3.5 vychází z Ohmova zákona ve tvaru

$$U(s) = I(s)Z(s). \tag{3.8}$$

Z je obecně komplexní číslo reprezentující celkovou impedanci prvku. U je napětí na prvku a I je proud protékající skrz prvek. Impedance je definována jako

$$Z = R + jX, \qquad Y(s) \equiv \frac{1}{Z(s)}. \tag{3.9}$$

R je rezistence, X je reaktance a Y je admitance. Z je komplexní veličina. Impedance umožňuje modelovat schopnost materiálu bránit průtoku proudu.

3.1.1 Nezávislé zdroje

Nezávislý ideální zdroj napětí má mezi uzly vždy pevné napětí. Dokáže do obvodu dodat takové množství proudu, které si obvod vyžádá. Je definován jako

$$U = V(s). \tag{3.10}$$

Zdroje mohou mít různé signály. Uvedu seznam signálů relevantních pro tuto práci. Každému signálu přiřadím název, kterým je tento signál definován v netlistu (viz sekci 4).

1. DC – stejnosměrné napětí, $V(t) = V_0$.
2. AC – střídavé napětí, $V(t) = V_0 sin(\omega t + \phi)$.
3. SIN – tlumený sinusoidální průběh, $V(t) = V_0 e^{-t\lambda} sin(\omega t + \phi)$. Určený pro tranzientní analýzu.
4. Ostatní periodické průběhy pro tranzientní analýzu. Například čtvercový signál, trojúhelníkový signál, pilový signál. Tyto druhy signálů zatím nejsou implementovány a jsou předmětem budoucího vývoje simulátoru.

Pokud $V(s) = 0$, tak se napěťový zdroj chová jako zkrat. Má na svorkách nulové napětí a protéká jím libovolný proud. Této vlastnosti můžeme využít a zkraty v obvodu modelovat jako nulové napěťové zdroje. Zkraty jsou jako obvodový prvek výhodné například pro modelování ideálních spínačů. Obráceně má proudový zdroj pevně daný protékající proud a neomezené napětí na svorkách.

$$I = J_s \tag{3.11}$$

3.1.2 Rezistor, induktor a kapacitor

Rezistor, induktor a kapacitor jsou základní obvodové prvky modelující tři různé druhy impedance, se kterými se v obvodu můžeme setkat. Jedná se o jednoportové prvky. Induktor a kapacitor jsou v časové oblasti definovány pomocí integro-diferenciálních rovnic. Induktor jako

$$u_L(t) = L\frac{\mathrm{d}i_L(t)}{\mathrm{d}t}, \qquad i_L(t) = \frac{1}{L}\int_0^t u_L(\tau)\mathrm{d}\tau + i_L(0) \tag{3.12}$$

a kapacitor jako

$$i_C(t) = C\frac{\mathrm{d}u_C(t)}{\mathrm{d}t}, \qquad u_C(t) = \frac{1}{C}\int_0^t i_C(\tau)\mathrm{d}\tau + u_C(0). \tag{3.13}$$

Rezistor je definován jako

$$u_R(t) = Ri_R(t). \tag{3.14}$$

Jejich zápis v s-oblasti můžeme definovat snadno pomocí ohmova zákona v komplexním tvaru (viz rovnici 3.8) a Laplaceovy transformace jako

$$U_R(s) = RI_R(s), \tag{3.15}$$

$$U_L(s) = sLI_L(s), \tag{3.16}$$

$$U_C(s) = \frac{I_C(s)}{sC}. \tag{3.17}$$

Příslušné impedance můžeme tedy zapsat jako

$$Z_R = R, \tag{3.18}$$

$$Z_L = sL, \tag{3.19}$$

$$Z_C = \frac{1}{sC}. \tag{3.20}$$

Induktor a kapacitor jsou frekvenčně závislé. Rezistor je frekvenčně nezávislý. Kapacitor a induktor mohou mít při tranzientní analýze na počátku nenulovou hodnotu setrvačné veličiny. V takovém případě má prvek nenulovou počáteční podmínku. Je důležité podotknout, že vliv počátečních podmínek se v simulaci používá jen pro tranzientní analýzu.

Inverzí rezistoru je konduktor, ten je definován svou admitancí. Admitance Y prvku definuje jeho propustnost (viz druhý vztah 3.9). Protože rezistor a konduktor jsou zaměnitelné prvky, zavedu do simulátoru jen jeden z nich a to rezistor.

3.1.3 OAMP, nulátor a norátor

Abych mohl dobře definovat ideální operační zesilovač (IOAMP), je dobré nejprve zmínit nulátor a norátor. Jedá se o čistě teoretické prvky.

- Nulátorem neprotéká žádný proud a zároveň má nulové napětí mezi uzly.
- Norátorem naopak protéká libovolný proud a má libovolné napětí na svorkách.

Vstup IOAMPu můžeme definovat jako nulátor a výstup jako norátor. Pomocí nulátoru a norátoru lze modelovat i ideální tranzistor.

3.1.4 Řízené zdroje

Dalšími významnými dvouportovými prvky jsou řízené zdroje. Rozlišují se čtyři druhy:

- VCVS – napětím řízený zdroj napětí,
- VCCS – napětím řízený zdroj proudu,
- CCVS – proudem řízený zdroj napětí,
- CCCS – proudem řízený zdroj proudu.

Řízené zdroje snímají napětí nebo proud mezi vstupními uzly a na základě jeho velikosti mění velikost napětí nebo proudu řízeného zdroje. Pomocí řízených zdrojů lze dále vytvářet linearizované modely tranzistorů.

3.2 Typy analýzy obvodů

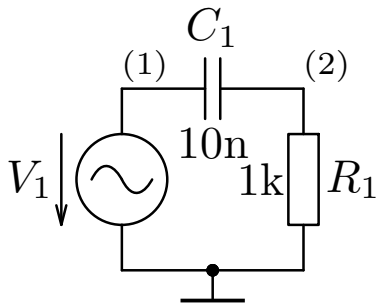
V této práci se budu zabývat stejnosměrnou (DC), střídavou (AC), tranzientní (tran), přenosovou (TF) a pole-zero (PZ) analýzou. Kromě těch existuje i mnoho dalších, například citlivostní analýza, spektrální analýza, noise analýza, teplotní analýza a další. Těmi se ale tato práce zabývat nebude.

3.2.1 TF, PZ a Tran

TF analýza je ze zmiňovaných analýz nejobecnější. Probíhá v s-oblasti. Všechny ostatní lze z jejich výsledků odvodit. Výsledkem této analýzy jsou funkce závislé na proměnné s. Například přenosová funkce RC článku 3.3 má v s-oblasti tvar

$$H(s) = \frac{CRs}{CRs + 1}. \tag{3.21}$$

S touto funkcí můžeme dále pracovat a získat z ní časový průběh (tran) nebo třeba póly a nuly obvodu (PZ). Transientní analýza umožňuje zkoumat chování obvodu v časové oblasti. Simuluje situaci, kdy je obvod v přechodovém



Obrázek 3.3: Schéma RC článku (staženo z [15])

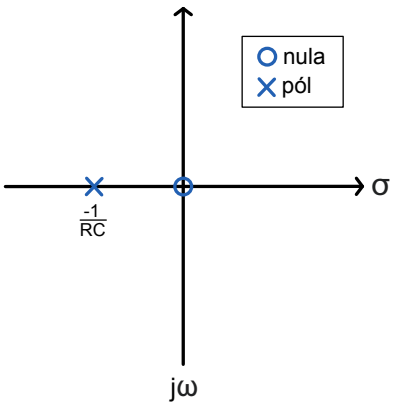
stavu. Vstupní signál může být stejnosměrný i střídavý. Obvod řešíme typicky v s-oblasti a následně pomocí zpětné Laplaceovy Transformace (viz sekci 3.3) transformujeme výsledky do časové oblasti. Pro RC článek (obrázek 3.3) je tvar časového průběhu přechodového jevu pro vybitý kapacitor a stejnosměrný zdroj napětí

$$h(t) = e^{-\frac{t}{CR}}. \tag{3.22}$$

Výsledky tranzientní analýzy je vhodné dále graficky vizualizovat. Jedná se o nezbytný nástroj pro návrh téměř každého analogového obvodu.

PZ analýza je užitečná například při návrhu filtrů a při analýze stability systémů. K získání PZ stačí ve výsledku TF v čitateli a jmenovateli přenosové funkce nalézt nulové body.

- Nulový bod čitatele se nazývá nula (zero).
- Nulový bod jmenovatele se nazývá pól (pole).



Obrázek 3.4: PZ analýza RC článku

Nuly a póly se obvykle vynášejí na komplexní rovinu do grafu (viz obrázek 3.4).

TF analýzu je možné vnímat jako obecný předstupeň všech ostatních analýz, které jsou v této práci popsané. V principu jsou AC i DC jejím speciálním případem. PZ a tranzientní analýza jsou jen její dodatečné zpracování.

3.2.2 AC a DC analýza

AC analýza simuluje situaci, ve které je systém v ustáleném stavu a zároveň je buzen sinusoidálním signálem. Tento stav se v literatuře označuje zkratkou HUS (harmonický ustálený stav). Tato analýza probíhá ve frekvenční oblasti. Typicky požadovaným výsledkem AC analýzy je přenosová funkce mezi dvěma uzly obvodu. Výsledná přenosová funkce může být frekvenčně závislá. Jak jsem již zmínil v kapitole 3.1, frekvenční oblast je podoblastí s-oblasti, kde $\sigma = 0$. Dosazením $s = j\omega$ tedy z TF analýzy přímo získáme výsledky AC analýzy. Jako příklad dosadím do rovnice 3.21:

$$H(j\omega) = \frac{CRj\omega}{CRj\omega + 1}. \tag{3.23}$$

Vlastnosti obvodu ve frekvenční oblasti je možné vizualizovat pomocí grafických metod (např. Bodeho graf).

Rozdíl mezi DC a AC analýzou v principu spočívá jen ve frekvenci vstupního signálu. Pokud je obvod buzen v obou případech signálem se stejnou amplitudou, platí mezi přenosovou funkcí AC a DC analýzy následující vztah

$$H_{DC} = \lim_{f \to 0} H_{AC}(f), \tag{3.24}$$

kde H je obecná přenosová funkce obvodu. Typicky se pro provedení DC analýzy v simulátorech tento vztah nepoužívá, má převážně teoretický význam. Z tohoto vztahu lze odvodit na základě rovnic 3.19 a 3.20, že induktor se v DC analýze bude chovat jako zkrat a kapacitor jako rozpojený obvod:

$$Z_L = \lim_{s \to 0} sL = 0, \qquad Z_C = \lim_{s \to 0} \frac{1}{sC} = \infty. \tag{3.25}$$

Pomocí limit je tedy možné řešit DC analýzu přímo z TF výsledků. Druhá možnost je sestavit soustavu obvodových matic přímo pro DC analýzu. Při překladu netlistu stačí nahradit induktory zkratem a kapacitory do soustavy rovnic vůbec nezavádět. Stavu obvodu ve kterém provádíme DC analýzu se říká SUS (stejnosměrný ustálený stav). Pro případ RC článku je výsledkem DC analýzy

$$H = 0. \tag{3.26}$$

Při AC a DC analýzách se nepoužívají počáteční podmínky kapacitoru a induktoru.

3.3 Inverzní Laplaceova transformace (iLT)

Tuto sekci jsem se rozhodl sepsat, protože jsem v průběhu práce naprogramoval vlastní implementaci iLT. V tuto chvíli používá můj simulátor iLT z balíčku SymPy, protože dosahuje lepších výsledků než moje implementace.

Laplaceova transformace (LT)

$$F(s) = \int_{0^-}^{\infty} f(t)e^{-st}dt \tag{3.27}$$

je integrální transformace, která transformuje funkci závislou na čase na její ekvivalent závislý na proměnné s. Jinými slovy převádí funkci z časové oblasti do s-oblasti.

