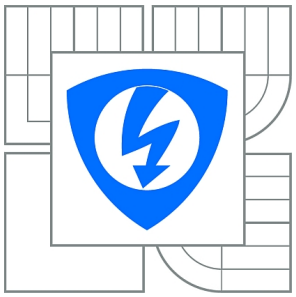
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF MICROELECTRONICS

POČÍTAČOVÁ ANALÝZA SPÍNANÝCH OBVODŮ V KMITOČTOVÉ OBLASTI

FREQUENCY DOMAIN COMPUTER ANALYSIS OF SWITCHED CIRCUITS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

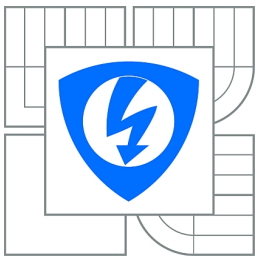
AUTOR PRÁCE — Bc. VLADISLAV PECH

AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. DALIBOR BIOLEK, CSc.

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav mikroelektroniky

Diplomová práce

magisterský navazující studijní obor
Mikroelektronika

Student:	Bc. Vladislav Pech	***ID:***	77748
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Počítačová analýza spínaných obvodů v kmitočtové oblasti

POKYNY PRO VYPRACOVÁNÍ:

Proveďte teorerický rozbor metod algoritmické počítačové analýzy obvodů se spínanými kapacitory s uvažováním lineárních reálných vlivů. Zaměřte se na metodu, kterou následně využijete k tvorbě vlastního programu.
Vytvořte program pro analýzu reálných spínaných obvodů s periodickým externím spínáním v C++ nebo podobném jazyku. Program by měl číst data o simulační úloze ze speciálního netlistu, jehož syntaxe by měla vycházet ze syntaxe SPICE. Výstupy programu budou přenosové funkce v semisymbolickém tvaru v rovině z a kmitočtové charakteristiky.

DOPORUČENÁ LITERATURA:

BIOLEK, D. Počítačová analýza reálných spínaných obvodů. Dílčí výzkumná zpráva úkolu "Syntetické prvky vyššího řádu" Grantové agentury ČR, č. 102/93/2079, ÚTEL FEI Brno, 1994.
BIOLEK, D. S-Z Semi-Symbolic Simulation Of Switched Networks. ISCAS'97 Hongkong, Vol.2, pp.1748-1751.
BIOLEK, D. Modeling of Periodically Switched Networks by Mixed s-z Description. IEEE Trans. on CAS-I, Vol. 44, No. 8, August 1997, pp. 750-758.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: prof. Ing. Dalibor Biolek, CSc.

prof. Ing. Vladislav Musil, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt:

Projekt se zabývá problematikou počítačové analýzy obvodů s externím spínáním. Nejprve popisuje tvorbu vstupu programu CIRNAM – popis obvodu, modifikovaný netlist. Práce také popisuje teorii využitou pro řešení analyzovaných obvodů. V druhé části je podrobně popsán vytvořený program CIRNAM. Dále na příkladech je ukázána práce v programu a možnosti výstupu programu – vykreslení kmitočtových charakteristik v programu CIRNAM nebo export vypočítaných dat do programu MATLAB.

Jsou zde popsány také zdrojové kódy programu CIRNAM pro prvotní orientaci případného programátora, který by rozšiřoval možnosti tohoto programu.

Abstract:

This project deals with the computer analysis of circuits with external switching. At the first of all there is a description of the creation of the entry of the program CIRNAM – description of the circuit, modified netlist. The work also discusses the theory used for solutions of analysed circuits. In the other part there is a description of the program CIRNAM in full details. In the next parts of this project there is the illustration of work in the program, which is shown on the examples with short discussion of output options – he rendering of the frequency characteristics in the program CIRNAM or the export of calculated data to MATLAB.

There are also described the source code of CIRNAM for the initial orientation of the programmer, which would extend the possibilities of this program.

Klíčová slova:

Spínaný kapacitor, analýza, netlist, mfile, CIRNAM, SPICE, MATLAB, C++.

Keywords:

Switched capacitor, analysis, netlist, mfile, CIRNAM, SPICE, MATLAB, C++.

Bibliografická citace díla:

PECH, V. *Počítačová analýza spínaných obvodů v kmitočtové oblasti – diplomová práce.* Brno, 2011. 64 s. Vedoucí diplomové práce prof. Ing. Dalibor Biolek, CSc. FEKT VUT v Brně

Prohlášení autora o původnosti díla:

Prohlašuji, že jsem tuto vysokoškolskou kvalifikační práci vypracoval samostatně pod vedením vedoucího diplomové práce, s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 20. 5. 2011

…….

Poděkování:

Děkuji vedoucímu diplomové práce prof. Ing. Daliboru Biolkovi, CSc. za metodickou, pedagogickou a odbornou pomoc při zpracování zadání dílčích prací a diplomové práce a další pomoc a rady při tvorbě vlastního programu, který je výsledkem této práce.

V Brně dne 20. 5. 2011

…….

OBSAH

1 Úvod

Diplomová práce shrnuje a završuje znalosti nabyté při tvorbě bakalářské práce a semestrálních projektů. Jejím cílem je vytvořit program pro analýzu filtrů se spínanými kapacitory. Ačkoliv podobné simulační programy v dnešní době existují, lze jimi řešit pouze idealizované obvody obsahující ideální spínače s nulovým odporem v sepnutém stavu, kapacitory a ideální operační zesilovače. Výstupem těchto programů je obvykle kmitočtová charakteristika obvodů, která se počítá z jeho systémové funkce v oblasti *z*.

U nás byly například vyvinuty programy COCOSC a SPASO pro symbolickou analýzu, SCSK pro semisymbolickou a kmitočtovou analýzu a program SCC pro časovou analýzu. Objevily se ale i programy, např. dánský SCANET, pro analýzu reálných spínaných obvodů, ale ty při kmitočtové analýze berou v úvahu pouze diskrétní složku signálů a zanedbávají analogový charakter přechodných dějů způsobených spínáním. Dále byly na zakázku pro velké zahraniční firmy vyvinuty speciální simulační programy v Kanadě, které pracují na výkonných počítačových stanicích, čímž jsou pro běžné účely nedostupné.

Většina programů neposkytuje výstup ve formě klasických kmitočtových charakteristik, protože jejich definice v případě reálných spínaných obvodů vlastně nebyla oficiálně zavedena. Z obdobných důvodů není možný výstup ve formě nulových bodů a pólů přenosových funkcí, neboť obecný spínaný obvod nelze popsat klasickými přenosovými funkcemi typu racionálních lomených funkcí operátoru *s* nebo *z*. [3]

Z výše uvedených důvodů pan prof. Ing. Dalibor Biolek vytvořil vlastní program, který byl za jeho pomoci modernizován a převáděn do jazyku C++ a programu MATLAB.

V bakalářské práci byla popsána základní teorie počítačové analýzy reálných spínaných obvodů, převzatá z [2], dle které byl vytvářen program CIRNAM. Vstupem tohoto programu je tzv. rozšířený netlist spínaného obvodu a výstupem jsou vytvořené obvodové matice pro jednotlivé fáze. Tento program měl zahrnovat popsané výpočty vedoucí k matematickému modelu tzv. zobecněným přenosovým funkcím $\mathbf{K}_{11}$, $\mathbf{K}_{12}$, $\mathbf{K}_{21}$ a $\mathbf{K}_{22}$ pro spínaný obvod, který je buzen jediným signálem. Na základě tohoto matematického modelu měl pak vykreslit frekvenční charakteristiku analyzovaného obvodu.

Jelikož ani v průběhu semestrálních projektů na magisterském studiu se nepodařilo odladit, některé části výpočtu převzaté z programu pana profesora Biolka, byl program CIRNAM tvořený na teoretické základně popsané v bakalářské práci modifikován dle nové teorie, vytvořené panem profesorem.

V této práci bude popsáno řešení obvodů se spínanými kapacitory, využité při tvorbě programu CIRNAM. Vstupní data budou jednoduché netlisty obsahující modely vybraných obvodových prvků podle pokynů vedoucího práce. Dále by se v projektu měla navrhnout nová syntaxe pro zápis behaviorálních modelů operačních zesilovačů a spínačů, které by byly součástí programu pro analýzu obvodů s periodicky řízenými analogovými spínači. Po vytvoření třídících funkcí a syntaxí by se dále vytvořili funkce, které by zpracovali zjištěné informace a vypočítaly by se hodnoty pro vykreslení kmitočtové charakteristiky dle zadání vstupního a výstupního uzlu a typu buzení obvodu. Ty by dále byly zpřístupněny ke zpracování v programu MATLAB, především pro jejich grafické zpracovaní v podobě grafů.

2 Syntaxe použité v netlistu

V této části je zpracována problematika syntaxí zápisu součástek v rozšířeném netlistu. Jelikož je program koncipován v širším rozsahu, než ve kterém je v současné době naprogramován, jsou zde uvedeny i zápisy součástek, které program rozpozná, ale dále je pro analýzu obvodu nevyužívá. Některé syntaxe jsou shodné se zápisem v jazyce SPICE a také každý řádek je možné okomentovat tak, že před komentář napíšeme středník.

2.1 Zdroje – U

Jedná se o obyčejný napájecí zdroj, který má jeden vstup a jeden výstup a bývá často připojen mezi zem a vstupní svorku obvodu.

Syntaxe:

`<název zdroje> <vstup> <výstup> <hodnota>`

Popis syntaxe:

`<název zdroje>`	zahrnuje jméno zdroje, začínající písmenem *U*
`<vstup>`	označení vstupní svorky zdroje
`<výstup>`	označení výstupní svorky zdroje
`<hodnota>`	číselná hodnota zdroje, za níž může následovat inženýrská notace

2.2 Zdroje řízené napětím – E a G

Zdroje *E* a *G* mají stejnou syntaxi. Písmeno *E* je označení pro zdroj napětí řízený napětím a písmeno *G* značí zdroj proudu řízený také napětím.

Syntaxe:

`<název zdroje> <výstup+> <výstup-> <vstup+> <vstup-> <přenos>`

Popis syntaxe:

`<název zdroje>`	zahrnuje jméno zdroje, začínající buď písmenem *E* nebo *G*, podle typu zdroje
`<výstup+>`	označení kladné výstupní svorky zdroje
`<výstup->`	označení záporné výstupní svorky zdroje
`<vstup+>`	označení kladné vstupní svorky zdroje
`<vstup->`	označení záporné vstupní svorky zdroje

`<přenos>`	číselná hodnota udávající velikost přenosu

2.3 Zdroje řízené proudem – F a H

Zdroje *F* a *H* mají stejnou syntaxi. Písmeno *F* je označení pro zdroj proudu řízený proudem a písmeno *H* značí zdroj napětí řízený také proudem.

Syntaxe:

```
<název zdroje> <výstup+> <výstup-> <název řídícího zdroje> <přenos>
```

Popis syntaxe:

`<název zdroje>`	zahrnuje jméno zdroje, začínající buď písmenem *F* nebo *H*, podle typu zdroje, který definujeme
`<výstup+>`	označení kladné výstupní svorky zdroje
`<výstup->`	označení záporné výstupní svorky zdroje
`<název řídícího zdroje>`	název řídícího zdroje začínající vždy písmenem *V*
`<přenos>`	číselná hodnota udávající velikost přenosu

2.4 Pasivní součástky – R a C

Následující deklarace odporu (*R*) a kondenzátoru (*C*) jsou stejné, proto je uvádím opět ve stejné podkapitole. Syntaxe je vlastně stejná jako u zdroje *U*, pouze se liší inicializačním písmenem.

Syntaxe:

```
<název součástky> <vstup> <výstup> <hodnota>
```

Popis syntaxe:

`<název součástky>`	název součástky, který musí začínat jedním z písmen *R* nebo *C*, záleží na tom, jakou součástku právě deklarujeme
`<vstup>`	označení vstupní svorky součástky
`<výstup>`	označení výstupní svorky součástky
`<hodnota>`	číselná hodnota součástky, za níž může následovat inženýrská notace

2.5 Pasivní součástka – L

Následující deklarace cívky (*L*) je odlišná od deklarace odporů a kondenzátorů, proto je uváděna samostatně v podkapitole. Syntaxe obsahuje definici vnitřního odporu cívky.

Syntaxe:

<název součástky> <vstup> <výstup> <hodnota L_s> <hodnota R_s>

Popis syntaxe:

<název součástky>	název součástky, který musí začínat písmenem *L*
<vstup>	označení vstupní svorky součástky
<výstup>	označení výstupní svorky součástky
<hodnota L_s>	číselná hodnota indukčnosti součástky, za níž může následovat inženýrská notace
<hodnota R_s>	číselná hodnota vnitřního odporu součástky, za níž může následovat inženýrská notace

2.6 Vzájemná indukčnost – M

Deklarace vzájemné indukčnosti (*M*) obsahuje deklaraci dvou cívek a hodnotu součinitele vazby mezi těmito cívkami.