Inverze (iLT) této transformace

$$f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s)e^{st}ds \tag{3.28}$$

umožňuje transformovat funkci z s-oblasti do časové oblasti. Pro tuto práci je významná pouze iLT, umožňuje totiž transformovat výsledky TF analýzy na časové průběhy bez nutnosti výpočtu složitých integro-diferenciálních rovnic (viz sekci 3.1). Na papíře běžně LT a ILT počítáme pomocí rozkladu funkce na parciální zlomky a tabulky elementárních vzorů. Vzhledem k tomu, že LTI systémy mají konečné množství stavebních prvků, lze předpokládat, že existuje i konečné množství vzorů těchto parciálních zlomků. K výpočtu ILT lze také využít residuovou metodu

$$f(t) = \sum_{p} Res[F(s)e^{st}], \tag{3.29}$$

kde p je seznam všech pólů (nulových bodů jmenovatele) funkce $F(s)$.

iLT jsem implementoval jako kombinaci tabulkové a residuové metody. Tabulková metoda je ideální z hlediska časové náročnosti samotné transformace. Nejprve je ale nutné funkci rozložit na parciální zlomky, což je pro symbolické funkce problematické. V tabulce se nemusí všechny možné vzory vyskytovat. Je proto vhodné pro případy, které se v tabulkách nevyskytují, řešit transformaci jinak. Pro tyto případy jsem zvolil reziduovou metodu. V mojí implementaci je nejprve vstupní funkce rozložena na parciální zlomky. Každý zlomek je následně porovnán s tabulkovými vzory. Proces porovnání spočívá ve vyhledání dostatečného množství ukazatelů, které jednoznačně dokazují, že je zlomek podobný nějakému tabulkovému vzoru. Pokud je vzor zlomku v tabulce, vygeneruje se jeho obraz v časové oblasti. Pokud v tabulce není, zpracuje se transformace pomocí residuové metody.

Kapitola 4

Definice netlistu pro parser

Výsledek mojí práce by měl být vhodný pro použití ve webovém simulátoru GEEC. Převzal jsem proto definici netlistu kterou používá PraCAn. Díky tomu by měl být můj simulátor s GEECem dobře kompatibilní. Netlist je textová reprezentace elektronického obvodu. V netlistu je každý prvek reprezentován

1. názvem prvku,
2. uzly ke kterým je připojen,
3. parametry daného prvku (rezistence, kapacita...).

Nějaká forma netlistu je používána ve většině nástrojů pro simulaci obvodů. Existuje mnoho různých formátů (např. SPICE, Verilog, VHDL, EDIF) a použitý formát závisí na hlavním účelu nástroje. Jak jsem již zmínil, rozhodl jsem se převzít formát navržený pro PraCAn, tento formát je založený na SPICE formátu [18], ale obsahuje drobné úpravy a změny. Nepodporuje zápis příkazů pro analýzu do netlistu, místo toho uživatel volí typ analýzy pomocí parametrů přímo v programu. První řádek netlistu je vždy rezervován pro název. V netlistu mohou být komentáře

```
* řádky začínající hvězdičkou jsou ignorovány
```

Pro úplnost uvedu všechny příkazy, které jsou v netlistu pro tento projekt povolené. První řádek je obecná definice a druhý je konkrétní příklad.

```
Rezistor:            RXXX N+ N- ODPOR
                     R1 1 3 1k

Kapacitor:           CXXX N+ N- KAPACITA <IC=POČÁTEČNÍ PODMÍNKA>
                     C12 0 1 10n

Induktor:            LXXX N+ N- INDUKČNOST <IC=POČÁTEČNÍ PODMÍNKA>
                     La 2 11 3p IC=2

Induktorová vazba:   KXXX LYYY LZZZ K
                     K1 L1 L2 0.5
```

```
Zdroj napětí:       VXXX N+ N- dc NAPĚTÍ <ac AMPLITUDA <FÁZE>>
                    Vs a~b dc 10 ac 10 0

Zdroj proudu:       IXXX N+ N- dc PROUD <ac AMPLITUDA <FÁZE>>
                    Is a~b dc 100m ac 20m 45

IOamp:              AXXX Nout1 Nout2 Nin+ Nin-
                    A1 1 0 2 3

VCVS:               EXXX N1 N2 Ncontrol+ Ncontrol- ZESÍLENÍ
                    E1 1 2 3 4 e

CCCS:               FXXX N1 N2 VSENSE ZESÍLENÍ
                    F1 1 2 3 4 f

VCCS:               GXXX N1 N2 Ncontrol+ Ncontrol- ZESÍLENÍ
                    G1 1 2 3 4 g

CCVS:               HXXX N1 N2 VSENSE ZESÍLENÍ
                    E1 1 2 3 4 h
```

Parametry v ostrých závorkách jsou volitelné. Uzly mohou být číslo nebo znak. Název prvku je vždy velké písmeno identifikující typ prvku následované libovolnou sekvencí písmen a čísel. Hodnoty prvků lze zadat v těchto tvarech:

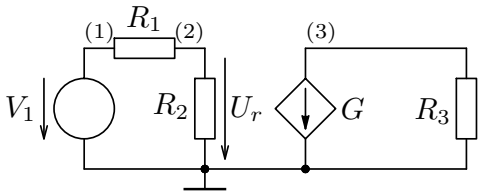
- celé číslo: 10, 16, 112567,
- desetinné číslo: 0.5, 15.165,
- racionální číslo: 1/15, 25/7,
- symbol: R, R3, D12,
- výraz: 1/G, 10*R/3,
- číslo s jednotkou: 1m, 5k, 3n, 16meg.

Jednotky, které umí simulátor zpracovat, jsou v tabulce 4.1. Netlist RC článku (obrázek 3.3) by mohl vypadat například takto:

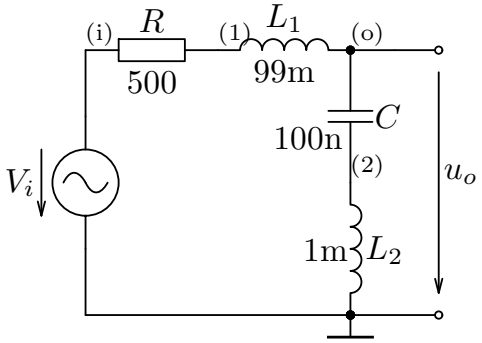
```
Basic RC
V1 1 0 dc 1 ac 1
C1 1 2 10n
R1 2 0 1k
```

Jednotka	název	symbol	hodnota
T	terra	T	10^{12}
G	giga	G	10^{9}
M	mega	meg	10^{6}
k	kilo	k	10^{3}
m	mili	m	10^{-3}
μ	micro	u	10^{-6}
n	nano	n	10^{-9}
p	pico	p	10^{-12}
f	femto	f	10^{-15}

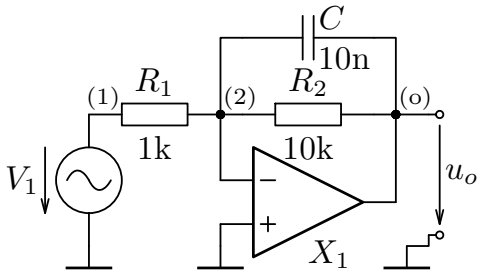
Tabulka 4.1: Tabulka jednotek v netlistu



Obrázek 4.1: Schéma obvodu s VCCS (staženo z [15])



Obrázek 4.2: Schéma RLLC filtru (staženo z [15])



Obrázek 4.3: Schéma obvodu s IOAmpem (staženo z [15])

Příklad použití napětím řízeného proudového zdroje (schéma na obrázku 4.1)

```
Obvod s VCCS
V1 1 0 dc 8
R1 1 2 8k
R2 2 0 2k
R3 3 0 4k
G 3 0 2 0 5m
```

U nezávislých zdrojů lze navíc zadat na konec netlistového řádku parametr

```
sin OFFSET AMPLITUDA FREKVENCE <ZPOŽDĚNÍ> <TLUMENÍ>
```

Tento parametr umožňuje simulovat časový průběh obvodu buzeného sinusoidálním zdrojem při spuštění tranzientní analýzy. Příklad jeho použití je k dispozici v netlistu RLLC filtru (schéma na obrázku 4.2)

```
RLLC
C o 2 100n
L2 2 0 1m
Vi i 0 dc 0 ac 1 0 sin 0 1 1.6k 0 0
R i 1 500
L1 1 o 99m
```

Netlist může také obsahovat podobvody (subcircuit). Podobvod je definice obvodové buňky, kterou může uživatel v obvodu použít vícekrát, aniž by musel celou buňku opisovat. Obecně lze podobvod definovat takto:

```
.subckt N1 <N2 N3 ...> NÁZEV <PARAMS: <P1=DEFAULT1 P2=DEFAULT2 ...>>
<Zde je potřeba definovat podobvod pomocí povolených prvků>
.ends NÁZEV
```

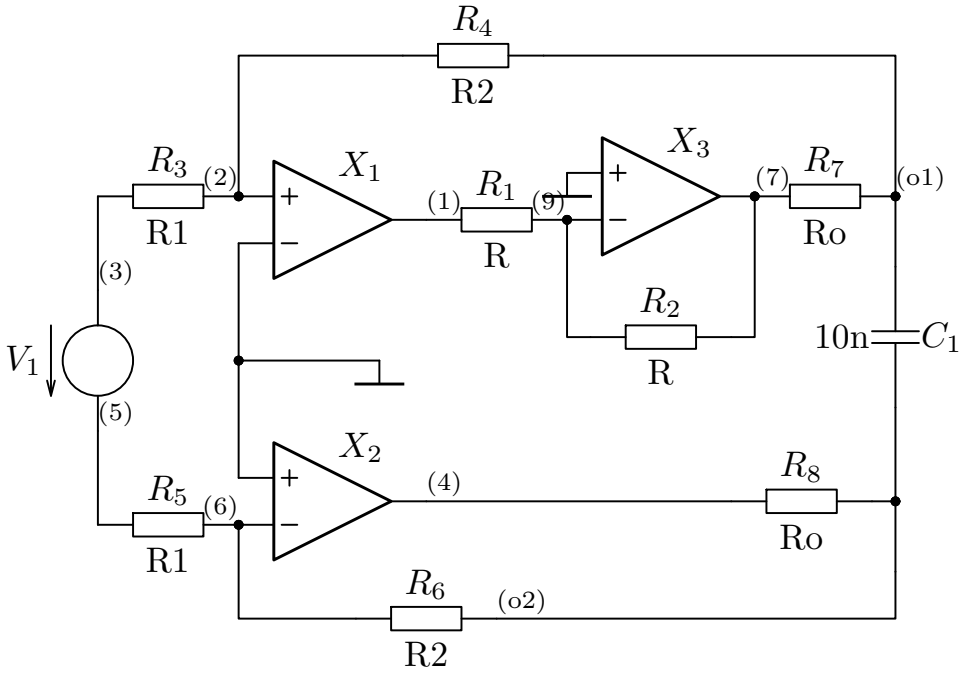
Subcircuit je instancován příkazem

```
XYYY N1 <N2 N3 ...> NÁZEV <PARAMS: <P1=VAL1 P2=VAL2 ...>>
```

Typickým případem použití je obvod, ve kterém se vícekrát vyskytuje model nějakého složitějšího prvku. Jednoduchý příklad jeho použití je obvod s jedním operačním zesilovačem (schéma na obrázku 4.3):

```
Operační zesilovač
V1 1 0 dc 0 ac 1 0 sin 0 1 1500 0 0
R1 1 2 1k
R2 2 o 10k
X1 o 0 2 idealAmpliferWithGroud
.subckt idealAmpliferWithGroud 1 2 3
A1 1 0 2 3
.ends idealAmpliferWithGroud
C 2 o 10n
```

Další příklad využití subcircuitu je netlist zapojení třech operačních zesilovačů (schéma na obrázku 4.4).



Obrázek 4.4: 3OAMP - Schéma obvodu s třemi IOAmpy (staženo z [15])

```
3OpAmp
R1 1 9 R
R2 9 7 R
R3 3 2 R1
R4 2 o1 R2
R5 5 6 R1
R6 6 o2 R2
V1 3 5 dc 1 ac 1
R7 7 o1 Ro
R8 4 o2 Ro
X1 2 0 1 linearOpamp PARAMS: A_par=A0
.subckt linearOpamp inplus d out PARAMS: A_par=100k
Gm 0 t inplus d {A_par}
Rtoa t 0 1
Ao out 0 t out
.ends linearOpamp
X2 0 6 4 linearOpamp PARAMS: A_par=A0
X3 0 9 7 linearOpamp PARAMS: A_par=A0
C1 o1 o2 10n
```

Kapitola 5

Metody algoritmizace analýzy elektrických obvodů

Elektronické obvody se řeší na základě

- prvního Kirchhoffova zákona 3.1 metodou uzlových napětí,
- nebo druhého Kirchhoffova zákona 3.2 metodou smyčkových proudů.

Každá z těchto metod má své výhody a při ručním výpočtu se obvykle využívají obě. Pro výběr metody vhodné pro počítačové zpracování je nejdůležitější, zda se dá algoritmizovat a jaká bude její výpočetní náročnost. V publikaci Computer Methods For Circuit Analysis [14] jsou vhodné metody vysvětleny. Lépe algoritmizovatelná je metoda uzlových napětí, jejíž variaci používá většina obvodových simulátorů. V této kapitole uvedu přehled vybraných maticových metod vhodných k počítačové simulaci a nastíním jejich odvození a použití. Všechny vybrané metody jsou založené na metodě uzlových napětí.

Jako první zmíním tableau formulaci, která je nejobecnější metodou obvodové formulace. Poté uvedu metody vyřazení nadbytečných rovnic z tableau formulace. Touto cestou se nakonec dostanu k dvěma metodám: modifikovaná metoda uzlových napětí (MMUN, v anglické literatuře MNA) a dvojgrafová modifikovaná metoda uzlových napětí (DMMUN). Mezi MMUN a DMMUN jsem vybral pro svůj simulátor metodu DMMUN, protože pro některé obvody vytváří menší soustavy rovnic. Pro porovnání jsem implementoval i tableau metodu. Metodu MMUN v tuto chvíli implementuje můj kolega Filip Špimr.

5.1 Tableau formulace

Tableau formulace je nejobecnější možnou obvodovou formulací. Je to v principu kolekce Kirchhoffových zákonů v maticovém tvaru a obecné rovnice obvodového prvku. Kirchhoffův zákon proudu je možné na základě 3.1 maticově zapsat jako

$$\mathbf{A}\mathbf{I}_b = \mathbf{0}, \tag{5.1}$$

zákon uzlových napětí pomocí 3.2 jako

$$\mathbf{V}_b - \mathbf{A}^t\mathbf{V}_n = \mathbf{0}, \tag{5.2}$$

a obecná rovnice obvodového prvku pomocí 3.5 jako

$$\mathbf{Y}_b\mathbf{V}_b + \mathbf{Z}_b\mathbf{I}_b = \mathbf{W}_b. \tag{5.3}$$

Rovnice 5.1, 5.2, 5.3 spojím do jedné maticové rovnice

$$\begin{bmatrix} \mathbf{E}_b & \mathbf{0} & -\mathbf{A}^t \\ \mathbf{Y}_b & \mathbf{Z}_b & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_b \\ \mathbf{I}_b \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}_b \\ \mathbf{0} \end{bmatrix}, \tag{5.4}$$

- $\mathbf{E}_b$ je jednotková diagonální matice. Její velikost je rovná počtu větví v obvodu (b).
- $\mathbf{Y}_b$ je admitanční matice. Je to čtvercová matice o velikosti b.
- $\mathbf{Z}_b$ je impedanční matice. Je to čtvercová matice o velikosti b.
- $\mathbf{V}_b$ je vektor, který obsahuje symboly napětí na prvcích. Počet jeho řádků je rovný počtu větví v obvodu.
- $\mathbf{I}_b$ je vektor, který obsahuje symboly proudů skrz prvky. Počet jeho řádků je rovný počtu větví v obvodu.
- $\mathbf{V}_n$ je vektor, který obsahuje symboly uzlových napětí. Počet jeho řádků je rovný počtu větví v obvodu.
- $\mathbf{W}_b$ je smíšený vektor proudů a napětí. Obsahuje hodnoty zdrojů napětí, zdrojů proudu a také vliv počátečních podmínek induktorů a kapacitorů. Počet jeho řádků je rovný počtu větví v obvodu.
- $\mathbf{A}$ je incidenční matice. Má tolik sloupců, kolik je větví v obvodu (b). Počet jejích řádků je rovný počtu uzlů v obvodu (n), referenční uzel se nezapočítává. Reprezentuje topologii daného obvodu a bude jí věnována samostatná sekce 5.2. $\mathbf{A}^t$ je transpozice matice $\mathbf{A}$.

Řešením této maticové rovnice jsou hodnoty všech symbolů obsažených ve vektorech $\mathbf{V}_b$, $\mathbf{I}_b$, $\mathbf{V}_n$. Nevýhodou tableau formulace je velký rozměr matic i pro malé obvody. Rozměr matice je

$$rank(\mathbf{M}) = 2b + n, \tag{5.5}$$

kde b je počet větví a n je počet uzlů (bez referenčního).

Tableau formulace je první metoda, kterou jsem zvolil pro implementaci. Rozhodl jsem se pro ni, protože je velice snadno algoritmizovatelná a je tedy relativně snadné otestovat její funkčnost. Byl to také ideální vstupní bod do této problematiky na začátku mé práce. Matici tableau formulace lze dále redukovat, čímž se dají získat kompaktnější formulace.

5.1.1 Příklad stavby tableau matice pro RC článek

Jako příklad postavím tableau metodou matici RC článku 3.3. Ten má tři větve a dva uzly (referenční uzel se nepočítá). Matice bude mít podle 5.5 velikost

$$rank(\mathbf{M}_{RC}) = 8. \tag{5.6}$$

Maticovou rovnici 5.4 je možné pro 3 větve a 2 uzly rozepsat obecně do tvaru

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -A_{11} & -A_{21} \\ 0 & 1 & 0 & 0 & 0 & 0 & -A_{12} & -A_{22} \\ 0 & 0 & 1 & 0 & 0 & 0 & -A_{13} & -A_{23} \\ Y_{11} & Y_{12} & Y_{13} & Z_{11} & Z_{12} & Z_{13} & 0 & 0 \\ Y_{21} & Y_{22} & Y_{23} & Z_{21} & Z_{22} & Z_{23} & 0 & 0 \\ Y_{31} & Y_{32} & Y_{33} & Z_{31} & Z_{32} & Z_{33} & 0 & 0 \\ 0 & 0 & 0 & A_{11} & A_{12} & A_{13} & 0 & 0 \\ 0 & 0 & 0 & A_{21} & A_{22} & A_{23} & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{b_1} \\ V_{b_2} \\ V_{b_3} \\ I_{b_1} \\ I_{b_2} \\ I_{b_3} \\ V_{n_1} \\ V_{n_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ W_{b_1} \\ W_{b_2} \\ W_{b_3} \\ 0 \\ 0 \end{bmatrix}. \tag{5.7}$$

V RC článku jsou 3 prvky. Rezistor, kapacitor a zdroj napětí. Pro rezistor podle tabulky stavebních rovnic 3.1 platí vztahy

$$V_b - R_b I_b = 0, \qquad Y_R = 1, \qquad Z_R = -R_b, \qquad W_R = 0. \tag{5.8}$$

Pro kapacitor s počáteční podmínkou platí vztahy

$$sC_bV_b - I_b = C_bV_0, \qquad Y_C = sC_b, \qquad Z_C = -1, \qquad W_C = C_bV_0. \tag{5.9}$$

Pokud chci ale získat jiný než tranzientní výsledek, musím použít vztahy bez počáteční podmínky

$$sC_bV_b - I_b = 0, \qquad Y_C = sC_b, \qquad Z_C = -1, \qquad W_C = 0. \tag{5.10}$$

DC, AC a TF analýza totiž probíhají po odeznění přechodových jevů a proto pro ně nesmím počáteční podmínku použít. Pro zdroj napětí platí

$$V_b = U_b, \qquad Y_V = 1, \qquad Z_V = 0, \qquad W_V = U_b. \tag{5.11}$$

Musím si určit pořadí zápisu prvků. Já zvolím jako první zdroj napětí, druhý rezistor a jako poslední kapacitor. Prvky incidenční matice se vyplňují tímto postupem:

- A_{nb} reprezentuje vztah uzlu n a prvku b
- Pokud je uzel n vstupní uzel prvku b, tak $A_{nb} = 1$.
- Pokud je uzel n výstupní uzel prvku b, tak $A_{nb} = -1$.
- Pokud není uzel n uzel prvku b, tak $A_{nb} = 0$.

$\mathbf{A}_{RC}$	V	R	C
1	1	0	1
2	0	1	-1

Tabulka 5.1: Incidenční matice RC článku zapsaná do tabulky

Incidenční matici RC článku jsem pro názornost zapsal do tabulky 5.1. Do maticové soustavy rovnic 5.7 dosadím hodnoty $\mathbf{A}$ a $\mathbf{A}^t$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 \\ Y_V & 0 & 0 & Z_V & 0 & 0 & 0 & 0 \\ 0 & Y_R & 0 & 0 & Z_R & 0 & 0 & 0 \\ 0 & 0 & Y_C & 0 & 0 & Z_C & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_V \\ V_R \\ V_C \\ I_V \\ I_R \\ I_C \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ W_V \\ W_R \\ W_C \\ 0 \\ 0 \end{bmatrix}. \tag{5.12}$$

V impedanční a admitanční matici jsem vynuloval členy mimo hlavní diagonálu. Pro obvody z jednoportových prvků mají vždy $\mathbf{Y}$ a $\mathbf{Z}$ diagonální tvar. To vyplývá z rovnice 3.5. Nakonec stačí dosadit hodnoty ze vztahů 5.8, 5.11 a 5.10 do matice 5.12.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -R & 0 & 0 & 0 \\ 0 & 0 & sC & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_V \\ V_R \\ V_C \\ I_V \\ I_R \\ I_C \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ U \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{5.13}$$

Řešením maticové rovnice 5.13 jsou výsledky pro všechny proudy a napětí v obvodu.

$$V_C = \frac{U}{CRs+1},$$
$$V_R = V_2 = \frac{CRUs}{CRs+1},$$
$$V_V = V_1 = U.$$
$$I_V = -\frac{CUs}{CRs+1},$$
$$I_R = \frac{CUs}{CRs+1},$$
$$I_C = \frac{CUs}{CRs+1}$$

5.2 Incidenční matice

Topologii obvodu je možné matematicky zapsat v podobě grafu. Takový graf můžeme reprezentovat incidenční maticí. Pro dva uzly a tři prvky má incidenční matice tvar

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix}. \tag{5.14}$$

Sloupce incidenční matice reprezentují prvky obvodu a řádky reprezentují uzly. Pro přehlednost zopakuji pravidla zápisu do incidenční matice:

- A_{nb} reprezentuje vztah uzlu n a prvku b.
- Pokud je uzel n vstupní uzel prvku b, tak $A_{nb} = 1$.
- Pokud je uzel n výstupní uzel prvku b, tak $A_{nb} = -1$.
- Pokud není uzel n uzel prvku b, tak $A_{nb} = 0$.

Referenční uzel (GND) se do incidenční matice nezavádí.

Některé prvky je možné topologicky popsat různě z pohledu proudu a napětí. Tento fakt mi umožní sestavit dva různé grafy, což otevře cestu k dalšímu zjednodušení maticové formulace obvodu. Takové grafy se nazývají I-graf a V-graf. Pro jejich sestavení jsem využil pravidla uvedená v publikaci [14].

1. Pokud není proud protékající větví důležitý (jeho výsledek mě nezajímá), zkolabuji hranu větve v proudovém grafu.
2. Pokud větví neprotéká proud, vymažu hranu prvku z I-grafu.
3. Pokud není napětí na větvi důležité, vymažu hranu prvku z napěťového grafu.
4. Pokud je napětí na větvi nulové, zkolabuji hranu větve v napěťovém grafu.

Tato pravidla vychází z principu chování proudu a napětí v obvodu. Podle těchto pravidel jsem vytvořil obrázek 5.1, který jsem s drobnými úpravami převzal z dříve zmíněné publikace [14]. Na základě oddělených I a V grafů lze odvodit dvojgrafovou modifikovanou metodu uzlových napětí (DMMUN).