Syntaxe:

<název> <a> <b> <c> <d> <L_1> <L_2> <k>

Popis syntaxe:

<název>	název vzájemné indukčnosti, který musí začínat písmenem M
<a>	označeni vstupní svorky první cívky
<b>	označeni výstupní svorky první cívky
<c>	označeni vstupní svorky druhé cívky
<d>	označeni výstupní svorky druhé cívky
<L_1>	číselná hodnota indukčnosti první cívky
<L_2>	číselná hodnota indukčnosti druhé cívky
<k>	číselná hodnota součinitele vazby cívek, která může nabývat hodnot 0-1

2.7 Operační zesilovače – OA

Operační zesilovače (*OA*) jsou velmi obsáhlým tématem a jejich syntaxe se liší v závislosti na typu. V současné době program zpracovává 3 typy operačních zesilovačů. Název syntaxe začínající „*IOA*“ označuje ideální diferenční operační zesilovač (IDOZ), syntaxe začínající „*OAI*“ značí zesilovač s jednotkovým zesílením (BUFFER) a syntaxe začínající „*OA1P*“ značí operační zesilovač s jedním kmitočtem lomu.

Syntaxe:

`<název IOA> <vstup+> <vstup-> <vystup>`
`<název OAI> <vstup> <výstup>`
`<název OA1P> <vstup+> <vstup-> <vystup>` $<A_0>$ $<f_T>$ $<R_0>$

Popis syntaxe:

`<název IOA>`	název musí začínat „*IOA*“ – ideální operační zesilovač
`<název OAI>`	název musí začínat „*OAI*“ – zesilovač s jednotkovým zesílením
`<název OAI1P>`	název musí začínat „*OAI*1*P*“ – operační zesilovač s jedním kmitočtem lomu
`<vstup+>`	označení kladné vstupní (neinvertující) svorky
`<vstup->`	označení záporné vstupní (invertující) svorky
`<výstup>`	označení výstupní svorky
$<A_0>$	stejnosměrné zesílení, za níž může následovat inženýrská notace
$<f_T>$	číselná hodnota tranzitního kmitočtu, za níž může následovat inženýrská notace
$<R_0>$	číselná hodnota výstupního odporu, za níž může následovat inženýrská notace

2.8 Spínače – S

Obecná syntaxe reálných spínačů, které by v budoucnu mohla být modifikována na dva typy spínačů: Spínač řízený proudem a spínač řízený napětím, které jsou v problematice behaviorálního modelování velmi důležité.

Syntaxe:

```
S<název> <vstup> <výstup> <on/off> <hodnota>
```

Popis syntaxe:

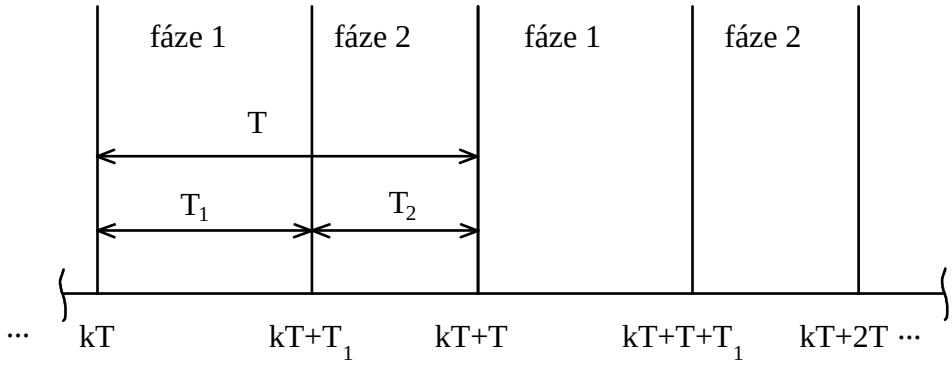
`<název>`	název spínače, který musí začínat písmenem „*S*“
`<vstup>`	označení vstupní svorky
`<výstup>`	označení výstupní svorky
`<on/off>`	stav spínače v první fázi – sepnutý (on) nebo rozepnutý (off), ve 2. fázi je spínač automaticky uvažován v druhém stavu
`<hodnota>`	číselná hodnota odporu v sepnutém stavu, pokud je hodnota nulová, pak se nemusí zadávat

3 Osnova řešení zpracování dat

Další úvahy řešení se týkají spínaných obvodů s diagramem činnosti podle obr. 1. Uvažují se dvě fáze, každá s rozdílnou dobou trvaní. Počítačovou simulaci těchto obvodů rozdělíme do následujících etap [2]:

- Sestavení obvodových rovnic a „supermatice".

- Analýza kmitočtových charakteristik na základě „supermatice“.



Obr. 1: Spínací diagram.

4 Sestavení obvodových rovnic

4.1 Obecná koncepce

Problematika analýzy obvodů se spínanými kapacitory je v tomto projektu řešena sestavováním obvodových rovnic na základě modifikované metody uzlových napětí (MMUN) a jejich následnému zápisu do „supermatice“. Vzhledem k tomu, že pro každý uzel je ve výsledné „supermatici“ jeden řádek, je vhodné i v dnešní době výkonných počítačů minimalizovat jejich počet. Řešení je následující [2]:

- Pro každou spínací fázi se nalezne soubor nezávislých branových veličin:

$$\begin{aligned} \mathbf{v}_1 &= \left[v_1^1, v_1^2, .., v_1^{n_1}\right]^T \text{... fáze 1,} \\ \mathbf{v}_2 &= \left[v_2^1, v_2^2, .., v_2^{n_2}\right]^T \text{... fáze 2,} \end{aligned} \tag{1}$$

kde n_1, resp. n_2 je počet nezávislých veličin ve fázi 1, resp. 2.

- Pro branové veličiny lze sestavit v každé fázi soustavu diferenciálních rovnic modifikované metody uzlových napětí:

$$\begin{aligned} &\text{fáze}\,1 : t \in \langle kT, kT + T_1) : \mathbf{D}_1 w_1(t) = \mathbf{G}_1^M \mathbf{v}_1(t) + \mathbf{C}_1^M \frac{d}{dt}\mathbf{v}_1(t), \\ &\text{fáze}\,2 : t \in \langle kT + T_1, kT + T) : \mathbf{D}_2 w_2(t) = \mathbf{G}_2^M \mathbf{v}_2(t) + \mathbf{C}_2^M \frac{d}{dt}\mathbf{v}_2(t), \end{aligned} \tag{2}$$

kde $\boldsymbol{G}_1{}^{\mathrm{M}}$, resp. $\boldsymbol{G}_2{}^{\mathrm{M}}$ je modifikovaná matice vodivostí rozměru (n_1 x n_1), resp. (n_2 x n_2),

$\boldsymbol{C}_1{}^{\mathrm{M}}$, resp. $\boldsymbol{C}_1{}^{\mathrm{M}}$ je modifikovaná matice reaktancí rozměru (n_1 x n_1), resp. (n_2 x n_2),

$\boldsymbol{D}_1$, resp. $\boldsymbol{D}_2$ jsou jisté incidenční vektory rozměru (n_1 x 1), resp. (n_2 x 1),

w_1, resp.w_2 jsou budicí signály ve fázi 1, resp. 2.

Rovnice (2) znamenají rovnosti zobecněných proudů (v případě klasické metody uzlových napětí by se jednalo o rovnost klasických proudů). Na levé straně jsou proudy tekoucí do uzlů obvodu z vnějších zdrojů a na pravé straně jsou proudy, které jsou vyvolány vnějšími proudy, tekoucí z uzlů dovnitř do obvodu.

Proudové rovnice (2) lze převést na nábojové rovnice integrací rovnic v časových intervalech, uvedených v (2):

$$\begin{aligned} &\text{fáze}1:\ \mathbf{D}_1 q_1(kT+T_1^-) = \mathbf{G}_1^M \boldsymbol{\varphi}_1(kT+T_1^-) + \mathbf{C}_1^M\left[\mathbf{v}_1(kT+T_1^-) - \mathbf{v}_1(kT^+)\right], \\ &\text{fáze}\,2:\ \mathbf{D}_2 q_2(kT+T^-) = \mathbf{G}_2^M \boldsymbol{\varphi}_2(kT+T^-) + \mathbf{C}_2^M\left[\mathbf{v}_2(kT+T^-) - \mathbf{v}_2(kT+T_1^+)\right], \end{aligned} \tag{3}$$

kde horní indexy + a – znamenají limity zprava, resp. zleva, a veličiny typu q a $\boldsymbol{\varphi}$ jsou definovány časovými integrály:

$$\begin{aligned} q_1(kT+T_1^-) &= \int_{kT^+}^{kT+T_1^-} w_1(\alpha)d\alpha,\ q_2(kT+T^-) = \int_{kT+T_1^+}^{kT+T^-} w_2(\alpha)d\alpha, \\ \boldsymbol{\varphi}_1(kT+T_1^-) &= \int_{kT^+}^{kT+T_1^-} \mathbf{v}_1(\alpha)d\alpha,\ \boldsymbol{\varphi}_2(kT+T^-) = \int_{kT+T_1^+}^{kT+T^-} \mathbf{v}_2(\alpha)d\alpha \end{aligned} \tag{4}$$

Veličina q_i, i = 1 nebo 2, je zobecněný náboj, který přiteče během trvání spínací fáze č. i do budicího uzlu. Veličina $\boldsymbol{\varphi}_i$, i = 1 nebo 2, je vektor zobecněných toků, resp. časových integrálů zobecněných uzlových napětí, přičemž integrace je realizována v rámci spínací fáze č. i.

Rovnice (3) však ještě nejsou vhodné pro simulaci. Za předpokladu, že jednotlivé spínací fáze na sebe navazují, pak vektor počátečních podmínek těsně po sepnutí spínačů musí být jednoznačně určen vektorem počátečních podmínek těsně před sepnutím. Konkrétně pro lineární spínaný obvod musí platit:

$$\begin{aligned} \mathbf{v}_1\left(kT^+\right) &= \mathbf{S}_{12}\mathbf{v}_2\left(kT^-\right), \\ \mathbf{v}_2\left(kT+T_1^+\right) &= \mathbf{S}_{21}\mathbf{v}_1\left(kT+T_1^-\right), \end{aligned} \tag{5}$$

kde $\boldsymbol{S}_{12}$, resp. $\boldsymbol{S}_{21}$ jsou jisté transformační matice řádu (n_1 x n_2), resp. (n_2 x n_1).

Po dosazení (5) do (3):

$$\begin{aligned} &\text{fáze}1:\ \mathbf{D}_1 q_1(kT+T_1^-) = \mathbf{G}_1^M \boldsymbol{\varphi}_1(kT+T_1^-) + \mathbf{C}_1^M \mathbf{v}_1(kT+T_1^-) - \mathbf{C}_{12}^M \mathbf{v}_2(kT^-), \\ &\text{fáze}\,2:\ \mathbf{D}_2 q_2(kT+T^-) = \mathbf{G}_2^M \boldsymbol{\varphi}_2(kT+T^-) + \mathbf{C}_2^M \mathbf{v}_2(kT+T^-) - \mathbf{C}_{21}^M \mathbf{v}_1(kT+T_1^-), \end{aligned} \tag{6}$$

kde

$$\mathbf{C}_{12}^M = \mathbf{C}_1^M \mathbf{S}_{12},\ \mathbf{C}_{21}^M = \mathbf{C}_2^M \mathbf{S}_{21}. \tag{7}$$

Poslední úpravou je dosaženo toho, že všechny obvodové veličiny vystupují v rovnicích ve formě vzorků v okamžicích ukončení spínacích fází ve smyslu limit zleva. Rovnice jsou

tak připraveny pro kmitočtovou analýzu ve smyslu metody zobecněných přenosových funkcí [2].

Kmitočtová analýza probíhá ve dvou krocích: Nejprve jsou vzorky obvodových veličin v (6) „proloženy“ tzv. ekvivalentní signály. Pak je na tyto signály aplikován klasický operátorový počet s následnou substitucí $s = j\omega$.

Pro ekvivalentní signály (s horním indexem *e*) platí upravené rovnice (6):

$$\begin{aligned} &\text{fáze}\,1: \ \mathbf{D}_1 q_1^e(t) = \mathbf{G}_1^M \boldsymbol{\varphi}_1^e(t) + \mathbf{C}_1^M \mathbf{v}_1^e(t) - \mathbf{C}_{12}^M \mathbf{v}_2^e(t - T_1), \\ &\text{fáze}\,2: \mathbf{D}_2 q_2^e(t) = \mathbf{G}_2^M \boldsymbol{\varphi}_2^e(t) + \mathbf{C}_2^M \mathbf{v}_2^e(t) - \mathbf{C}_{21}^M \mathbf{v}_1^e(t - T_2). \end{aligned} \tag{8}$$

Po aplikaci Laplaceovy transformace:

$$\begin{aligned} &\text{fáze}\,1: \ \mathbf{D}_1 Q_1^e(s) = \mathbf{G}_1^M \boldsymbol{\Phi}_1^e(s) + \mathbf{C}_1^M \mathbf{V}_1^e(s) - \mathbf{C}_{12}^M \mathbf{V}_2^e(s)\, e^{-sT_1}, \\ &\text{fáze}\,2: \mathbf{D}_2 Q_2^e(s) = \mathbf{G}_2^M \boldsymbol{\Phi}_2^e(s) + \mathbf{C}_2^M \mathbf{V}_2^e(s) - \mathbf{C}_{21}^M \mathbf{V}_1^e(s)\, e^{-sT_2}. \end{aligned} \tag{9}$$

Laplaceovy obrazy veličin jsou od zápisu těchto veličin v čase odlišeny velkými písmeny.