5.3 Redukce matice

Tableau formulace je jednoduchá, ale obsahuje přebytečné informace. Pro pasivní prvky s dvěma uzly (rezistor, induktor, kapacitor) nám stačí znát pouze proud (nebo napětí) mezi uzly, na které je napojen. Druhou veličinu lze snadno vypočítat pomocí Ohmova zákona 3.8. Proto si můžeme dovolit

z tableau matice eliminovat všechna napětí na prvcích nebo proudy skrz prvky. Tím zásadně snížíme řád matice

$$rank(\mathbf{M}) = b + n. \tag{5.15}$$

Pokud chci do obvodu zavést prvky se čtyřmi uzly (dvojbrany – řízené zdroje, ideální operační zesilovač), je výhodné zachovat v matici proudy větví. Napětí na prvku lze vždy vypočítat rozdílem uzlových napětí

$$U_b = U_{n_{in}} - U_{n_{out}}. \tag{5.16}$$

Pokud rovnici 3.2 substituujeme do rovnice 3.5, získáme následující sadu rovnic

$$\mathbf{Y}_b\mathbf{A}^t\mathbf{V}_n + \mathbf{Z}_b\mathbf{I}_b = \mathbf{W}_b, \tag{5.17}$$

$$\mathbf{A}\mathbf{I}_b = 0. \tag{5.18}$$

Spojeny do jedné matice vypadají takto

$$\begin{bmatrix} \mathbf{Y}_b\mathbf{A}^t & \mathbf{Z}_b \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{V}_n \\ \mathbf{I}_b \end{bmatrix} = \begin{bmatrix} \mathbf{W}_b \\ \mathbf{0} \end{bmatrix}. \tag{5.19}$$

Tato metoda snižuje řád maticové rovnice, aniž by znatelně komplikovala implementaci. Protože maticové násobení je výpočetně náročná operace, je vhodné sestavit člen v pravém horním rohu matice rovnou. Pomocí vhodného algoritmu mohu při stavbě obvodových matic sestavit všechny maticové násobky nebo transpozice, aniž bych musel provést danou matematickou operaci. Díky tomu bude výpočet rychlejší.

5.4 Modifikovaná metoda uzlových napětí MMUN

V předchozí sekci jsem ukázal jak z matice eliminovat všechna napětí na prvcích. Tato metoda navíc definuje pravidla pro eliminaci i některých proudů. Tím ještě víc snižuje řád obvodové matice, aniž by ztratila užitečnou informaci. Prvky obvodu musíme nejprve rozdělit do dvou kategorií:

1. Prvky, které mají admitanční/impedanční definici: rezistor, induktor a kapacitor.
2. Prvky, které ji nemají: nezávislé zdroje, řízené zdroje a IOAMP.

Pro první kategorii není potřeba zachovat ani napětí na větvi, ani proud větví. Napětí na větvi mohu snadno dopočítat pomocí rozdílu uzlových napětí 5.16. Proud větví mohu získat pomocí Ohmova zákona 3.8.

Pro druhou kategorii nepotřebuji zachovávat napětí na prvcích. Mohu ho snadno dopočítat jako u první kategorie. Musím zachovat proudy větví, protože pro jejich výpočet nelze použít Ohmův zákon 3.8. Není totiž k dispozici jejich impedanční popis.

Obecně lze obvodovou matici MMUN zapsat jako

$$\begin{bmatrix} \mathbf{Y}_{n_1} & \mathbf{A}_2 \\ \mathbf{Y}_2\mathbf{A}_2^t & \mathbf{Z}_2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_n \\ \mathbf{I}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_n \\ \mathbf{W}_2 \end{bmatrix}, \tag{5.20}$$

kde:

- $\mathbf{I}_2$ obsahuje proměnné proudů prvků, které nemají admitanční reprezentaci.
- $\mathbf{V}_n$ obsahuje proměnné uzlových napětí.
- $\mathbf{Y}_2$, $\mathbf{Z}_2$ a $\mathbf{W}_2$ reprezentují popis prvků z druhé kategorie (5.3).
- $\mathbf{A}_2$ je incidenční matice prvků z druhé kategorie a $\mathbf{A}_2^t$ její transpozice.

$\mathbf{Y}_{n1}$ a $\mathbf{J}_n$ jsou zkratky pro

$$\mathbf{Y}_{n1} = \mathbf{A}_1\mathbf{Y}_1\mathbf{A}_1^t, \qquad \mathbf{J}_n = -\mathbf{A}_z\mathbf{J}_z. \tag{5.21}$$

- $\mathbf{Y}_1$ je admitanční matice prvků první kategorie.
- $\mathbf{A}_1$ je incidenční matice prvků první kategorie.
- $\mathbf{A}_z$ je incidenční matice nezávislých zdrojů.
- $\mathbf{J}_z$ obsahuje hodnoty nezávislých zdrojů.

Tato metoda je z programátorského hlediska náročnější než předchozí. Díky další eliminaci matice ale zmenšuje výslednou obvodovou matici na

$$rank(\mathbf{M}) = b_2 + n, \tag{5.22}$$

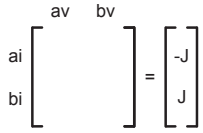
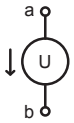
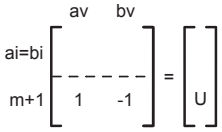
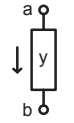
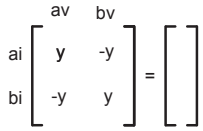
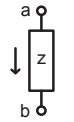
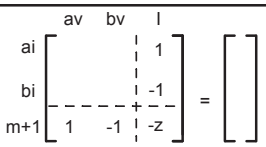
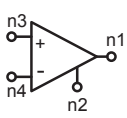
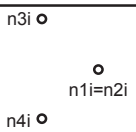
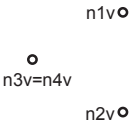
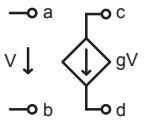
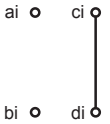
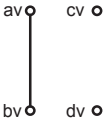
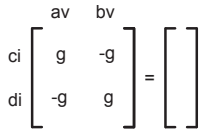
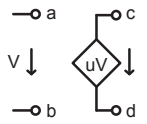
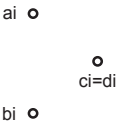
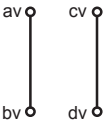
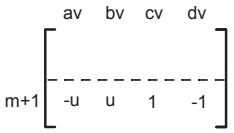
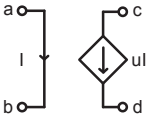
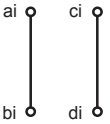
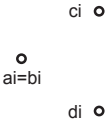
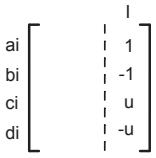
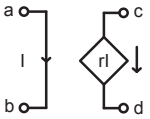
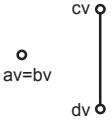
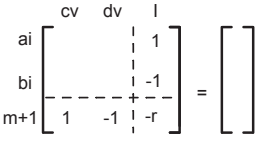
kde b_2 je počet prvků v druhé kategorii. Blíže se této metodě věnují práce [14] a [19].

5.5 Dvojgrafová modifikovaná metoda uzlových napětí

V minulých sekcích jsem ukázal, jak se dá zjednodušit tableau matice pomocí vyřazení nadbytečných rovnic. Další úprava, kterou mohu provést, je využití I a V grafů místo jednoho obecného (viz sekci 5.2). Pro přehlednost se vrátím zpět na začátek k tableau formulaci a upravím ji pro I a V graf. Tableau formulace je složená ze tří maticových rovnic KCL (5.1), KVL (5.2) a stavební rovnice prvků (5.3). KCL a KVL obsahují v původní formulaci obecnou incidenční matici. I-graf se dosadí do KCL a V-graf do KVL.

$$\mathbf{A}_i\mathbf{I}_b = \mathbf{0}, \tag{5.23}$$

$$\mathbf{V}_b - \mathbf{A}_v^t\mathbf{V}_n = \mathbf{0}. \tag{5.24}$$

Prvek	Symbol	I-graf	V-graf	Zápis do obvodové matice
Zdroj proudu				
Zdroj napětí				
Admitance				
Impedance				
IOAmp				U IOAMPu není třeba nic zapisovat do matice. Jeho chování je plně popsáno v grafech.
VCCS				
VCVS				
CCCS				
CCVS				

Obrázek 5.1: Návod k zápisu prvků do matice DMMUN metodou (převzato z [14])

kde $\mathbf{A}_v$ je V-graf a $\mathbf{A}_i$ je I-graf. Pro přehlednost opíšu i stavební rovnici prvků

$$\mathbf{Y}_b\mathbf{V}_b + \mathbf{Z}_b\mathbf{I}_b = \mathbf{W}_b.$$

Uvedené rovnice mohu opět posbírat do jedné maticové rovnice

$$\begin{bmatrix} \mathbf{E}_b & \mathbf{0} & -\mathbf{A}_v^t \\ \mathbf{Y}_b & \mathbf{Z}_b & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_i & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_b \\ \mathbf{I}_b \\ \mathbf{V}_n \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{W}_b \\ \mathbf{0} \end{bmatrix}. \tag{5.25}$$

Matici 5.25 mohu dále eliminovat stejným způsobem jako standardní tableau formulaci. Kvůli využití dvou grafů je ale formální formulace matice DMMUN mnohem náročnější než v případě předchozích metod. Nebudu v této práci obecnou matici DMMUN odvozovat i z toho důvodu, že se z praktického hlediska jedná převážně o pomůcku k odvození pravidel pro zápis obvodových prvků do matice. Odvození je provedeno v publikaci [14]. Pravidla pro zavedení prvků do obvodové matice DMMUN jsem shrnul na obrázku 5.1. Pro příklad uvedu DMMUN matici pro RC článek (schéma na obrázku 3.3):

$$\begin{bmatrix} -sC & sC + \frac{1}{R} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0 \\ U \end{bmatrix}. \tag{5.26}$$

V tabulce 5.2 jsem matici 5.26 obecně popsal, aby bylo jasné jak jsem ji z tabulky 5.1 sestavil.

	1_v	2_v
2_i	$-Y_C$	$Y_C + Y_R$
$m+1$	1	0

Tabulka 5.2: Sestavení DMMUN matice pro RC článek

S výjimkou impedance a CCVS žádný ze zavedených prvků nezvětší obvodovou matici nad počet uzlů. Právě proto je vhodné zavádět resistory, induktory a kapacitory admitančně a ne impedančně. Řešením obvodové matice vytvořené pomocí DMMUN je často nekompletní seznam uzlových napětí a jen některé proudy. Je proto nezbytné při stavbě matice zaznamenat, které uzly byly ve V-grafu ztotožněny, abychom mohli doplnit seznam uzlových napětí. Z doplněného seznamu uzlových napětí je snadné dopočítat napětí na prvcích a proudy, které jimi protékají.

Metoda DMMUN je ze všech uvedených implementačně nejsložitější a vyžaduje systematické dopočítávání některých uzlových napětí, proudů a napětí na prvcích. Díky tomu je ale matice obvodových rovnic kompaktní a umožňuje analýzu mnohem větších obvodů než předchozí metody. V této podobě také metoda nepočítá proudy skrz napěťové zdroje, to je možné napravit upraveným zavedením napěťového zdroje. Takovou úpravu jsem zatím neprovedl, ale v budoucnu ji do simulátoru přidám. Největší výhodou této metody je, že IOAMPy se do matice vůbec nezapisují, ale jen ztotožní některé uzly. Díky tomu každý přidaný IOAMP zmenšuje obvodovou matici o jedna. Je to druhá metoda, kterou jsem se rozhodl implementovat.

Část II

Praktická část

Kapitola 6

Implementace simulátoru

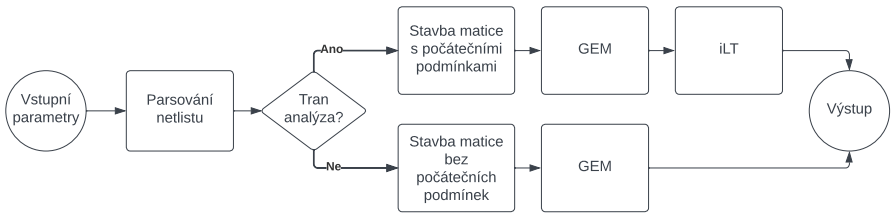
Na úvod připomenu, že jsem svůj simulátor navrhoval k použití simulátoru GEEC. Ten vyžaduje pro zadaný obvod vždy kompletní analýzu (seznam všech napětí na větvích a proudů větvemi) zvoleného typu (DC, AC, TF, tran). V tuto chvíli jsou v mém simulátoru plně implementované dvě metody stavby obvodové matice: tableau 5.1 a DMMUN 5.5.

Ze začátku jsem se rozhodl implementovat co nejsnazší metodu, která by mi umožnila dobře pochopit principy maticových metod obvodové simulace a promyslet strukturu simulátoru. Dalo by se říct, že tableau formulace sloužila hlavně jako prototyp pro metodu DMMUN. Nezávisle na použité metodě stavby maticové rovnice je totiž většina programu stejná. Jediná zásadní změna je v procesu stavby obvodové matice a zpracování jejích výsledků. Naprogramoval jsem proto nejprve tableau formulaci a kolem ní celý zbytek simulátoru. Snadnost implementace tableau metody mi umožnila efektivní proces debuggingu celého balíčku a tak zkrátila čas nutný k implementaci simulátoru. Poté, co jsem měl první funkční a otestovanou verzi simulátoru, jsem se rozhodl znovu prozkoumat složitější metody stavby obvodových matic. Jako druhou metodu jsem implementoval DMMUN. V porovnání s balíčkem Lcapy je analýza v SymCiPy rychlejší nezávisle na použité metodě, což demonstruji na konci práce v sekci 7.4.

V této kapitole především popíšu a vysvětlím postupy, kterými jsem navrhnul a naprogramoval SymCiPy. Pro efektivní návrh programu je nejprve třeba dobře definovat jeho vstupy a výstupy. Vstupem jsou v tomto případě parametry důležité pro analýzu:

- Netlist obsahující definici analyzovaného obvodu.
- Typ analýzy (TF, DC, AC, tran).
- Symbolický nebo semisymbolický výstup.

Nakonec je důležité zmínit, že veškeré matematické operace můj simulátor provádí pomocí SymPy. Přibližná struktura simulátoru je naznačena na obrázku 6.1.



Obrázek 6.1: Flowchart procesu simulace obvodů

6.1 Struktura balíčku

Můj simulátor je složen z několika různých souborů. Balíček je díky tomu rozdělen do tématických celků pro větší přehlednost kódu a snažší vývoj:

- *analysis.py* obsahuje implementace jednotlivých analýz (DC, AC, TF, tran) a metod tvorby obvodové matice (tableau, DMMUN). Obsahuje také pomocné funkce pro pohodlné použití balíčku.
- *parse.py* je soubor funkcí určených k překladu netlistu do formátu vhodného pro další zpracování knihovnou SymPy.
- *component.py* obsahuje definice datových objektů, které parse.py využívá pro uložení jednotlivých komponentů. V těchto objektech jsou uloženy všechny prvky obvodu.
- *laplace.py* je soubor funkcí, které provedou zpětnou Laplaceovu transformaci (iLT). Tento soubor se v tuto chvíli nepoužívá, protože jsem přešel na iLT implementaci ze SymPy.
- *pole_zero.py* obsahuje funkce pro pole-zero (PZ) analýzu.
- *utils.py* je kolekce pomocných funkcí používaných napříč celým balíčkem.

Zvolil jsem přístup objektového programování. Hlavní část balíčku je třída, pomocí které uživatel spustí simulaci. Uvedu jí s hlavními parametry:

```
AnalyseCircuit(netlist, analysis_type="DC",
               method="two_graph_node", symbolic=True)
```

- *netlist* vyžaduje string obsahující netlist.
- *analysis_type* vyžaduje string obsahující název žádané analýzy.
- *method* vyžaduje string obsahující název metody simulace.
- *symbolic* vyžaduje Bool a určující jestli bude analýza symbolická nebo semisymbolická.

V průběhu inicializace třídy dojde k přeložení (parsování) netlistu. Poté je z načteného obvodu vytvořena soustava rovnic v podobě matice (viz kapitolu 5). Pomocí Gaussovy eliminace je matice vyřešena a výsledky dále zpracovány do formátu vhodného pro GEEC.

Po inicializaci třídy AnalyseCircuit se spustí parser netlistu (více v sekci 6.3). Jeho výstup se do objektu uloží jako tři proměnné:

- *self.components* obsahuje dictionary komponentů.
- *self.node_dict* obsahuje dictionary s indexy jednotlivých uzlů obvodu. Tato informace je zásadní pro správné sestavení obvodové matice.
- *self.node_count* je integer obsahující počet uzlů obvodu.

Poté se vytvoří seznam SymPy symbolů pro uzlová napětí. Jako poslední se spustí funkce *self._analyse*(). Ta pomocí *self._build_system_eqn*() sestaví obvodovou matici podle zadané metody, vyřeší ji a navrátí tři datové objekty:

- *self.eqn_matrix* obsahuje sestavenou maticovou rovnici, která dále slouží pouze k případné vizualizaci uživatelem.
- *self.solved_dict* je dictionary, ve kterém jsou uloženy výsledky analýzy. Klíčem je symbol obvodové veličiny (napětí, proud) a záznamem je SymPy funkce. Není určen k přímému zobrazení uživatelem. Je dále zpracováván uživatelskými funkcemi, o kterých budu psát v sekci 6.5.
- *self.symbols* je seznam všech SymPy symbolů napětí a proudů obsažených v *self.solved_dict*

Funkce *self._analyse*() nejprve spustí *self._build_system_eqn*(), která vrátí postavenou obvodovou matici a seznam korespondujících SymPy symbolů. Návratové hodnoty *self._build_system_eqn*() jsou následně použity jako vstup pro knihovnu SymPy, která obvodovou matici vyřeší s návratovou hodnotou *solved_dict*. Poté funkce *self._analyse*() vybere jednu ze čtyř cest, podle toho, jakou analýzu uživatel vybral (DC, AC, TF, tran). O těchto cestách bude více v sekci 6.4. Tímto je analýza dokončená a uživatel má k dispozici objekt s uloženými výsledky. Uživatelskými funkcemi může pohodlně získat výsledky, které zrovna potřebuje.

6.2 Podporované prvky

Jak jsem již zmínil v kapitole 4, můj simulátor podporuje následující ideální prvky: rezistor, induktor, kapacitor, vázané induktory, nezávislý zdroj napětí a proudu, ideální operační zesilovač a řízené zdroje. V budoucích verzích přibydou linearizované diody a tranzistory. V souboru *component.py* je definovaná obecná třída (parent class) obsahující společné parametry všech druhů prvků. Na jejím základě jsou dále definovány jednotlivé třídy (child class) pro každý podporovaný prvek. Celý obvod je díky tomu možné uložit jako slovník objektů indexovaný podle jména prvku. Tento přístup je velmi

výhodný pro případné budoucí rozšíření balíčku o nové uživatelské funkce a zásadně zjednodušuje budoucí vývoj. Je také přehledný pro uživatele.

6.3 Parser

Pro tento projekt jsem napsal vlastní parser netlistu. Formát netlistu je popsaný v sekci 4. Tato část kódu prošla mnoha úpravami a variantami. Implementačně nejnáročnější se ukázalo být zpracování subcircuitů. Parsing je rozdělen do tří hlavních funkcí:

- $parse()$,
- $unpack()$,
- $parse\_subcircuits()$.

Funkce $parse()$ je spuštěna vždy na začátku inicializace třídy $AnalyseCircuit()$. Nejprve zavolá funkci $parse\_subcircuits()$, která prohlédne netlist a všechny nalezené subcircuity „rozbalí“ pomocí funkce $unpack()$. Vytvoří se tak nový netlist, který obsahuje stejný obvod jako původní netlist, ale bez subcircuitů. Parser se bohužel snadno neobejde bez dvojité iterace netlistu, protože definice subcircuitu může být umístěna kdekoliv v netlistu a daný subcircuit může být použit více než jednou. Metoda „rozbalení“ je snadno algoritmizovatelná, přehledná a poskytuje tak vysokou stabilitu řešení. V praxi tvoří tato rutina zanedbatelnou část celkového výpočetního času. Samotné parsování netlistu probíhá v cyklu, během kterého se prvek po prvku celý netlist přeloží do slovníku objektů. Typ prvku je identifikován podle prvního charakteru každého řádku. Následně jsou uloženy jeho uzly a další parametry. Uzly je třeba označit číselnými indexy pro budoucí zpracování, protože do netlistu může být zadaný název uzlu i písmeno. Většina prvků má kromě názvu a uzlů alespoň jeden další parametr. Může to být například rezistence, kapacita, indukčnost nebo hodnota napětí a proudu. U takových prvků je nutné hodnotu správně načíst pro další zpracování. K tomu slouží funkce $value\_enum()$, která pomocí několika dalších funkcí identifikuje, zda se jedná o numerickou či symbolickou hodnotu a správně ji uloží do proměnné.

Uživatel může zadat do netlistu numerickou hodnotu prvku. V takovém případě je vygenerována symbolická hodnota na základě jeho názvu. Pokud je v netlistu přímo zadána symbolická hodnota, tak se jen převede pomocí SymPy do formátu symbol. V případě nezávislých zdrojů se dynamicky načítají hodnoty pro DC, AC a tran analýzu. Záleží na tom, jaké parametry byly zadané v netlistu. Pokud není zadaná žádná tranzientní hodnota, použije simulátor pro tran analýzu DC hodnotu. Výstupem parsování je obvod přeložený do objektového formátu výhodného pro další zpracování. V této fázi program nerozlišuje mezi symbolickou a semisymbolickou analýzou, načítá tedy z netlistu veškeré dostupné informace. Pokud netlist neodpovídá definovaným parametrům, je program ukončen.

6.4 Implementované typy analýzy

Můj simulátor momentálně nabízí DC, AC, TF a tran analýzu. Z výsledků TF analýzy lze pomocí uživatelských funkcí získat PZ analýzu. Nejprve vždy proběhne TF analýza, která se dále upravuje do tvaru požadované analýzy (viz sekci 3.2). DC analýzu by bylo optimální dělat zvlášť, protože takto je nutné počítat pro obvody s induktory a kapacitory limity. Je to ale práce navíc a obvody, na kterých se provádí DC analýza, často induktory a kapacitory neobsahují. Do budoucna plánuji dělat DC analýzu zvlášť. K dispozici je jak plně symbolická, tak i semisymbolická varianta všech analýz.

6.5 Uživatelské funkce

Můj simulátor disponuje několika uživatelskými funkcemi pro práci s výsledky simulace. Jednu po druhé je v této sekci krátce představím a popíšu.

- $self.get\_node\_voltage(node)$ má jako vstup string obsahující název obvodového uzlu. Její výstup je dané uzlové napětí.
- $self.component\_voltage(name)$ má jako vstup string obsahující název prvku. Její výstup je napětí na prvku.
- $self.component\_current(name)$ má jako vstup string obsahující název prvku. Její výstup je proud na prvku.
- $self.component\_values(name)$ má stejný vstup jako dvě předchozí funkce. Její výstup je dictionary obsahující proud a napětí prvku indexované názvy daných symbolů. Pokud uživatel nezadá parametr name, vrátí funkce dictionary všech napětí a proudů prvků.
- $self.node\_voltages()$ vrátí uživateli slovník všech uzlových napětí.
- $self.transfer\_function(node1, node2)$ má jako vstup dva uzly a jako výstup přenosovou funkci mezi nimi.
- $self.count\_components()$ spočítá počet komponentů v obvodu.

6.6 Implementace metod stavby soustavy rovnic

Stavbu obvodových matic můj simulátor spouští pomocí funkce

```
self._build_system_eqn()
```

Podle toho, jakou metodu uživatel při spuštění simulace zvolil, se vybere buď tableau metoda, nebo DMMUN metoda. Nejprve popíšu implementaci tableau metody a poté implementaci DMMUN.

6.6.1 Implementace tableau

Nejprve se na základě seznamu prvků, které zpracoval parser, vypočte velikost výsledné obvodové matice. Poté se pomocí SymPy inicializuje správně velká matice plná nul. Do levého horního rohu se vyplní jednotková diagonální submatice $\mathbf{E}_b$. Vytvoří se také pravá strana rovnice (vektor proudů a napětí zdrojů). Nakonec se v cyklu systematicky vyplní matice prvek po prvku. Metodiku jsem již popsal v sekci 5.1. Matice je pomyslně rozdělena do sektorů a každý z nich vyplněn. Tyto sektory jsou submatice $\mathbf{A}$, $-\mathbf{A}^t$, $\mathbf{Y}_b$, $\mathbf{Z}_b$ a $\mathbf{W}_b$.

- Pro každý prvek se zapíše vstup do incidenční matice (dole uprostřed, $\mathbf{A}$).
- Do její transpozice (pravý horní roh, $-\mathbf{A}^t$) se zapíše ten samý údaj, jen transponovaně. To je velmi snadné, stačí zaměnit řádek a sloupec zápisu.
- Do admitanční ($\mathbf{Y}$) a impedanční ($\mathbf{Z}$) matice se doplňuje zároveň. Doplní se do nich na základě stavební rovnice prvku 5.3.
- Pokud se jedná o nezávislý zdroj, je jeho hodnota zapsána do $\mathbf{W}_b$.