V rovnicích (9) figurují Laplaceovy obrazy jak napětí, tak i toků. Z praktických důvodů je vhodné převést toky na napětí.

Z rovnice (4) vyplývá pro ekvivalentní signál k vektoru $\boldsymbol{\varphi}_1$:

$$\boldsymbol{\varphi}_1^e(t) = \int_{t-T_1}^{t} \mathbf{v}_1(\alpha)d\alpha = \int_{-\infty}^{t} \mathbf{v}_1(\alpha)d\alpha - \int_{-\infty}^{t-T_1} \mathbf{v}_1(\alpha)d\alpha = \int_{-\infty}^{t} \mathbf{v}_1(\alpha)d\alpha - \int_{-\infty}^{t} \mathbf{v}_1(\alpha - T_1)d\alpha\,. \tag{10}$$

Pak jeho Laplaceův obraz bude:

$$\boldsymbol{\Phi}_1^e(s) = \frac{1 - e^{-sT_1}}{s} \mathbf{V}_1\,. \tag{11}$$

Obdobným způsobem bychom odvodili vztah pro Laplaceův obraz ekvivalentního signálu k vektoru $\boldsymbol{\varphi}_2$:

$$\boldsymbol{\Phi}_2^e(s) = \frac{1 - e^{-sT_2}}{s} \mathbf{V}_2\,. \tag{12}$$

Protože ekvivalentní signály se obecně liší od signálů originálních, nebudou totožné ani jejich Laplaceovy obrazy. Pro účely kmitočtové analýzy je však možné učinit zjednodušující předpoklad shody obou obrazů, pokud bude kmitočet signálu dostatečně nižší než kmitočet spínání f_S, tedy $s = j\omega = j2\pi f$, $f << f_s$. Pak bude platit:

$$\mathbf{v}_i(t) \approx \mathbf{v}_i^e(t) \Rightarrow \mathbf{V}_i(s) \approx \mathbf{V}_i^e(s)\,,\ i = 1, 2. \tag{13}$$

Za tohoto předpokladu pak lze kombinováním rovnic (9), (11) a (12) dospět k výsledku:

$$\begin{aligned} &\text{fáze}\,1:\ \mathbf{D}_1 Q_1^e(s) = [\mathbf{G}_1^M f_1(s) + \mathbf{C}_1^M]\mathbf{V}_1^e(s) - \mathbf{C}_{12}^M \mathbf{V}_2^e(s)\, e^{-sT_1}, \\ &\text{fáze}\,2:\ \mathbf{D}_2 Q_2^e(s) = [\mathbf{G}_2^M f_2(s) + \mathbf{C}_2^M]\mathbf{V}_2^e(s) - \mathbf{C}_{21}^M \mathbf{V}_1^e(s)\, e^{-sT_2}, \end{aligned} \tag{14}$$

kde je pro přehlednost označeno:

$$f_i(s) = \frac{1 - e^{-sT_i}}{s},\ i = 1, 2. \tag{15}$$

Rovnice (14) lze přepsat do maticového tvaru:

$$\begin{bmatrix} \mathbf{D}_1 Q_1^e \\ \mathbf{D}_2 Q_2^e \end{bmatrix} = \begin{bmatrix} \mathbf{G}_1^M f_1 + \mathbf{C}_1^M & -\mathbf{C}_{12}^M\, e^{-sT_1} \\ -\mathbf{C}_{21}^M\, e^{-sT_2} & \mathbf{G}_2^M f_2 + \mathbf{C}_2^M \end{bmatrix} \begin{bmatrix} \mathbf{V}_1^e \\ \mathbf{V}_2^e \end{bmatrix} \tag{16}$$

Rovnice (16) má klíčový význam pro počítačovou analýzu. Lze ji vždy sestavit pro jakýkoliv reálný lineární spínaný obvod bez ohledu na to, zdali je v okamžiku přepnutí do jiné fáze splněna podmínka konzistence počátečních podmínek. V praxi to znamená, že simulátor bude poskytovat správné výsledky i v případě, že spínaný obvod obsahuje části, kde v důsledku ideálních spínačů dochází ke skokovému přebíjení kapacitorů Diracovými impulsy. Jedinou omezující podmínkou je dostatečný odstup pracovní a spínací frekvence tak, aby byla splněna relace (13).

V následující části je popsán algoritmus hledání nezávislých obvodových veličin, dále pak algoritmus sestavování matic a vektorů Y_1^M, D_1, Y_2^M, D_2, C_{12}^M a C_{21}^M. V závěru ukážeme, jak z čtvercové „supermatice“ v rovnici (16) získáme kmitočtové charakteristiky spínaného obvodu.

4.2 Hledání soustavy nezávislých obvodových veličin

Při počítačové analýze je důležitým faktorem počet rovnic popisující daný obvod. Se zvyšujícím se počtem obvodových rovnic se prodlužuje čas potřebný k výpočtu a samozřejmě rostou také nároky na paměť počítače. Klasická modifikovaná metoda uzlových

napětí je navíc charakteristická tím, že k rovnicím 1. Kirchhoffova zákona pro nezávislé uzly přidává další rovnice, které zachycují lineární transformace obvodových veličin. Ty jsou způsobeny přítomností spínačů a neregulárních prvků v obvodu. Celkový počet rovnic tak neúměrně roste a je vhodné je minimalizovat.

Jedna z možností je sestavit obvodové rovnice, v nichž budou vystupovat pouze nezávislé branové veličiny. Pro nalezení souboru těchto nezávislých veličin je užito transformace a redukce proměnných ještě před sestavením obvodových rovnic [1].

4.2.1 Popis algoritmu

Každému uzlu v obvodu přiřadíme čtyři čísla. První dvě čísla označíme jako napěťový a proudový koeficient první spínací fáze, další dvě čísla jsou opět proudový a napěťový koeficient ale tentokrát druhé spínací fáze. Pro lepší orientaci při tvorbě admitančních matic se budou dále tyto koeficienty označovat následovně:

*a** napěťový koeficient v 1. spínací fázi,

*a*** proudový koeficient v 1. spínací fázi,

*b** napěťový koeficient ve 2. spínací fázi,

*b*** proudový koeficient v 2. spínací fázi.

V admitanční matici udává proudový koeficient číslo řádku, který odpovídá rovnici 1. Kirchhoffova zákona. Napěťový koeficient pak značí číslo sloupce této matice, který přísluší právě uzlovému napětí daného uzlu. Dále pro sestavování matic je zvolen referenční uzel s číslem 0, jemuž jsou přiřazeny odpovídající koeficienty 0*, 0**, 0*, 0**.

Pro případ, kdy klasický analogový obvod obsahuje pouze regulární prvky vzhledem k modifikované metodě uzlových napětí, budou oba koeficienty uzlů shodné a budou odpovídat pořadí uzlu. Pokud ale bude uvažován ideální spínač mezi dvěma uzly (s nulovým odporem v sepnutém stavu a nekonečným v rozepnutém stavu) a v jedné z fází bude tento spínač sepnut, pak budou napěťové i proudové koeficienty pro oba uzly shodné. Obecně pak *x* různých uzlů, v tomto případě zkratovaných v dané fázi, je popsáno jedinou rovnicí 1. Kirchhoffova zákona místo původních *x* rovnic. Také může nastat situace, kdy se v obvodu vyskytne uzel, který je v určité fázi izolovaný. Tomu jsou pak přiřazeny stejné koeficienty jako uzlu referenčnímu, čímž je vyloučeno vytváření dalších řádků a sloupců v admitanční matici, které by obsahovali pouze nulové hodnoty.

V obvodu mimo jiné může být také zapojen ideální operační zesilovač. Uzlová napětí tohoto operačního zesilovače jsou pak na jeho diferenčních vstupech shodná a příslušným

uzlům jsou přiřazeny stejné napěťové koeficienty. Proudové koeficienty ale zůstanou obecně různé, neboť oba vstupy lze budit nezávisle různými zdroji proudu. Buzeni do výstupu operačního zesilovače, tj. ideálního zdroje napětí, nemá smysl, proto proudový koeficient výstupního uzlu je vždy nulový. Z uvedených modifikací koeficientů uzlů je zřejmé, že ideální operační zesilovač zmenši o jedničku počet nezávislých uzlových napětí a proudů v každé spínací fázi, čímž je i snížen řád modifikovaných admitančních matic. Naproti tomu u klasické MMUN řad těchto matic roste.

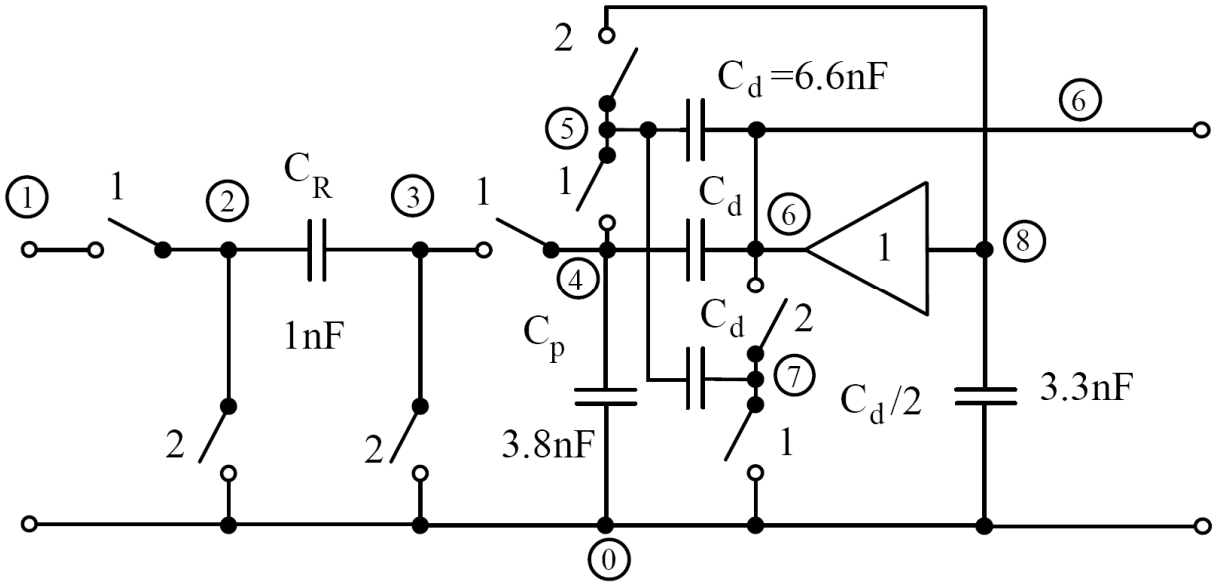
Působení jednotkového zesilovače v obvodu má obdobné důsledky jako zmíněný ideální operační zesilovač. Uzlová napětí na vstupu a výstupu jsou totožná, a proto příslušné uzly mají stejné napěťové koeficienty. Proudový koeficient výstupního uzlu je opět nulový.

Dále je popsán i způsob tvorby maticového popisu pro reálný operační zesilovač. Je uvažován s jeho konečným stejnosměrným zesílením, prvním kmitočtem lomu a nenulovým výstupním odporem. V tomto případě pak dochází k růstu řádu admitančních matic o jedna, stejně jako by tomu bylo u obvodů s vlastními a vzájemnými indukčnostmi, které nejsou v programu pro analýzu začleněny.

Pokud je zajištěno průběžné číslování uzlů v obou fázích, pak nejvyšší číslo koeficientu udává řád admitanční matice v dané fázi, neboť z výše uvedených principů vyplývá, že celkový počet navzájem různých napěťových a proudových koeficientů uzlů je v každé fázi stejný.

4.2.1 Praktická ukázka algoritmu

Zmíněné principy redukce počtu obvodových rovnic jsou blíže objasněny na přikladu spínaného filtru s SC dvojným kapacitorem na obr. 2. Je uvažován ideální jednotkový zesilovač a ideální spínače.



Obr. 2: Dolní propust 2. řádu s dvojným kapacitorem [1].

Z Tab. 1 je možné vysledovat algoritmus použitý v programu pro přiřazování koeficientů ke každému uzlu. Nejprve proběhne průběžné číslování. Dále je provedena transformace spínači. Uzly spojené spínačem se označí stejným koeficientem, při čemž se vybere nižší číslo z pořadových čísel spojených uzlů. Referenční uzel 0 nebo uzel nepřipojený se označí koeficienty 0*, 0** v obou fázích. V této fázi algoritmu jsou napěťové a proudové koeficienty stejné. Další transformace, která proběhne je transformace neregulárními prvky. V tomto praktickém případě se jedná o jednotkový zesilovač, tzv. buffer.

V poslední části algoritmu je provedeno vzestupné přečíslovaní uzlů. Zde je možné přímo vidět podle nejvyšší hodnoty čísla uzlů výsledná velikost admitanční matice. V uvedeném příkladě je rozměr matice v první fázi 3 x 3 a ve druhé fázi 2 x 2. Nezávislé obvodové veličiny jsou pak tyto:

fáze 1: u_1, u_3, u_6, i_1, i_3, i_6,

fáze 2: u_4, u_5, i_4, i_5.

Tab. 1: Princip hledáni soustavy nezávislých obvodových veličin filtru z obr. 2.

Fáze	Uzel	Průběžné číslování		Transformace				Vzestupné přečíslovaní	
				Spínači		IZN			
		*	**	*	**	*	**	*	**
1	1	1	1	1	1	1	1	1	1
	2	2	2	1	1	1	1	1	1
	3	3	3	3	3	3	3	2	2
	4	4	4	3	3	3	3	2	2
	5	5	5	3	3	3	3	2	2
	6	6	6	6	6	6	6	3	0
	7	7	7	0	0	0	0	0	0
	8	8	8	8	8	6	8	3	3
2	1	1	1	0	0	0	0	0	0
	2	2	2	0	0	0	0	0	0
	3	3	3	0	0	0	0	0	0
	4	4	4	4	4	4	4	1	1
	5	5	5	5	5	5	5	2	2
	6	6	6	6	6	5	0	2	0
	7	7	7	6	6	5	0	2	0
	8	8	8	5	5	5	5	2	2

Shrnutím předchozího výkladu je zřejmé, že hledání nezávislých obvodových veličin v programu CIRNAM se skládá z těchto čtyř částí:

a) transformace působením spínačů,

b) vynechání nepřipojených uzlů,

c) transformace vlivem neregulárních prvků,

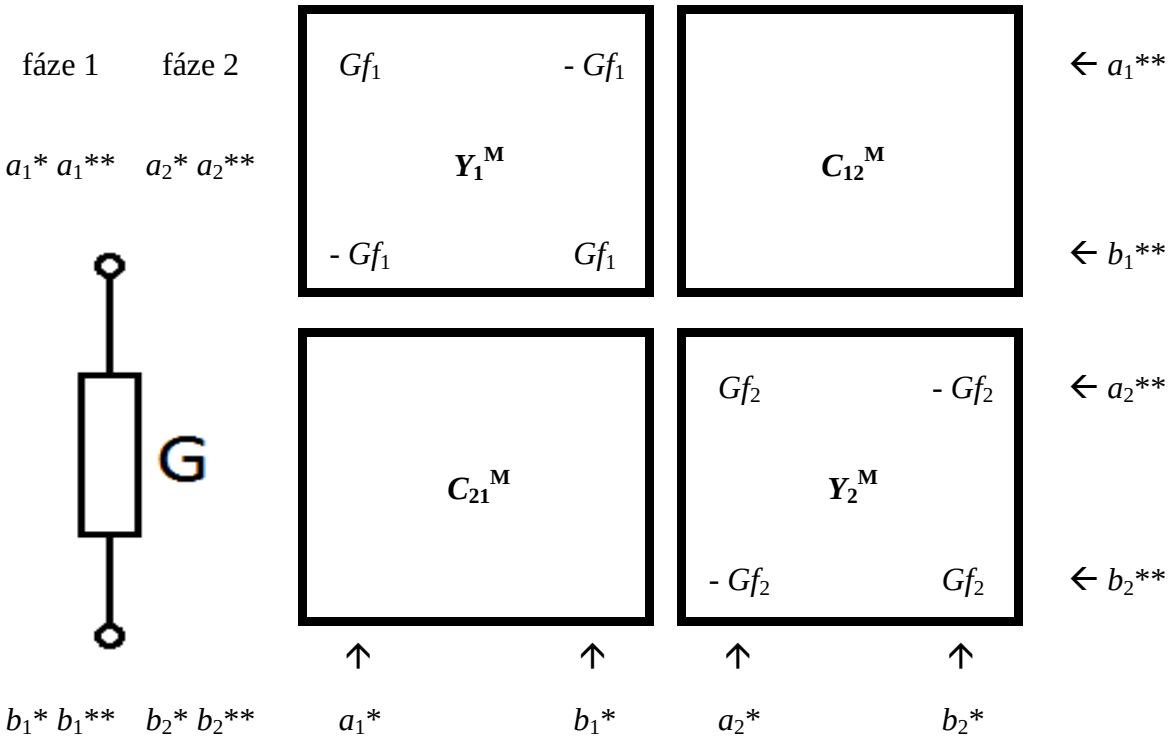
d) vzestupné přečíslování koeficientů.

4.3 Tvorba obvodových matic

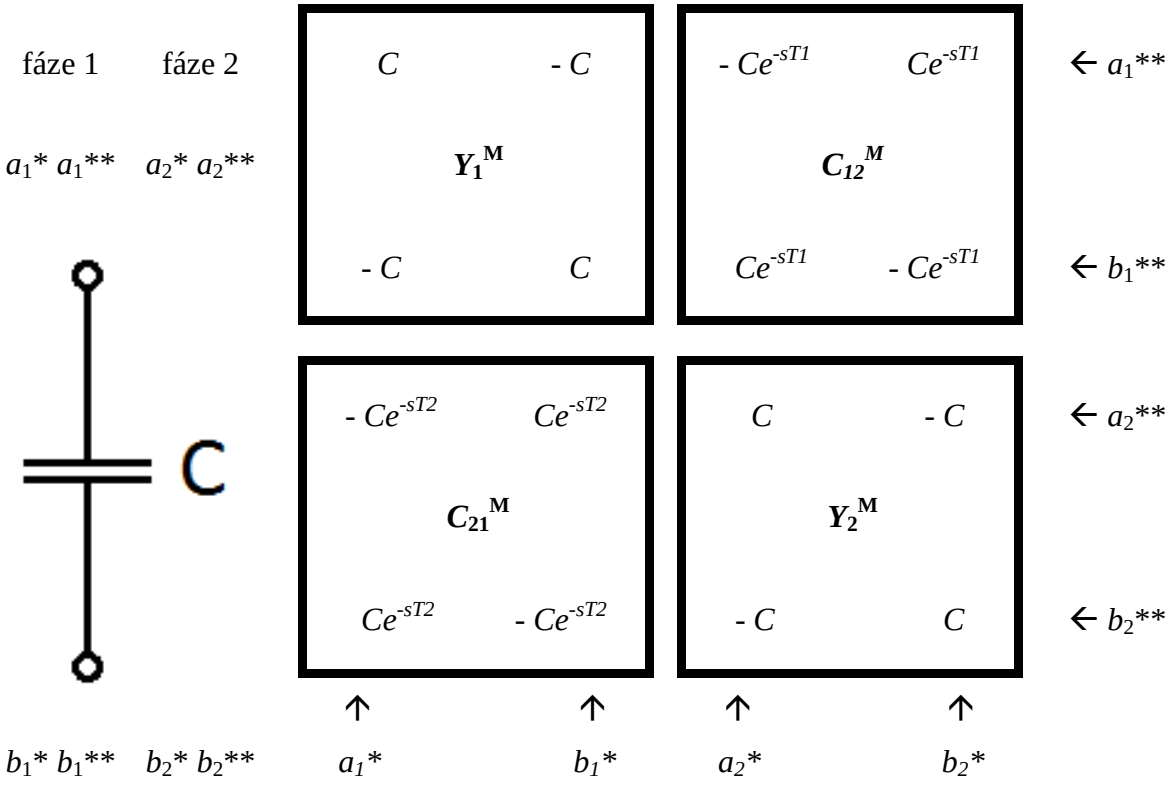
V této části je popsána tvorba obvodových matic, neboť každá součástka obvodu se zapisuje jiným způsobem. Výsledné matice jsou pak ořezány o sloupec a řádek referenčního uzlu, tedy o první řádek a sloupec, a jejich spojením se vytvoří tzv. supermatice, ze které jsou pak vypočítány hodnoty kmitočtové charakteristiky. Popis algoritmu jakým program vytvoří matice a zapíše data je uveden v kapitole 6.

4.3.1 Zápis submatic pasivních prvků R a C

Zápis je prováděn na základě zobecněného pravidla pro tvorbu admitanční matice analogových obvodů. Vzhledem k tomu, že je uvažován obvod s dvoufázovým spínáním, je třeba sestavit čtyři matice: klasickou admitanční matici pro každou fázi a dvě tzv. přechodové matice. Prvky přechodových matic jsou tvořeny parametry reaktančních členů, které zprostředkovávají přenos energie z jedné spínací fáze do druhé. Způsob zápisu do obvodových matic spínaného obvodu je vidět na obr. 3 a obr. 4.



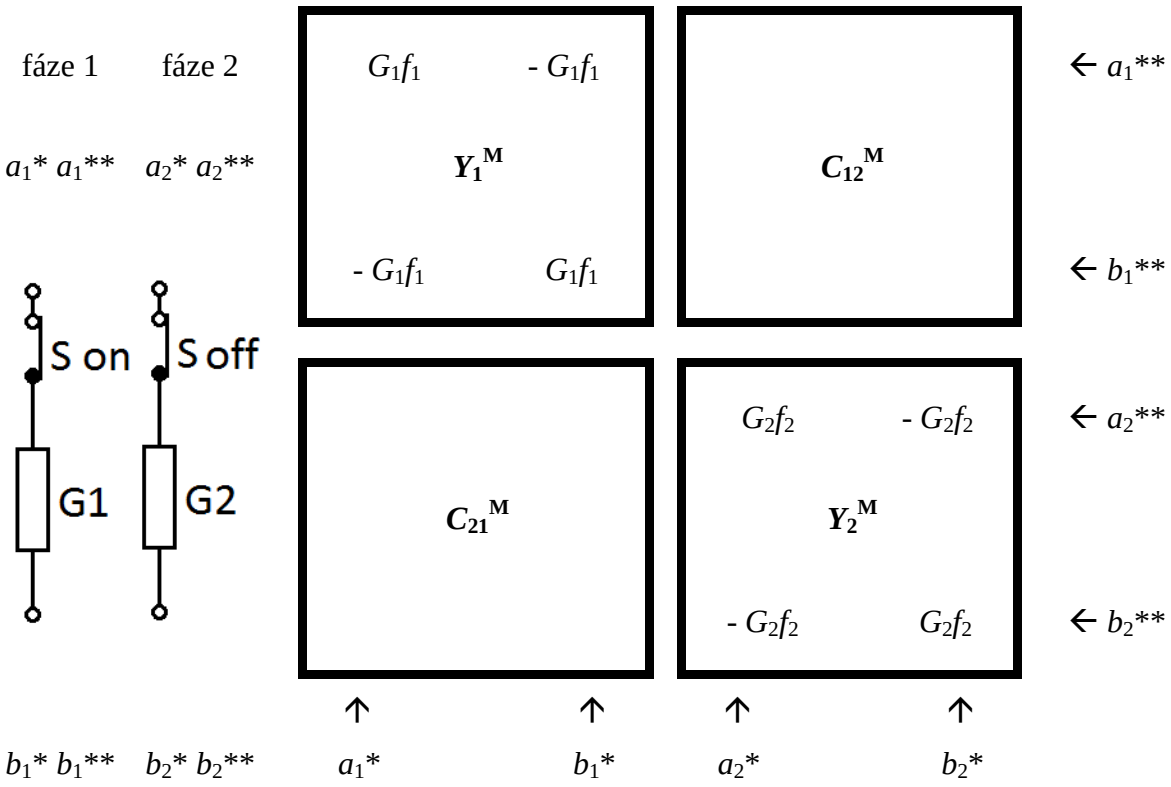
Obr. 3: Způsob zápisu vodivostí do obvodových matic spínaného obvodu.



Obr. 4: Způsob zápisu kapacit do obvodových matic spínaného obvodu.