Postupně je také generován vektor symbolů, které definují neznámou jednotlivých sloupců. Je to seznam symbolů reprezentujících napětí prvků, proudy prvků a uzlová napětí. Výsledná obvodová matice se spojí s vektorem pravé strany a je vložena společně s vektorem neznámých do SymPy funkce

```
sympy.solve_linear_system(eqn_matrix, *symbols)
```

Ta metodou Gaussovy eliminace (GEM) soustavu rovnic vyřeší. Kromě GEM jsem zvažoval metodu inverze a LU dekompozici [1]. Jejich výhodou je, že po prvním vyřešení pro jednu pravou stranu je možné řešit problém i pro jiné pravé strany se sníženou časovou náročností. To je ale pro symbolickou simulaci ve většině případů zbytečné. Tyto metody mají využití hlavně v numerické simulaci. Výsledkem GEM je slovník všech proměnných s jejich výsledkem.

6.6.2 Implementace DMMUN

Metoda DMMUN probíhá ve dvou cyklech.

- V prvním cyklu se vytvoří seznam všech uzlů prvků. Paralelně s ním se ukládá informace o totožnosti uzlů. Pokud je pro daný prvek v tabulce 5.1 v I nebo V grafu ztotožnění uzlů, tak se totožnost uzlů uloží pomocí funkce $self.collapse()$.
- Mezi cykly proběhne systematické zkolabování uzlů v I a V seznamech.
- V druhém cyklu se do obvodové matice postupně vyplní informace o prvcích na základě tabulky 5.1.

Řešení výsledné maticové rovnice je stejné jako pro tableau matici (viz sekci 6.6.1). Při analýze metodou DMMUN nezíská simulátor hodnoty napětí a proudy prvků přímo vyřešením maticové rovnice (viz sekci 5.5). Musí tyto hodnoty dopočítat z uzlových napětí.

Kapitola 7

Výsledky práce

7.1 Testování výsledků simulace

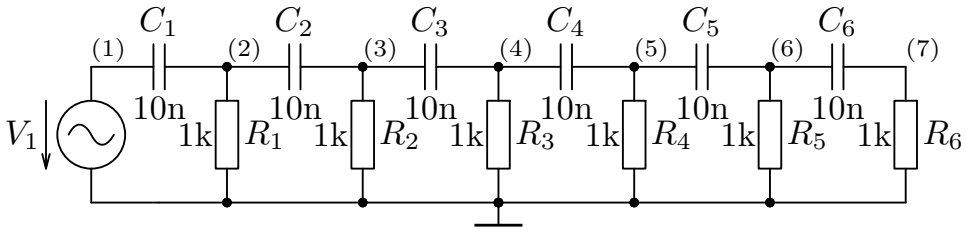
Správnost symbolických i semisymbolických výsledků simulace byla při vývoji pravidelně testována sérií testovacích skriptů, které jsem pro tento balíček napsal. Čerpají z databáze obvodových netlistů a databáze ověřených správných výsledků. Obě databáze jsem vytvořil pomocí GEECu.

7.2 Srovnání rychlosti TF analýzy tableau a DMMUN na vybraných obvodech

Nejdříve jsem testoval obě implementované metody na vybraných obvodech. Testoval jsem je pro TF analýzu. Výsledky jsou uvedeny v tabulce 7.1. Zajímavý výsledek je, že pro malé obvody (RC, RLLC) je tableau simulace rychlejší. Je to nejspíš proto, že

- obvodová tableau matice je pro malé obvody stále relativně malá (viz maticovou rovnici 5.13).
- DMMUN metoda musí na konci dopočítávat výsledky a upravovat jejich tvar pro lepší čitelnost. To je výpočetně náročné.
- Tableau metoda poskytuje rovnou čitelné výsledky a není jí proto třeba příliš upravovat.

Pro kaskádu šesti RC článků tableau metoda v rozumném čase vůbec nedojde k výsledku. DMMUN metoda získá výsledek za $23s$. Velký rozdíl je vidět u obvodu s více operačními zesilovači 4.4. DMMUN má poloviční výpočetní čas v porovnání s tableau. Potvrzuje to silnou stránku DMMUN – simulace obvodů s operačními zesilovači. Operační zesilovač totiž do DMMUN matice nic nepřidává a zároveň ubere jedno uzlové napětí z matice (viz tabulku 5.1).



Obrázek 7.1: 6RC - Schéma kaskády šesti RC článků (staženo z [15])

Obvod	tableau, t_t (s)	DMMUN, t_d (s)
RC (viz obrázek 3.3)	0,02	0,03
RLLC (viz obrázek 4.2)	0,1	0,15
6RC (viz obrázek 7.1)	timeout	23
3OAMP (viz obrázek 4.4)	5,85	2,25

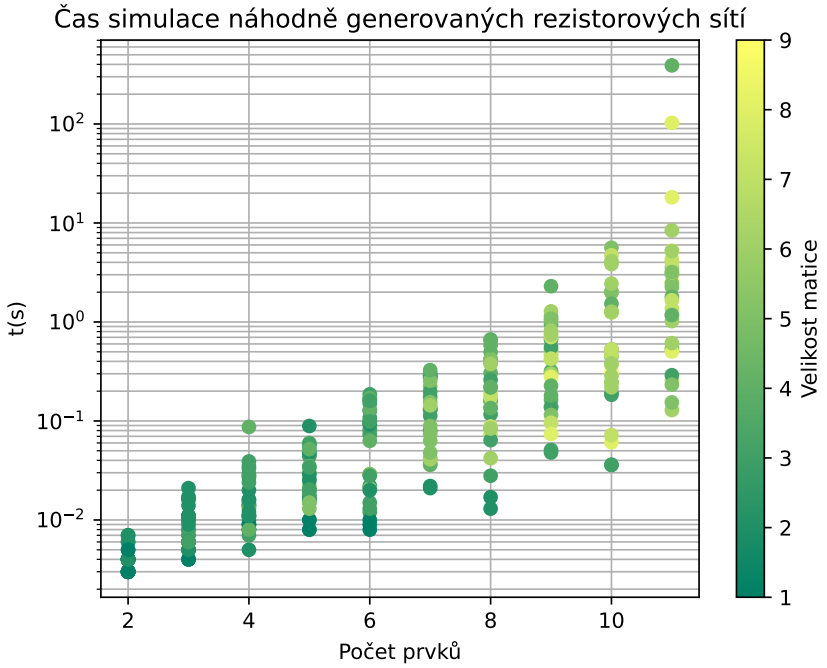
Tabulka 7.1: Přehled časů TF simulace vybraných obvodů

7.3 Statistické srovnání rychlosti metod na náhodně generovaných rezistorových sítích

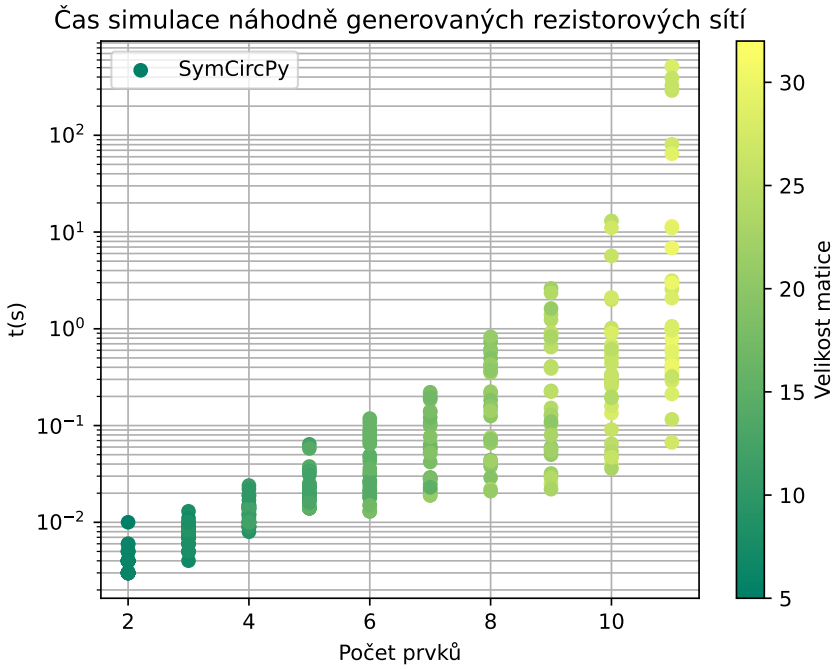
Abych demonstroval schopnosti mého simulátoru, provedl jsem statistickou simulaci závislosti výpočetního času na počtu prvků obvodu. Využil jsem k tomu generátor rezistorových sítí implementovaný v balíčku Lcapy [11]. Generátor jsem musel upravit, aby generoval netlisty ve formátu, který můj simulátor umí zpracovat. Každý vygenerovaný obvod obsahuje jeden napěťový zdroj a náhodně zapojenou permutaci zadaného počtu rezistorů. Nejprve jsem provedl simulaci 300 náhodně generovaných obvodů pro metody DMMUN a tableau. Pro přehlednost uvedu parametry simulace:

- Všechny hodnoty prvků jsou symbolické.
- Čas je měřen od počátku simulace do vypočítání všech hodnot napětí a proudů na prvcích. Pro tento průběh jsem se rozhodl, protože takový výstup požaduje simulátor GEEC, pro který je můj simulátor navržen.
- Pro každý počet prvků je vygenerováno 30 obvodů. Nejsměrodatnější jsou tedy pro každý počet prvků výsledky s vyšší časovou náročností.
- Simuloval jsem obvody obsahující 2 až 11 prvků (1 až 10 rezistorů a jeden zdroj napětí).

Na obrázku 7.2 je vidět výsledek simulace pro metodu DMMUN a na obrázku 7.3 pro metodu tableau. Body grafu jsou barevně označeny podle velikosti simulované obvodové matice. Abych mohl porovnat výsledky obou simulací mezi sebou, vypočítal jsem pro každý počet prvků maximum simulačních časů. Počítal jsem jen maxima, protože se z povahy simulace jedná o jediný směrodatný údaj. Mnoho z vygenerovaných obvodů může mít totiž triviální



Obrázek 7.2: Symbolická simulace pomocí metody DMMUN



Obrázek 7.3: Symbolická simulace pomocí metody tableau

řešení. Zapsal jsem je do tabulky 7.2. Shrnu všechny zajímavé poznatky a výsledky této simulace do seznamu:

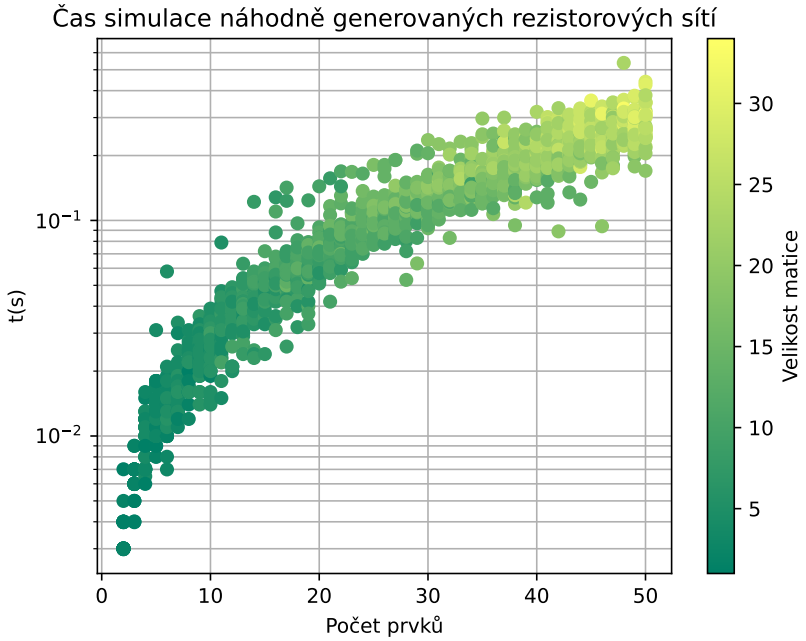
Počet prvků	Tableau, t_t (s)	DMMUN, t_d (s)	$\frac{t_t}{t_d}$
2	0,01	0,012	0,83
3	0,013	0,019	0,68
4	0,024	0,062	0,39
5	0,064	0,106	0,6
6	0,118	0,173	0,68
7	0,221	0,364	0,61
8	0,826	0,680	1,21
9	2,616	1,445	1,81
10	13,03	19,75	0,66
11	515,2	58,64	8,88