4.3.2 Zápis submatic reálných spínačů

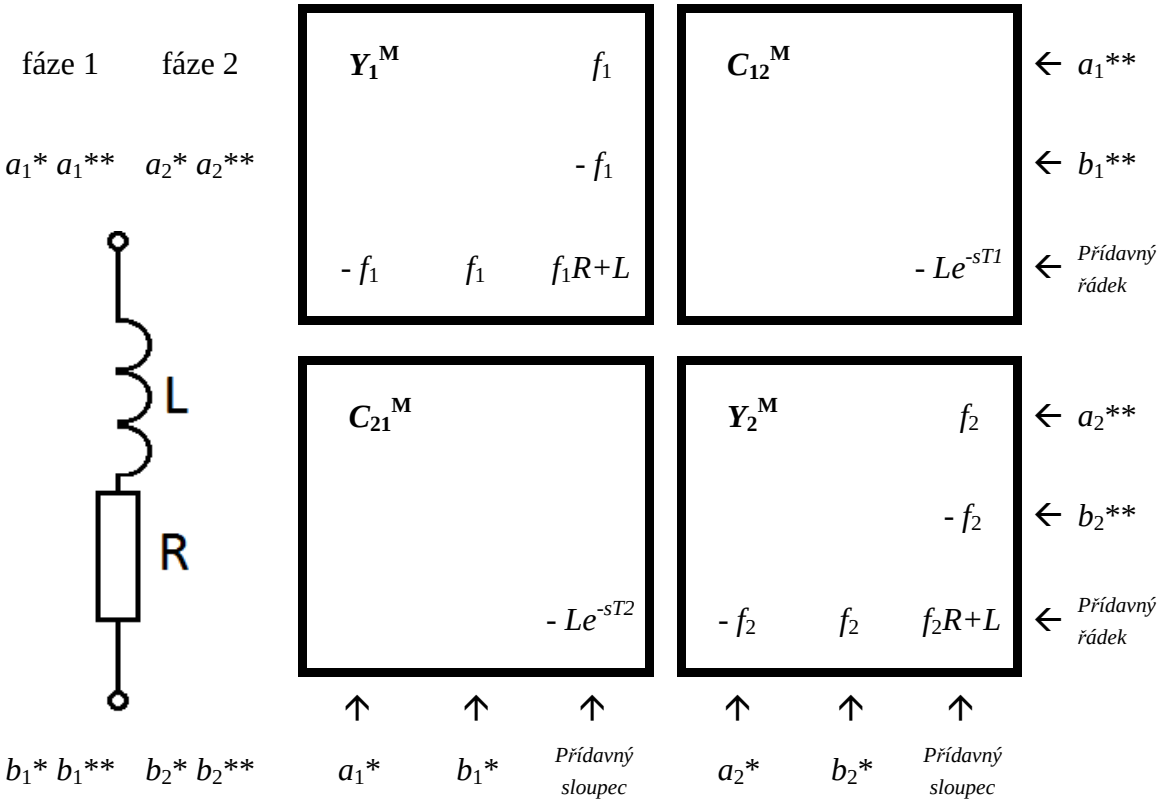
Zápis je podobný zápisu rezistorů. V případě reálných spínačů se odpor spínačů v sepnutém stavu projeví pouze v admitanční matici fáze, kdy je spínač sepnut. Způsob zápisu spínače sepnutého ve fázi 1 a spínače sepnutého ve fázi 2 do obvodových matic spínaného obvodu je vidět na obr. 5.



Obr. 5: Způsob zápisu spínačů do obvodových matic spínaného obvodu.

4.3.1 Zápis submatic vlastních indukčností

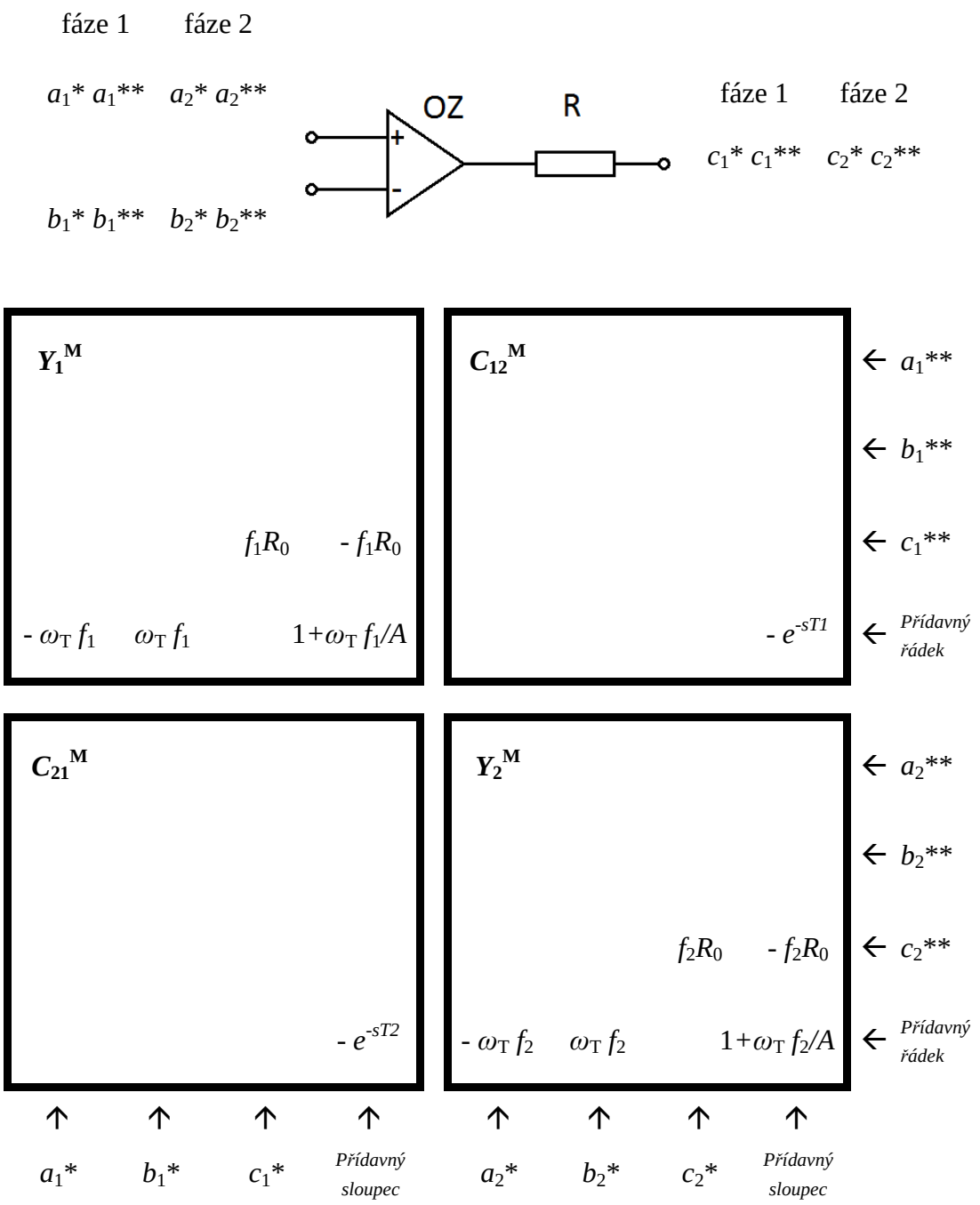
Induktor se susceptancí 1 / *(sL)* nevyhovuje teoretickému předpokladu, ve kterém musí admitanční matice obsahovat maximálně první mocniny operátoru *s*. Modifikovaná metoda uzlových napětí ale umožňuje použít impedanční popis ve formě přídavné rovnice. Vlivem proudu induktorem se rozšiřuje v každé fázi počet nezávislých branových veličin o jedna, čímž nám vzroste počet rovnic a dojde ke zvětšení matic. Způsob zápisu do obvodových matic spínaného obvodu je vidět na obr. 6.



Obr. 6: Způsob zápisu spínačů do obvodových matic spínaného obvodu.

4.3.1 Zápis submatic reálného operačního zesilovače

Program je schopen pracovat se dvěma typy operačních zesilovačů. Prvním typem je ideální operační zesilovač, jehož vliv na obvodové rovnice je pouze v redukci počtu nezávislých branových veličin. Druhý typ je lineární operační zesilovač se standardním průběhem kmitočtové charakteristiky. Při zápisu do netlistu je nutné zadat stejnosměrné zesílení A_0, tranzitní kmitočet ω_T a hodnotu výstupního odporu R_0. Způsob zápisu tohoto operačního zesilovače do obvodových matic spínaného obvodu je vidět na obr. 7.



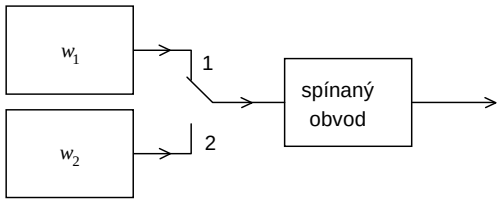
Obr. 7: Způsob zápisu spínačů do obvodových matic spínaného obvodu.

4.4 Buzeni obvodu

Obecné schéma buzení spínaného obvodu s dvoufázovým spínáním je znázorněn na obr. 8. Na jehož základě lze rozlišit tyto případy [1]:

- signál w_1, resp. w_2 má takové spektrum, že po jeho vzorkování s kmitočtem $f_v = 1 / T = 1 / (T_1 + T_2)$ nedojde k překrývání nenulových původních a periodizovaných spektrálních složek,
- jediný signál $w_1 = w_2 = w$. Spektrum tohoto signálu musí splňovat podmínku z předchozího případu,
- jediný vstupní signál schodovitého charakteru,
- jediný vstupní signál charakteru "Sample-Hold" s režimem "Hold" přes obě spínací fáze,
- signály w_1 a w_2 mají charakter "Sample-Hold".

V programu jsou zakomponovány pouze první tři způsoby buzení obvodu.



Obr. 8: Obecný způsob buzení dvoufázového spínacího obvodu různými signály v každé fázi [1].

4.5 Modifikace obvodových matic působením budícího zdroje

Je uvažován budící zdroj napětí u_{vst} o Laplaceově obrazu U_{vst} zapojený mezi uzel x a referenční uzel, pak lze rovnice 1. Kirchhoffova zákona napsat v obou fázích jednoduchou napěťovou rovnicí $U_{vst} = 1 \times U_x$, z čehož vyplývá, že modifikace obvodových matic bude následující [1]:

- provede se vynulování řádků matic odpovídajících proudovým koeficientům vstupního uzlu x:

$$Y_1^M\ (x_1{**}, k) = C_{21}^M\ (i_2{**}, k) = 0, \qquad k = 1, 2, \ldots, n_1,$$

$$Y_2^M\ (x_2{**}, k) = C_{12}^M\ (i_1{**}, k) = 0, \qquad k = 1, 2, \ldots, n_2.$$

- provede se dosazení jedniček do vynulovaných řádků admitančních matic na pozice odpovídající napěťovým koeficientům vstupního uzlu x:

$$Y_1^M\ (x_1{**}, x_1{*}\ ) = 1, \qquad Y_2^M\ (x_2{**}, x_2{*}\ ) = 1.$$

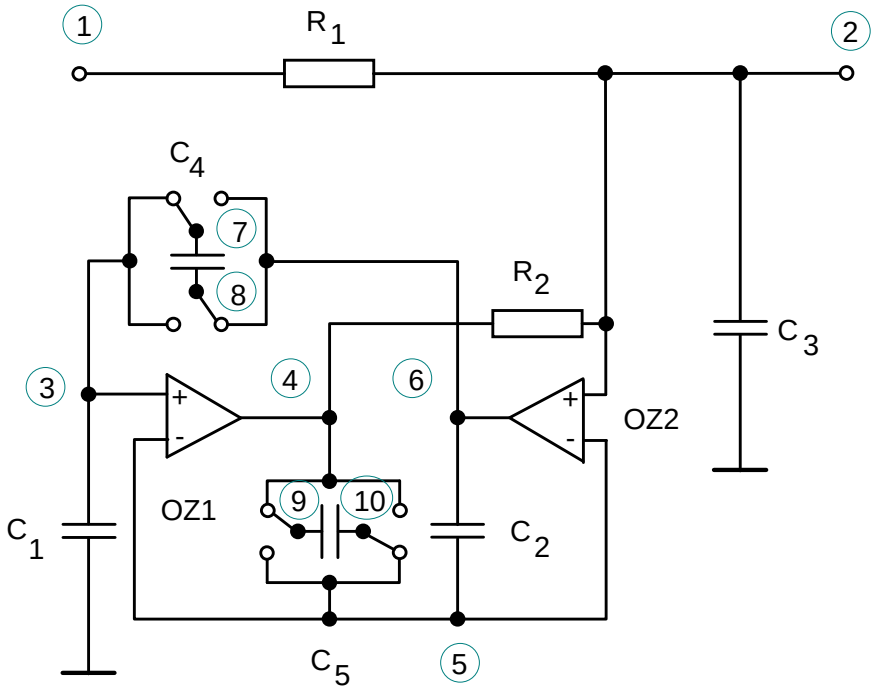
4.6 Příklad sestavení obvodových matic

V této části je na obvodu filtru typu dolní propust na obr. 9 ukázán algoritmus tvorby obvodových matic. Pro tuto ukázku jsou uvažovány ideální spínače a operační zesilovače se standardní kmitočtovou charakteristikou s parametry A_0, ω_T a R_0. Je tedy zřejmé, že proběhne pouze jediná transformace díky spínačům s nulovým odporem v sepnutém stavu. Dále je patrné, že koeficienty každého uzlu budou shodné a není třeba je rozlišovat [1]. Výsledek redukce proměnných je zachycen v tab. 2.

Obvodové matice budou co do zápisu téměř symetrické. Na obr. 10 je ukázán zápis pasivních prvků R a C vzorového obvodu do admitanční Y_1^M a přechodové C_{21}^M matice. Zbylé dvě matice si lze představit tak, že v matici Y_1^M se zamění f_1 za f_2 a v matici C_{21}^M se zamění T_1 za T_2. Na dalším obrázku (obr. 11) jsou matice z obr. 10 doplněny o zápis aktivních prvků obvodu, v tomto případě dvou operačních zesilovačů $OZ1$ a $OZ2$.

Na posledním obrázku (obr. 12) této části je pak ukázán vliv budícího zdroje napětí (připojený k uzlu 1) na obvodové matice.

Nevyplněná pole v matici obsahují hodnotu 0.

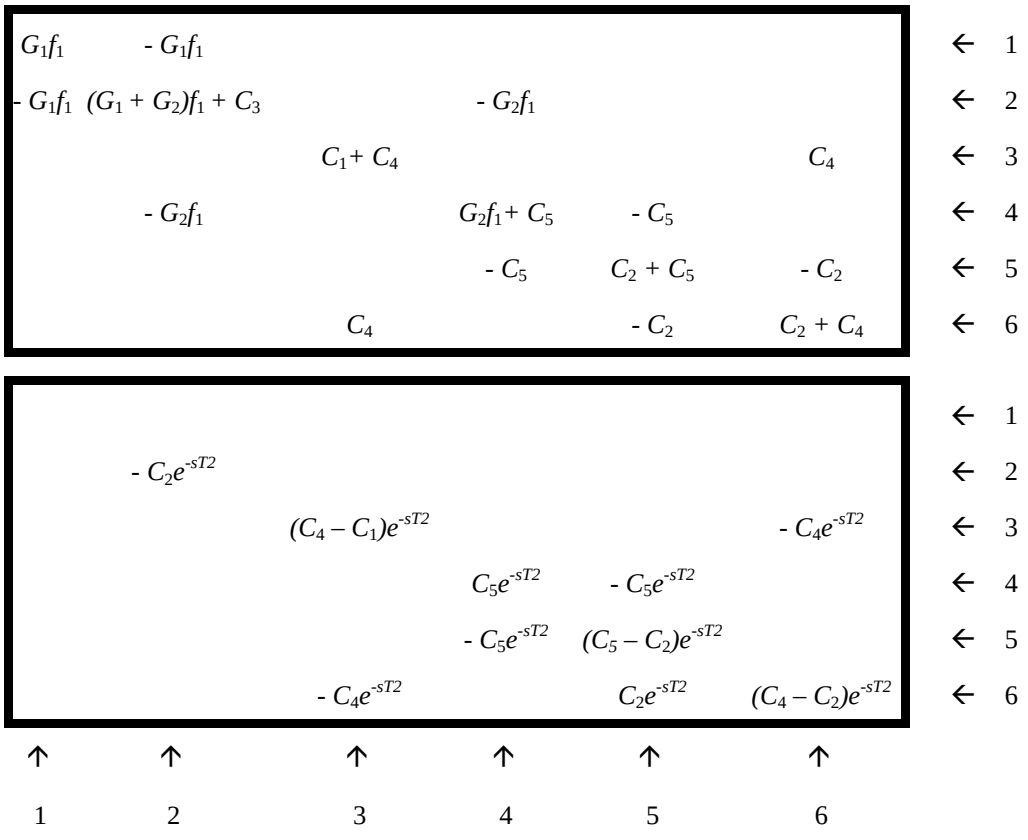


Obr. 9: Spínaný filtr typu dolní propust [1].

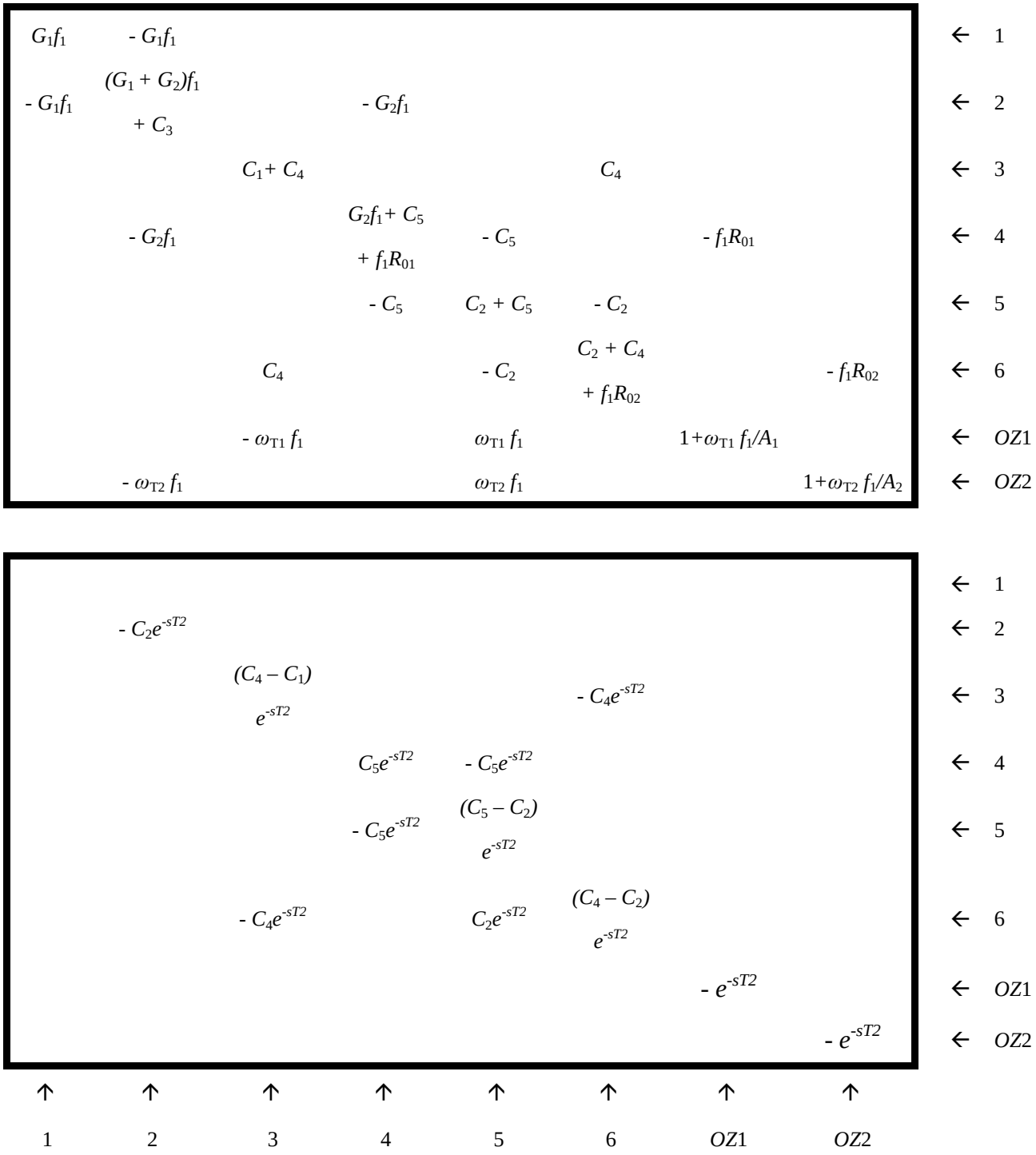
Tab. 2: Redukce proměnných.

Uzel	Koeficienty po transformaci spínači	
	Fáze 1	Fáze 2
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

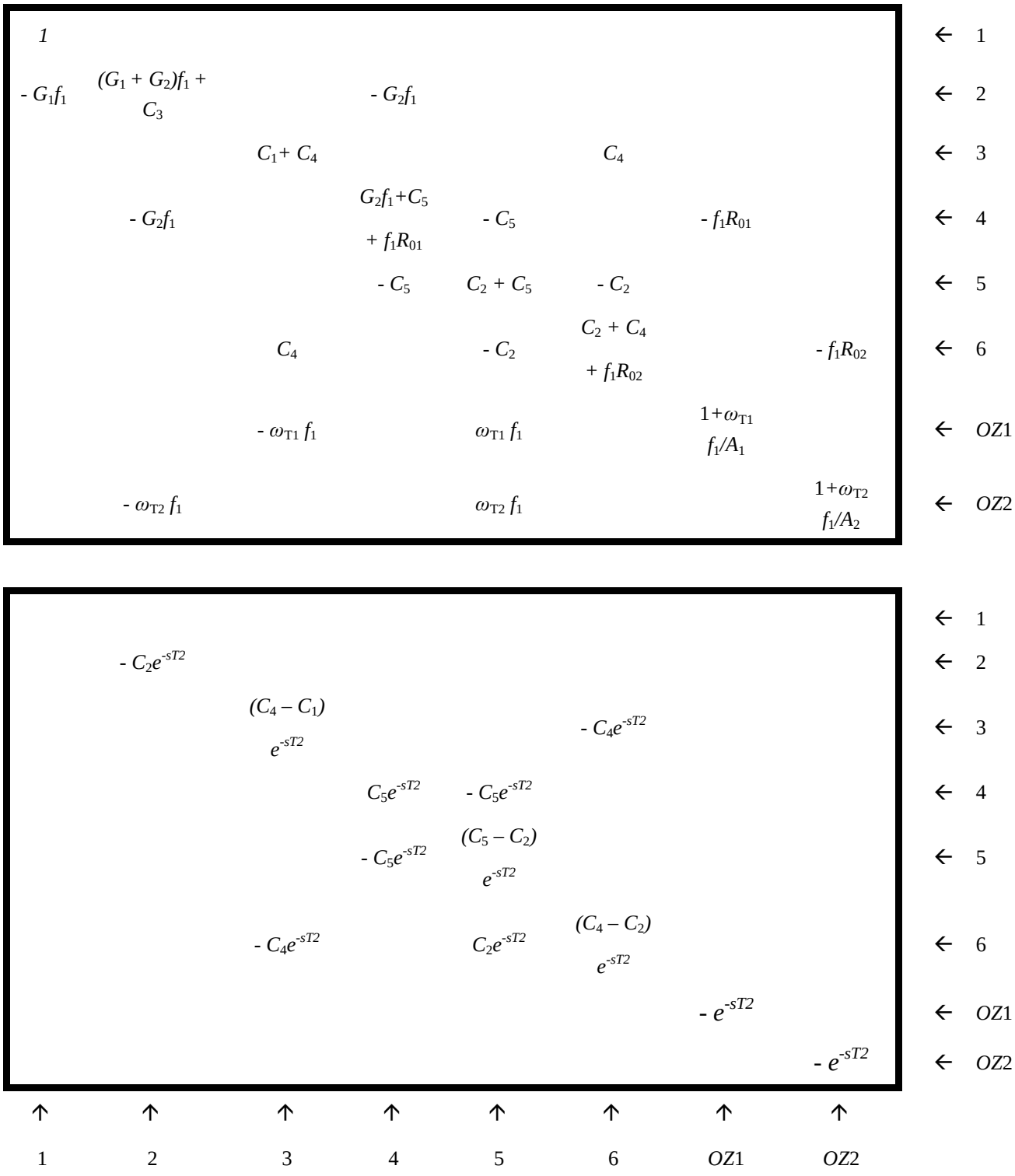
Uzel	Koeficienty po transformaci spínači	
	Fáze 1	Fáze 2
6	6	6
7	3	6
8	6	3
9	4	5
10	5	4



Obr. 10: Zápis pasivních prvků *R* a *C* do admitanční Y_1^M (horní) a přechodové C_{21}^M (spodní) obvodové matice.



Obr. 11: Zápis pasivních prvků *R* a *C* a aktivních prvků *OZ* do admitanční Y_1^M (horní) a přechodové C_{21}^M (spodní) obvodové matice.



Obr. 12: Vliv budícího zdroje napětí na obvodové matice.