Tabulka 7.2: Maximální čas simulace v závislosti na počtu prvků

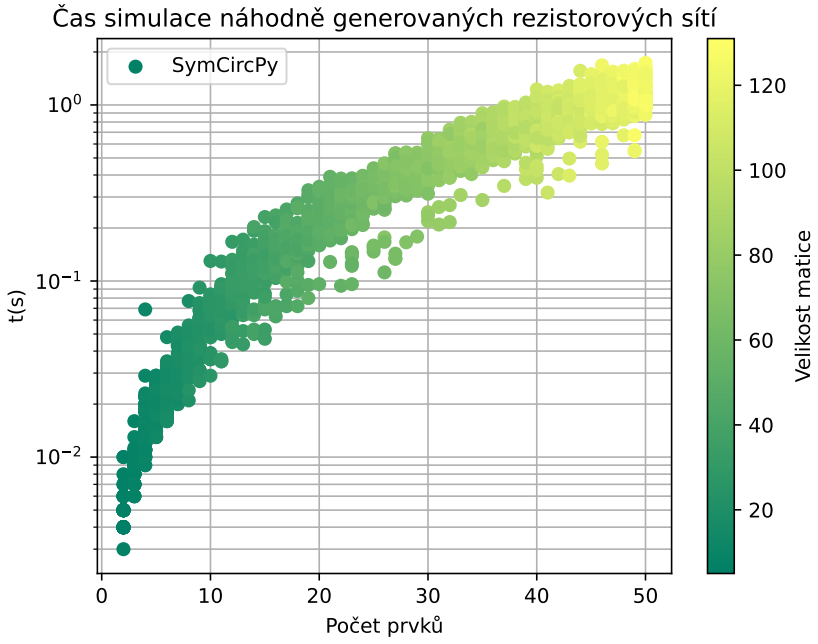
- 11 prvků je přibližný praktický limit čistě symbolické simulace pomocí metody tableau. Na základě tabulky 7.1 jsem již ověřil, že metoda DMMUN vypočítá čistě symbolicky i některé obvody s 12 symbolickými prvky.
- Rychlost metody DMMUN není příliš závislá na velikosti obvodové matice. Je to proto, že hodnoty, které se nevypočítají přímo z matice, je třeba dopočítat na konci simulace. Rychlost tableau je jednoznačně závislá na velikosti obvodové matice.
- Při symbolické simulaci na malém počtu prvků je tableau rychlejší. Pro větší počet prvků je rychlejší DMMUN.
- Pokud by v simulovaných obvodech byly operační zesilovače, zvítězila by jednoznačně metoda DMMUN. Generovat takové obvody by ale bylo složité.

Provedl jsem také semisymbolickou simulaci. Pro běžné využití simulátoru je tato forma simulace nejdůležitější. Na obrázcích 7.4 a 7.5 je její výsledek vynesený do grafů. Semisymbolická simulace byla provedena za následujících předpokladů:

- Všechny hodnoty rezistorů jsou numerické (1000Ω), hodnota zdroje je symbolická.
- Čas je měřen od počátku simulace do vypočítání všech hodnot napětí a proudů větví.
- Pro každý počet prvků je vygenerováno 50 obvodů.
- Simuloval jsem obvody obsahující 2 až 50 prvků (1 až 49 rezistorů).



Obrázek 7.4: Semisymbolická simulace pomocí metody DMMUN s jednou proměnnou. Simulováno na 2500 náhodně generovaných obvodech

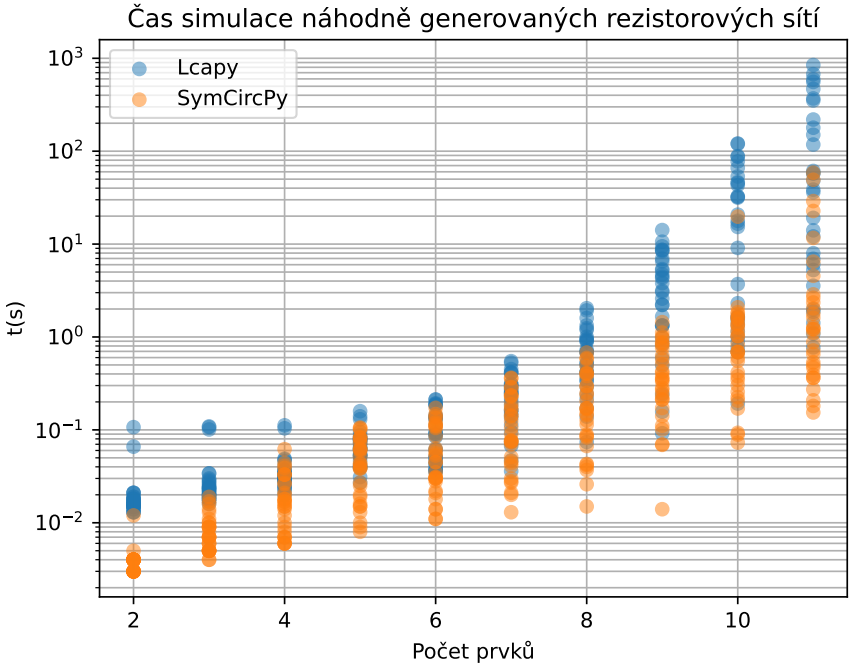


Obrázek 7.5: Semisymbolická simulace pomocí metody tableau s jednou proměnnou. Simulováno na 2500 náhodně generovaných obvodech

Výsledky semisymbolické simulace dále potvrzují, že čas simulace je převážně závislý na počtu proměnných, a ne tolik na velikosti matice. Semisymbolická simulace je proto velice významný nástroj. Umožní uživateli simulovat mnohem větší obvody, než čistě symbolická simulace, a zároveň zachová výhody symbolické simulace. Je vidět, že metoda DMMUN má značně lepší výsledky než metoda tableau.

- Semisymbolická simulace obvodů, které mají 49 prvků s numerickou hodnotou a jeden se symbolickou hodnotou, je metodou DMMUN 5× rychlejší než metodou tableau.

7.4 Porovnání výkonu mého simulátoru a Lcapy



Obrázek 7.6: Porovnání statistické simulace Lcapy a SymCircPy

Simulaci ze sekce 7.3 jsem podle stejných parametrů provedl i pro simulátor Lcapy. Její výsledky jsem dal do jednoho grafu s výsledky simulace metodou DMMUN. Na obrázku 7.6 je vidět, že můj simulátor dosahuje lepších časových výsledků. V tabulce 7.3 jsem uvedl maximální simulační časy obou simulátorů v závislosti na počtu prvků. Zároveň jsem vypočetl podíl výsledků Lcapy a mojí implementace, čímž jsem zjistil, kolikrát je můj simulátor v dané kategorii rychlejší. Maximum je v tomto případě nejvíc směrodatný údaj, protože náhodně generované resistorové sítě mohou mít často triviální řešení. V tabulce 7.4 jsem uvedl porovnání mediánů simulačních časů v závislosti na počtu prvků. Tabulky jsem vypočetl ze stejných dat, která jsou vynesena

Počet prvků	Lcapy, t_l (s)	Můj simulátor, t_s (s)	$\frac{t_l}{t_s}$
2	0,107	0,012	8,9
3	0,109	0,019	5,7
4	0,112	0,062	1,8
5	0,159	0,106	1,5
6	0,213	0,173	1,2
7	0,549	0,364	1,5
8	2,041	0,680	3
9	14,15	1,445	9,8
10	120,9	19,75	6,1
11	846,5	58,64	14,4

Tabulka 7.3: Maximální čas simulace v závislosti na počtu prvků

Počet prvků	Lcapy, t_l (s)	Můj simulátor, t_s (s)	$\frac{t_l}{t_s}$
7	0,245	0,083	3
8	0,6	0,168	3,6
9	3,097	0,350	8,8
10	17,19	0,691	24,9
11	43,43	1,215	35,7

Tabulka 7.4: Medián času simulace v závislosti na počtu prvků

do grafu 7.6. Z těchto výsledků je vidět, že moje implementace dosahuje při těchto parametrech simulace značně lepších výsledků než Lcapy.

7.5 Shrnutí vlastností mého simulátoru

- Je plně open-source a zdarma. Zdrojový kód je k dispozici na GitHubu [20]. Je k dispozici částečná online dokumentace [21], kterou se snažím pravidelně doplňovat.
- Obvody se zadávají pomocí intuitivního SPICE-like netlistu.
- Umí TF, AC, DC, PZ a tranzientní analýzy. Pro všechny je k dispozici symbolická a semisymbolická varianta.
- Umí simulovat rezistory, kapacitory, induktory, nezávislé zdroje, řízené zdroje a ideální OAMP. Metoda tableau navíc umí induktorové vazby. Pro DMMUN induktorové vazby ještě nejsou implementovány. V budoucnu přibydou linearizované diody a tranzistory. Umí zpracovat subcircuity.
- Uživatel má k dispozici několik funkcí, které může použít k rychlému získání požadovaných výsledků.

- Je napsaný v Pythonu a je proto možné simulátor spustit přímo v příkazové řádce. Balíček lze snadno nainstalovat pomocí příkazového řádku

```
pip install symcirc
```

- Je závislý jen na SymPy a je proto velice kompaktní. Není třeba instalovat mnoho balíčků, aby fungoval.
- Protože je naprogramovaný v Pythonu, lze výsledky simulace snadno vizualizovat pomocí některé z veřejně dostupných grafických knihoven (např. matplotlib). Díky dobré kompatibilitě balíčku numpy a sympy může uživatel snadno dosadit za symbolické hodnoty a převést výsledky do numerické formy (pro pohodlné vynesení do grafů).
- Projekt je objektově orientovaný a je proto dobře čitelný jeho kód. Simulace daného typu (DC, AC...) proběhne jen jednou a všechny její výsledky se uloží v objektu pro další manipulaci.
- Umožňuje snadno vizualizovat obvodovou matici použitou k simulaci, tato funkce je užitečná pro pochopení metod simulace a generování příkladů.
- V tuto chvíli je ve vývoji SC analýza (spínané kapacitory). Tuto část vyvíjí můj kolega Filip Špimr.

Kapitola 8

Závěr

V této práci jsem shrnul metody a nástroje, které jsou vhodné k realizaci symbolického simulátoru elektrických obvodů. V teoretické části jsem sepsal přehled metod použitelných k symbolické simulaci obvodů. V praktické části jsem z nich vybral tableau metodu a dvougrafovou modifikovanou metodu uzlových napětí (DMMUN), které jsem naprogramoval ve svém symbolickém simulátoru lineárních obvodů - SymCircPy. Obě metody jsou implementovány na základě publikace [14]. Obě metody jsem v kapitole 7 porovnal nejprve pomocí TF simulace na vybraných obvodech v sekci 7.2. Druhé srovnání jsem provedl pomocí DC simulace na náhodně generovaných rezistorových obvodech. Nejprve symbolicky a poté semisymbolicky v sekci 7.3. Pro symbolické simulace na malých obvodech (méně než 8 prvků) je výhodnější tableau metoda a na větších je výhodnější DMMUN metoda. Pro obvody obsahující operační zesilovače je nejvýhodnější metoda DMMUN. Ze statistických simulací dále vyplývá, že čas, který simulace trvá, je závislý převážně na počtu proměnných použitých v obvodu. Maximum prvků, pro které simulátor spolehlivě dojde k výsledku, je pro metodu tableau 11 prvků a pro metodu DMMUN 12 prvků. Svůj simulátor jsem stejnou statistickou simulací porovnal se simulátorem Lcapy. Můj simulátor poskytuje rychlejší simulaci než balíček Lcapy, jak jsem ukázal v tabulkách 7.3, 7.4 a grafu 7.6. Pro 7 proměnných je maximum simulačních časů v mém simulátoru 1,5× menší a pro 11 proměnných je 14× menší. Všechny simulace jsem udělal v souladu se simulací, kterou provedl autor balíčku Lcapy ve své práci [11] a ke generování obvodů jsem použil přímo generátor obvodů z balíčku Lcapy. Výsledky mojí simulace jsou díky tomu snadno ověřitelné a transparentní.

Výsledek mojí práce, SymCircPy, je open-source softwarový balíček napsaný v jazyce Python. Je závislý jen na knihovně SymPy. SymCircPy je symbolický simulátor lineárních obvodů, který úspěšně zvládá symbolickou a semisymbolickou analýzu typů TF, DC, AC, Tran a PZ.