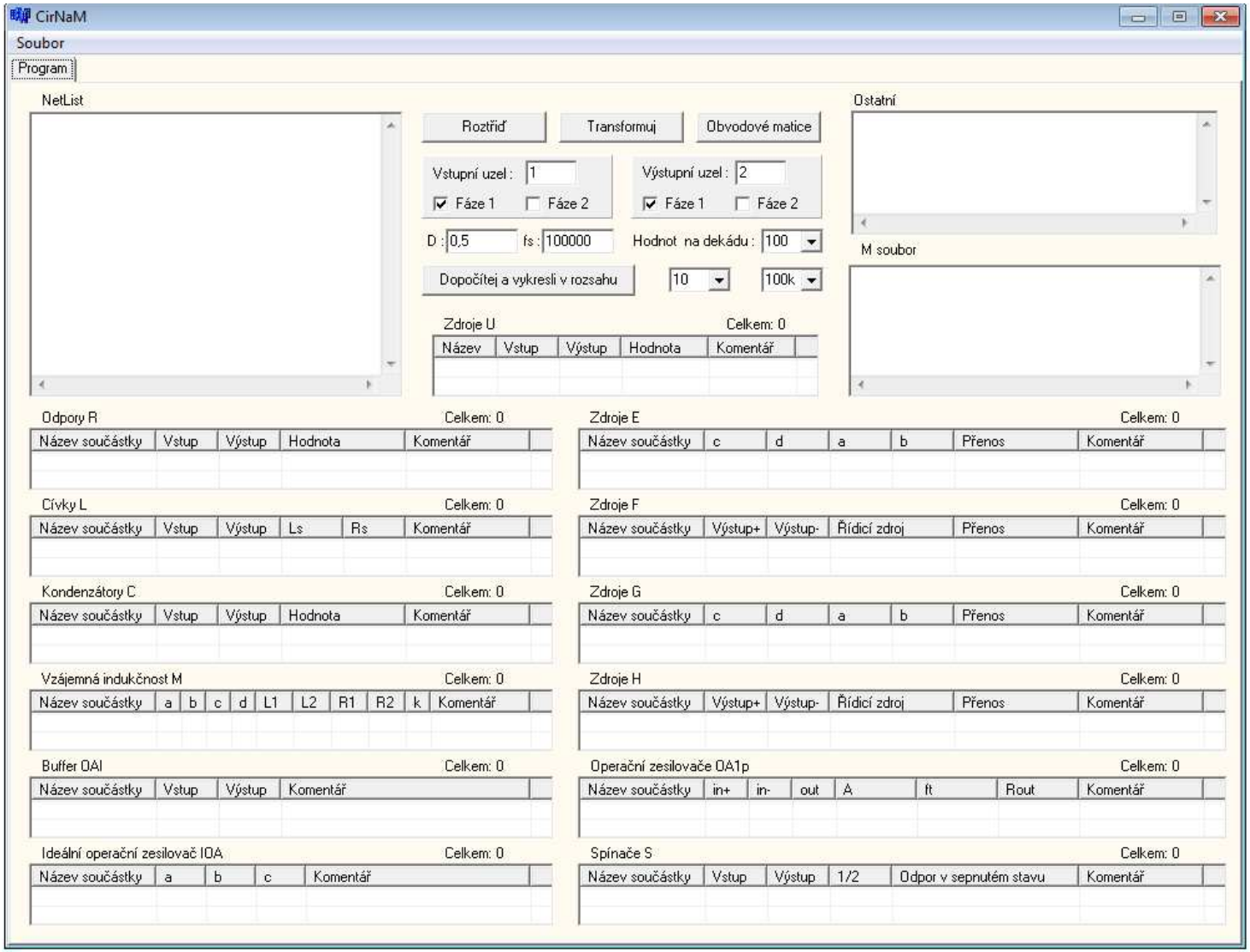
5 Program CIRNAM

Program pro analýzu obvodů vznikl postupným rozšiřováním původního konceptu, který měl za úkol rozpoznat jednotlivé součástky v modifikovaném netlistu. Postupem času byl tento koncept upravován tak, aby zredukoval počet obvodových rovnic a sestavil admitanční matice pro obě fáze a přechodové matice popisující děje mezi těmito fázemi. Takto zpracovaný obvod byl začleněn do dalšího zdrojového kódu a uložen do souboru, který lze spouštět programem MATLAB. Zdrojový kód obsažený ve výstupním souboru programu CIRNAM obsahoval syntaxe pro dopočítání hodnot kmitočtové charakteristiky analyzovaného obvodu.

Při dalším vývoji programu byla snaha osamostatnit program od dalších aplikací a začlenit veškeré algoritmy analýzy obvodu do jediného programu, na jehož výstupu by byly vykresleny kmitočtové charakteristiky.

Vzhledem k obtížnosti realizace složitých matematických operací v programovacím jazyce C++ byla v průběhu tvorby práce změněna teorie výpočtu kmitočtových charakteristik a oproti široce zaměřenému původnímu konceptu současné algoritmy obsažené v programu jsou omezeny na obvody, které obsahují následující součástky: ideální i reálné spínače, pasivní prvky *R*, *L* a *C*, jednotkové zesilovače, ideální operační zesilovače a operační zesilovače s jedním kmitočtem lomu.

Program umožňuje nejen načítat předem připravený popis obvodu, ale je možno tento popis editovat, upravovat a samozřejmě si jej i uložit. Také výstup z programu je možné uložit do podoby vstupního souboru programu MATLAB k pozdějšímu vyvolání výsledků analýzy. Ačkoliv program na konci analýzy obvodu vykreslí kmitočtové charakteristiky obvodu, neumožňuje tyto charakteristiky zálohovat do vlastní paměti a později je znovu vykreslit.



Obr. 13: Úvodní vzhled okna po spuštění programu CIRNAM.

5.1 Uživatelské prostředí

Program je uživatelsky velice intuitivní a obsahuje řadu ošetření, která pohlídají, aby uživatel vyplnil potřebná data pro analýzu obvodu. Po spuštění programu je vzhled okna programu velmi podobný tomu, který je vidět na obr. 13 (pro lepší názornost popisu jsou viditelné na tomto obrázku veškeré ovládací prvky).

Hlavní okno programu lze rozdělit do několika částí. Vstupní částí programu je velké textové pole v levé horní části nadepsané nápisem „NetList“, které obsahuje popis analyzovaného obvodu vytvořeného na základě syntaxí popsaných v kapitole 2.

Velkou část okna programu tvoří seznamy součástek, které jsou obsažené v našem popisu obvodu. Každý seznam je nadepsán příslušným názvem součástky, za kterým je uveden začátek jejího zápisu v netlistu, což může pomoci při dodatečné úpravě nebo psaní popisu obvodu přímo v programu CIRNAM. Nad každým seznamem je i pro informaci uveden počet nalezených součástek daného typu. V případě, že netlist obsahuje syntaxe, které program nerozpozná, jsou tyto syntaxe uvedeny v pravém horním rohu v textovém bloku nadepsaném „Ostatní“.

Pod tímto blokem je poslední textové pole obsahující výstupní M soubor, který obsahuje tři matice zapsané v syntaxi programu MATLAB a je tedy možno tyto matice přímo kopírovat do spuštěného programu MATLAB. Jedná se o matici „*freq*“, která obsahuje hodnoty frekvencí, pro které byly vypočítány hodnoty amplitudové (matice „*dB“)* a fázové (matice „*ph*“) charakteristiky.

Poslední a nejdůležitější část programu se nachází mezi výše zmíněnými textovými poli uprostřed horní poloviny okna. Tato ovládací část bude popsána v následující podkapitole.

5.2 Ovládání programu

Analýza obvodu je i v dnešní době časově náročná a program je tedy navržen tak, aby uživatel mohl jeho hlavní části výpočtu kontrolovat. Pro běžného uživatele je zobrazena pouze první záložka „Program“, ale je možné aktivovat další záložky, v nichž je možné vidět transformaci uzlů, náhled submatic nebo „supermatici“. Na základě této koncepce musí uživatel udělat o pár kliknutí při analýze navíc. Další výhodou této koncepce je možnost průběžně měnit veškeré zadávané parametry aniž by program průběžně vypočítával výstupní charakteristiky a zdržoval tak uživatele.

5.2.1 Načtení vstupních dat

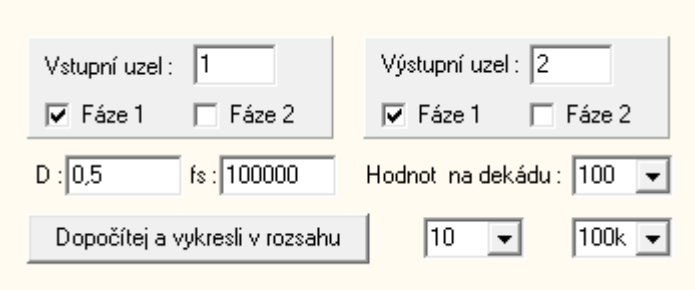
Vstupní data mohou být obsažena v klasickém textovém souboru s příponou *.txt* nebo ve formátu souboru PSpice s příponou *.cir*. Načtení vstupních dat lze provést přes standardní rolovací menu vlevo nahoře „Soubor“ → „Otevřít“ a v dialogovém okně se vybere daný soubor a potvrdím stiskem tlačítka „Otevřít“. Obsah souboru se vepíše do textového pole nadepsaného netlist, kde je možné ručně doladit syntaxe, popřípadě upravit popis obvodu.

5.2.1 Analýza obvodu

Uprostřed horní poloviny okna je místo, kde jsou postupně nabízena uživateli ke stisku tlačítka spouštějící jednotlivé hlavní kroky analýzy. Pokud je načten popis obvodu je možné stisknout tlačítko „Roztřiď“. Po jeho stisku jsou zaplněny seznamy součástkami obsaženými v popisu obvodu a je možné stisknout další tlačítko „Transformuj“. Jeho stisknutím program provede nalezení nezávislých branových veličin, přečísluje uzly obvodu a umožní stisk tlačítka „Obvodové matice“.

Po sestavení obvodových matic se v řídící části programu spolu s posledním potvrzujícím tlačítkem „Dopočítej a vykresli v rozsahu“ objeví několik polí, které je nutné vyplnit pro dokončení analýzy obvodu, obr. 13. Program vyžaduje zadání:

- čísla vstupního a výstupního uzlu,
- fáze, ve které bude obvod buzen (lze budit v obou fázích)
- fáze výstupního uzlu
- střídu D a spínací kmitočet f_s,
- rozsah frekvencí, v níž jsou počítány hodnoty kmitočtových charakteristik,
- počet hodnot na dekádu rozsahu.

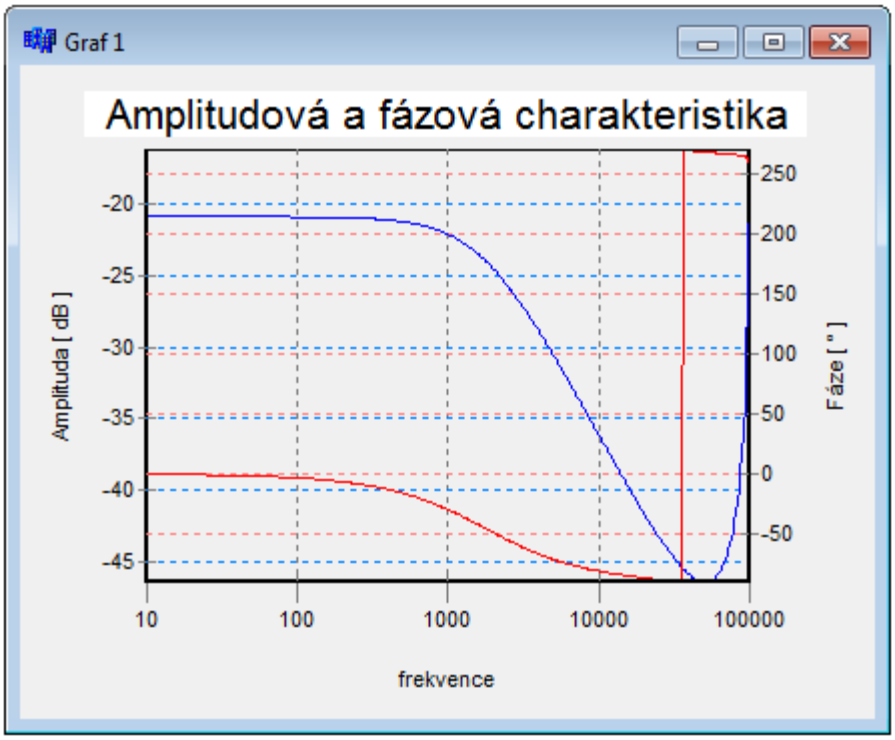


Obr. 13: Ovládací panel vyžadující zadání parametrů pro dokončení analýzy obvodu.

5.2.1 Ovládací prvky okna s kmitočtovou charakteristikou

Po dokončení analýzy, tedy po stisknutí tlačítka „Dopočítej a vykresli v rozsahu“, se po každé vytvoří nové okno s amplitudovou a fázovou charakteristikou obr. 14.

U tohoto okna je možné měnit jeho rozměry, tak jak je každý zvyklý z prostředí Windows. Dále je možné kliknutím na plochu grafu a tažením kurzoru směrem doprava dolů vybrat část charakteristiky, která bude zvětšena. Opačným pohybem po kliknutí se graf vrátí do původního stavu.



Obr. 14: Příklad okna s kmitočtovou charakteristikou.

5.2.1 Uložení vstupních a výstupních dat

Uložení popisu obvodu je možné přes horní menu volbou „Soubor“ → „Uložit NetList“, kdy se otevře dialogové okno vyzívající k zadání názvu souboru a k výběru přípony souboru (v tomto případě **.cir* nebo **.txt*). Podobný postup je pro uložení matic s vypočítanými hodnotami. V horním menu se vybere volba „Soubor“ → „Uložit Msoubor“ a opět se otevře dialogové okno vyzívající k zadání názvu souboru a k výběru přípony souboru (v tomto případě **.m nebo *.txt*).

6 Implementace teorie do programu

Zdrojový kód programu je značně obsáhlý a pro případ rozšiřování možností programu bude v této části uveden přehled vytvořených tříd, metod a funkcí a u některých z nich popsán podrobněji celý algoritmus.

6.1 Komplexní čísla

Většina výpočtů programu je v komplexní rovině. Pro tyto výpočty je vytvořena nová struktura *FComplex*, která se skládá ze dvou částí: reálné a imaginární části hodnoty komplexního čísla.

```
typedef struct FCOMPLEX {double r,i;} fcomplex;
```

6.2 Funkce s komplexními čísly

Tyto funkce zajišťují základní výpočty s komplexními čísly, tedy se strukturou *FComplex*. Mezi těmito funkcemi se nachází i některé často opakované výpočty s komplexními čísly.