Literatura

[1] BUKOVSKÝ, Ondřej. *Analýzy v grafickém editoru elektrotechnických obvodů*. Diplomová práce. Praha: ČVUT, FEL, 2021.

[2] *List of computer algebra systems*. Online. WIKIPEDIA. Wikipedia. 2023, aktualizováno 15.12.2023. Dostupné z: `https://en.wikipedia.org/wiki/List_of_computer_algebra_systems`. [cit. 2024-01-05].

[3] *Mathics*. Online. 2022. Dostupné z: `https://mathics.org`. [cit. 2024-01-05].

[4] *SageMath*. Online. Dostupné z: `https://www.sagemath.org`. [cit. 2024-01-05].

[5] PEETERS, Kasper. *Cadabra*. Online. 2023. Dostupné z: `https://cadabra.science`. [cit. 2024-01-05].

[6] MEURER, Aaron; SMITH, Christopher P.; PAPROCKI, Mateusz; ČERTÍK, Ondřej; KIRPICHEV, Sergey B. et al. *SymPy: symbolic computing in Python*. Online. PeerJ Computer Science. S. 27. ISSN 2376-5992. Dostupné z: `https://doi.org/https://doi.org/10.7717/peerj-cs.103.[cit.2022-12-08]`.

[7] FriCAS team. *FriCAS 1.3.9—an advanced computer algebra system*. Online. Available at `http://fricas.github.io`

[8] *Maxima*. Online. 2022. Dostupné z: `https://maxima.sourceforge.io`. [cit. 2024-01-05].

[9] VAN ROSSUM, Guido a DRAKE, Fred L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

[10] STEELE, Guy L. *Common LISP: the language*. 2nd ed. Newton, MA: Digital Press, 1990. ISBN 978-1-55558-041-4.

[11] HAYES, Michael. *Lcapy: symbolic linear circuit analysis with Python*. Online. PeerJ Computer Science. 2022. ISSN 2376-5992. Dostupné z: `https://doi.org/10.7717/peerj-cs.875`. [cit. 2023-10-20].

[12] *Ahkab*. Online. Apr 17, 2017. Dostupné z: `https://ahkab.readthedocs.io/en/latest/`. [cit. 2023-12-08].

[13] HOSPODKA J., BIČÁK J.: *PraCAn - Maple Package for Symbolic Circuit Analysis*, Digital Technologies 2008. Žilina: Žilinská universita, Elektrotechnická fakulta, 2008. ISBN 978-80-8070-953-2.

[14] VLACH, Jiří a SINGHAL, Kishore. *Computer methods for circuit analysis and design.* 2nd ed. Boston: Kluwer Academic Publishers, c1994. ISBN 04-420-1194-6.

[15] PAULŮ, Filip. *Grafický editor elektronických obvodů*. Bakalářská práce, vedoucí Jiří Hospodka. Praha: ČVUT, FEL, katedra mikroelektroniky., 2013. ISBN 978-80-261-0641-8. Dostupné také z: `https://dspace.cvut.cz/handle/10467/18505`.

[16] MAPLESOFT, A DIVISION OF WATERLOO MAPLE INC. *Maple*. Online. 2019. Dostupné z: `https://hadoop.apache.org`. [cit. 2023-12-30].

[17] MICHIGAN STATE UNIVERSITY. *Module 4: General Formulation of Electric Circuit Theory*. Online. Michigan State University. Dostupné z: `https://www.egr.msu.edu/emrg/sites/default/files/content/module4$_$general$_$formulation.pdf`. [cit. 2023-12-30].

[18] VLADIMIRESCU, Andrei. *The SPICE Book*. John Wiley & Sons, January 1994. ISBN 978-0-471-60926-1.

[19] HOSPODKA, Jiří. *Algoritmizace analýz elektrických obvodů*. Online. 2021. Dostupné z: `https://hippo.feld.cvut.cz/vyuka/soubory/mmun.pdf`. [cit. 2024-01-05].

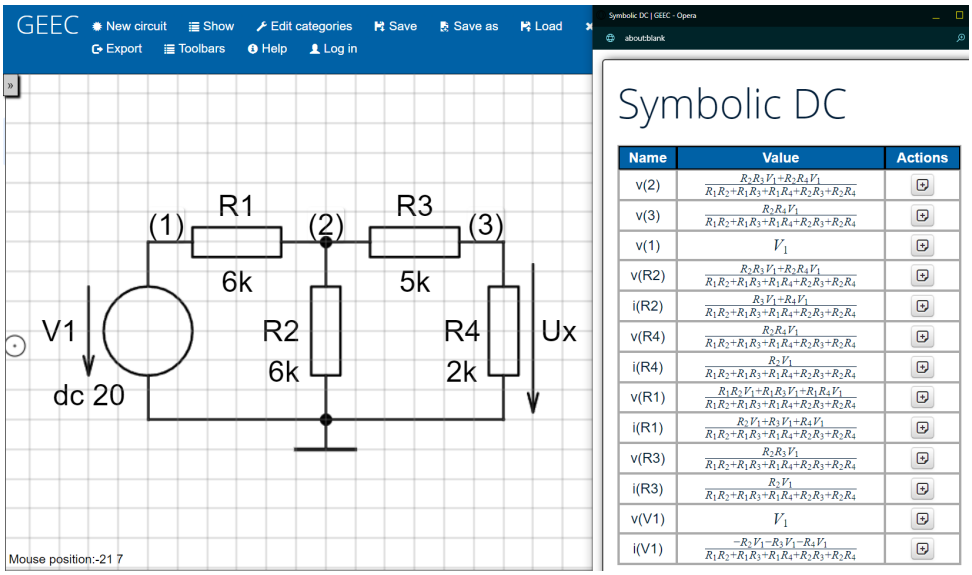
[20] VAŠEK, Matyáš. *SymCircPy Repozitář*. Online. 2023. Dostupné z: `https://github.com/MatyasVasek/SymCirc.[cit.2024-01-05]`.

[21] VAŠEK, Matyáš. *SymCircPy Dokumentace*. Online. 2022. Dostupné z: `https://matyasvasek.github.io/SymCirc/html/index.html.[cit.2024-01-05]`.

Přílohy

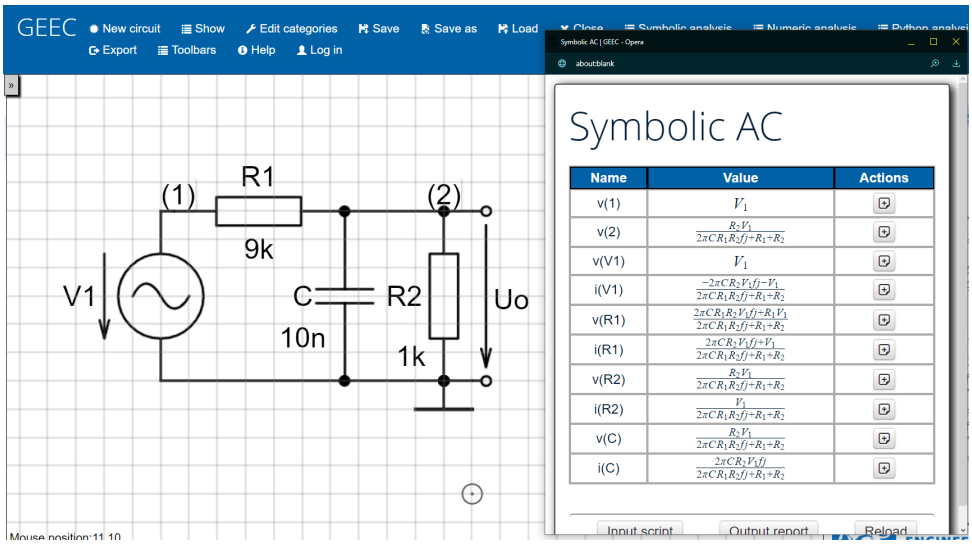
Příloha A

Screenshoty výsledků simulace v simulátoru GEEC vypočítané pomocí mého simulátoru

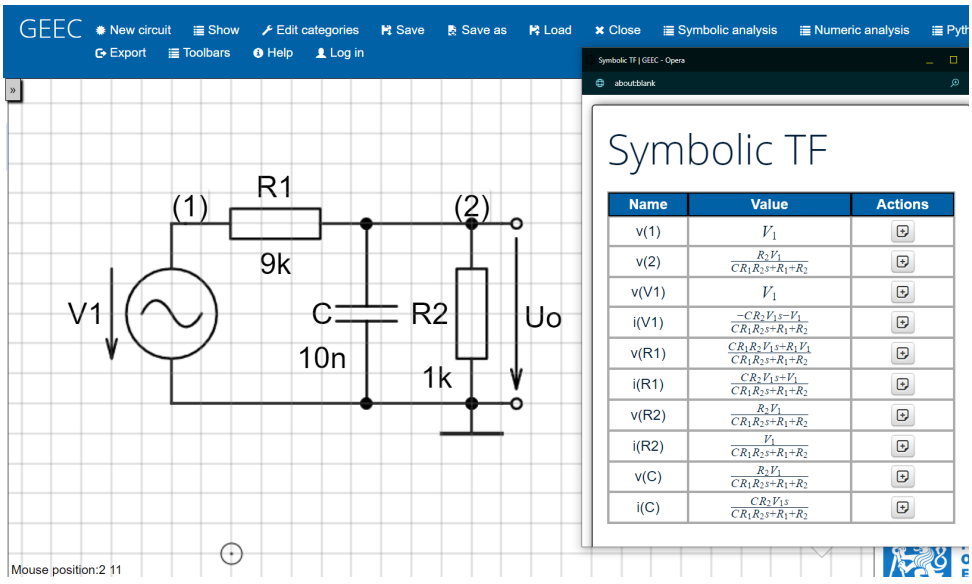


Name	Value	Actions
v(2)	$\frac{R_2R_3V_1+R_2R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
v(3)	$\frac{R_2R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
v(1)	V_1	
v(R2)	$\frac{R_2R_3V_1+R_2R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
i(R2)	$\frac{R_3V_1+R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
v(R4)	$\frac{R_2R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
i(R4)	$\frac{R_2V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
v(R1)	$\frac{R_1R_2V_1+R_1R_3V_1+R_1R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
i(R1)	$\frac{R_2V_1+R_3V_1+R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
v(R3)	$\frac{R_2R_3V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
i(R3)	$\frac{R_3V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	
v(V1)	V_1	
i(V1)	$\frac{-R_2V_1-R_3V_1-R_4V_1}{R_1R_2+R_1R_3+R_1R_4+R_2R_3+R_2R_4}$	

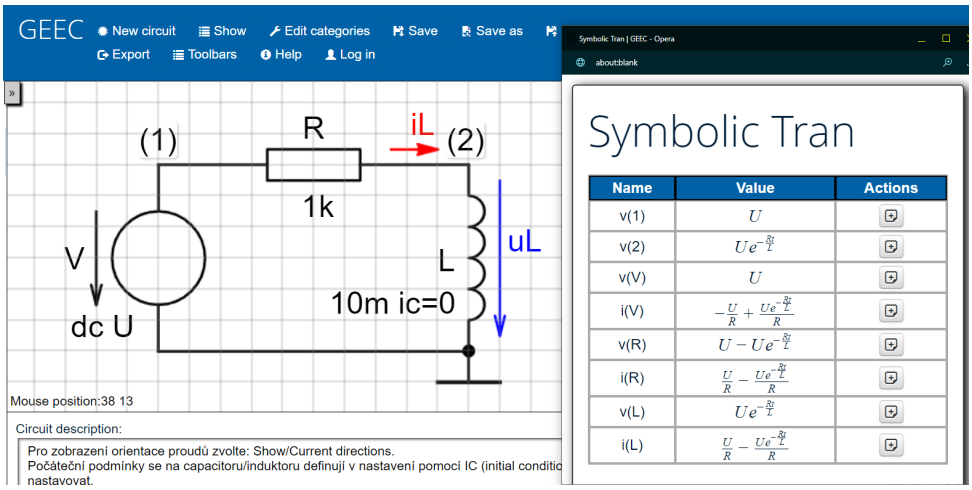
Obrázek A.1: Příklad DC analýzy v simulátoru GEEC pomocí SymCirc



Obrázek A.2: Příklad AC analýzy v simulátoru GEEC pomocí SymCirc



Obrázek A.3: Příklad TF analýzy v simulátoru GEEC pomocí SymCirc



Obrázek A.4: Příklad tranzientní analýzy v simulátoru GEEC pomocí SymCirc