Complex

Funkce naplní strukturu *FComplex*, čímž vytvoří komplexní číslo. Jako vstupní parametry přijímá dvě číselné proměnné typu *double*. První hodnota je hodnota reálné části komplexního čísla a druhá hodnota je hodnota imaginární části komplexního čísla. Funkce vrací hodnotu *fcomplex*.

```
fcomplex Complex(double re, double im)
```

Cadd

Funkce sečte dvě komplexní čísla. Jako vstupní parametry přijímá dvě hodnoty typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex Cadd(fcomplex a, fcomplex b)
```

Csub

Funkce odečte dvě komplexní čísla. Jako vstupní parametry přijímá dvě hodnoty typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex Csub(fcomplex a, fcomplex b)
```

Cmul

Funkce vynásobí dvě komplexní čísla. Jako vstupní parametry přijímá dvě hodnoty typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex Cmul(fcomplex a, fcomplex b)
```

Cdiv

Funkce vydělí dvě komplexní čísla. Jako vstupní parametry přijímá dvě hodnoty typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex Cdiv(fcomplex a, fcomplex b)
```

Conjg

Funkce vypočítá číslo komplexně sdružené k zadanému vstupnímu parametru. Vstupní parametr je typu *komplex* a výsledek je opět typu *fcomplex*.

```
fcomplex Conjg(fcomplex z)
```

Cabs

Funkce vypočítá absolutní hodnotu komplexního čísla. Vstupní parametr je typu *fcomplex* a výstupní hodnota typu *double*.

```
double Cabs(fcomplex z)
```

Csqrt

Funkce vypočítá druhou odmocninu komplexního čísla. Vstupní parametr je typu *fcomplex* a stejně tak i návratová hodnota je typu *fcomplex.*

```
fcomplex Csqrt(fcomplex z)
```

RCadd

Funkce přičte reálné číslo ke komplexnímu číslu. Jako vstupní parametry přijímá dvě hodnoty: reálné číslo typu *float* a komplexní číslo typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex RCadd(float x, fcomplex a)
```

RCmull

Funkce vynásobí reálným číslem komplexní číslo. Jako vstupní parametry přijímá dvě hodnoty: reálné číslo typu *float* a komplexní číslo typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex RCmul(float x, fcomplex a)
```

RCDiv

Funkce vydělí reálné číslo komplexním číslem. Jako vstupní parametry přijímá dvě hodnoty: reálné číslo typu *float* a komplexní číslo typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex RCDiv(float a, fcomplex b)
```

onedivC

Speciální případ funkce *RCDiv*, který je velmi často využíván při tvorbě matic. Funkce vypočítá převrácenou hodnotu komplexního čísla. Jako vstupní parametry přijímá komplexní číslo typu *fcomplex* a vrací hodnotu typu *fcomplex*.

```
fcomplex onedivC(fcomplex a)
```

6.3 Matice komplexních čísel TComplexArray

Veškeré hodnoty součástek se zapisují výše popsaným způsobem do matic, se kterými se dále provádějí výpočty vedoucí k výsledku analýzy. Pro tyto účely byla vytvořena třída *TComplexArray*. Tato třída má veřejný konstruktor, který má dva argumenty typu *integer*, z nichž první určuje počet sloupců a druhý počet řádků této matice.

```
TComplexArray(int AColCount, int ARowCount);
```

6.4 Metody třídy TComplexArray

Následuje výpis „počítání“ matic s komplexními čísly metod třídy *TComplexArray* s krátkým popisem. Orientace v matici probíhá oproti běžným zvyklostem od nultého sloupce a nultého řádku, neboť při programování se uvažuje první hodnota nula na rozdíl od běžné jedničky.

GetRowCount

Metoda vrací počet řádků komplexní matice.

```
int GetRowCount();
```

GetColCount

Metoda vrací počet řádků komplexní matice.

```
int GetColCount();
```

GetColumn

Metoda vrací daný sloupec matice. Vstupní parametr určuje číslo sloupce matice. Výstupem je vektor komplexních čísel.

```
TList* GetColumn(int AColIndex);
```

GetItem

Metoda vrací danou hodnotu matice. Vstupní parametry určují pozici hodnoty, první číslo sloupec a druhé číslo řádek matice. Výstupní hodnota je komplexní číslo ve struktuře fcomplex.

```
fcomplex GetItem(int AColIndex, int ARowIndex);
```

SetItem

Metoda zapíše komplexní číslo do matice na danou pozici. První vstupní parametr určuje číslo sloupce a druhé číslo řádku matice, na který má být třetí vstupní parametr (hodnota komplexního čísla typu fcomplex) zapsán.

```
void SetItem(int AColIndex, int ARowIndex, fcomplex NewItem);
```

SetItemReIm

Obdoba metody *SetItem* umožňuje zapsat přímo reálnou a imaginární část komplexního čísla do matice na danou pozici. První vstupní parametr určuje číslo sloupce a druhé číslo řádku matice, na který mají být zapsány třetí (hodnota reálné části komplexního čísla typu double) a čtvrtý (hodnota imaginární části komplexního čísla typu double) vstupní parametr.

```
void SetItemReIm(int AColIndex, int ARowIndex,
        double compRe, double compIm);
```

SetItemSum

Metoda přičte komplexní číslo k číslu v dané pozici matice. První vstupní parametr určuje číslo sloupce a druhé číslo řádku matice, na který má být třetí vstupní parametr (hodnota komplexního čísla typu fcomplex) zapsán.

```
void SetItemSum(int AColIndex, int ARowIndex,
                fcomplex NewItem);
```

SetItemSumReIm

Obdoba metody *SetItemSum* umožňuje přičíst přímo reálnou a imaginární část komplexního čísla k číslu v dané pozici matice. První vstupní parametr určuje číslo sloupce a druhé číslo řádku matice, na který mají být zapsány třetí (hodnota reálné části komplexního čísla typu double) a čtvrtý (hodnota imaginární části komplexního čísla typu double) vstupní parametr.

```
void SetItemSumReIm(int AColIndex, int ARowIndex,
        double compRe, double compIm);
```

NullAll

Metoda vynuluje matici.

```
void NullAll();
```

NullCol

Metoda vynuluje zvolený sloupec matice. Metoda vyžaduje zadání vstupního parametru typu *integer* udávajícího pozici sloupce v matici (číslováno od 0).

```
void NullCol(int AColIndex);
```

NullRow

Metoda vynuluje zvolený řádek matice. Metoda vyžaduje zadání vstupního parametru typu *integer* udávajícího pozici řádku v matici (číslováno od 0).

```
void NullRow(int ARowIndex);
```

OnesAll

Metoda vyplní matici komplexní hodnotou $1+i$.

```
void OnesAll();
```

OnesDiag

Metoda vyplní hlavní diagonálu matice komplexní hodnotou $1+i$.

```
void OnesDiag();
```

MulNum

Metoda vynásobí matici hodnotou komplexního čísla. Metoda vyžaduje zadání vstupního parametru typu *fcomplex* udávajícího násobitel.

```
void MulNum(fcomplex AValue);
```

DivlNum

Metoda vydělí matici hodnotou komplexního čísla. Metoda vyžaduje zadání vstupního parametru typu *fcomplex* udávajícího dělitel.

```
void DivNum(fcomplex AValue);
```

Trace

Metoda vypočítá stopu matice. Návratová hodnota metody je typu *fcomplex*.

```
fcomplex Trace();
```

PlusMat

Metoda přičte k matici matici třídy *TComplexArray*. Pokud jsou splněny podmínky pro sčítání matic a metoda proběhne, vrací hodnotu 1, v opačném případě hodnotu 0.

```
int PlusMat(TComplexArray* AMatrix);
```

MulMat

Metoda vynásobí matici maticí třídy *TComplexArray*. Pokud jsou splněny podmínky pro násobení matic a metoda proběhne, vrací hodnotu *TComplexArray*.

```
TComplexArray* MulMat(TComplexArray* AMatrix);
```

MulMatMat

Metoda vynásobí dvě matici maticí třídy *TComplexArray*, které jsou jejími vstupními parametry. Pokud jsou splněny podmínky pro násobení matic a funkce proběhne, vrací hodnotu 1, v opačném případě hodnotu 0.

```
int MulMatMat(TComplexArray* AMatrixA, TComplexArray* AMatrixB);
```

Luc

Speciální metoda, která provede LU rozklad matice. Tato funkce byla přejata a upravena z [6] a je použita při výpočtu inverzní matice.

```
void Luc(TComplexArray* AMatrix,TComplexArray* indx, int &d);
```

Solvec

Speciální metoda, která řeší rovnici $\boldsymbol{A} \,.\, x = \boldsymbol{B}$, kde $\boldsymbol{A}$ a $\boldsymbol{B}$ jsou matice komplexních čísel a $\boldsymbol{x}$ je reálné číslo. Tato funkce byla přejata a upravena z [6] a je použita při výpočtu inverzní matice.

```
void Solvec(TComplexArray* AMatrix, TComplexArray* indx,TComplexArray* b);
```

Switch

Metoda mění řádky za sloupce a naopak. Využívá se především při změně vektoru na sloupcový vektor.

```
void Switch();
```

SmallMatrix

Metoda ořeže matici o první řádek a první sloupec. Využívá se při tvorbě „supermatice“, kdy se admitanční a přechodové matice ořezávají o řádek a sloupec odpovídající referenčnímu uzlu.

```
TComplexArray* SmallMatrix();
```

SetNullRowAndValueReIm

Metoda vynuluje řádek matice a zapíše komplexní číslo na danou pozici matice. První vstupní parametr určuje číslo sloupce a druhé číslo řádku matice, na který má být třetí vstupní parametr (hodnota komplexního čísla typu fcomplex) zapsán. Funkce automaticky nuluje řádek odpovídající číslu řádku, na který je hodnota zapisována.

```
void SetNullRowAndValueReIm(int AColIndex, int ARowIndex, fcomplex AValue);
```

Metoda umožňuje zadání i čtyř parametrů. Potom poslední dvě hodnoty určují přímo reálnou a imaginární část komplexního čísla zapisovaného na pozici danou prvními dvěma parametry.

```
void SetNullRowAndValueReIm(int AColIndex, int ARowIndex, double AValueRe,
                            double AValueIm);
```

6.5 Vektor komplexních matic TVectorComArray

Třída *TVectorComArray* byla vytvořena pro numerické řešení Faddejevovi metody, která byla využita při řešení teorie popsáné v bakalářské práci [4]. Tato třída slouží pro vytvoření vektoru, jehož hodnoty jsou komplexní matice typu *TComplexArray*.

```
TVectorComArray(int ACount, int AColCount, int ARowCount);
```

6.6 Metody třídy TVectorComArray

Následuje výpis „počítání“ matic s komplexními čísly metod třídy *TComplexArray* s krátkým popisem. Orientace v matici probíhá oproti běžným zvyklostem od nultého sloupce a nultého řádku, neboť při programování se uvažuje první hodnota nula na rozdíl od běžné jedničky.

GetMatrix

Metoda vrací odkaz na komplexní matici typu *TComplexArray* na dané pozici vektoru určené vstupním parametrem typu *integer*.

```
TComplexArray* GetMatrix(int AIndex);
```

GetMatrixCopy

Metoda vrací komplexní matici typu *TComplexArray* na dané pozici vektoru určené vstupním parametrem typu *integer*.

```
TComplexArray* GetMatrixCopy(int AIndex);
```

SetMatrixCopy

Metoda zapíše komplexní matici typu *TComplexArray* na dané pozici vektoru určené vstupním parametrem typu *integer*.

```
void SetMatrixCopy(int AIndex, TComplexArray* ANewMatrix);
```

GetVectorFromAll

Metoda vrací vektor komplexních čísel typu *TComplexArray*. Vybere z každé matice jednu komplexní hodnotu z pozice určené vstupními parametry typu *integer*. První určuje sloupec a druhý řádek matice.

```
TComplexArray* GetVectorFromAll(int AColIndex, int ARowIndex);
```

GetVectorFromAll

Metoda vrací konkrétní komplexní číslo typu *fcomplex* z matice, jejíž pořadí je určeno prvním vstupním parametrem typu *integer* (číslování probíhá od *0*). Pozice v matici je určena dalšími vstupními parametry typu *integer*. První určuje sloupec a druhý řádek matice.

```
fcomplex GetItem(int AIndex, int AColIndex, int ARowIndex);
```

SetItem

Metoda zapíše konkrétní komplexní číslo typu *fcomplex* do matice, jejíž pořadí je určeno prvním vstupním parametrem typu *integer* (číslování probíhá od *0*). Pozice v matici je určena dalšími vstupními parametry typu *integer*. První určuje sloupec a druhý řádek matice.

```
void SetItem(int AIndex, int AColIndex, int ARowIndex, fcomplex NewItem);
```

SetItemReIm

Metoda zapíše hodnotu reálné a imaginární části komplexního čísla do matice, jejíž pořadí je určeno prvním vstupním parametrem typu *integer* (číslování probíhá od 0). Pozice v matici je určena dalšími vstupními parametry typu *integer*. První určuje sloupec a druhý řádek matice. Poslední dva vstupní parametry jsou hodnoty reálné a komplexní složky a jsou typu *double*.

```
void SetItemReIm(int AIndex, int AColIndex, int ARowIndex, double compRe,
                                                    double compIm);
```

6.7 Obecná matice TStringArray

Třída *TStringArray* byla vytvořena jako základ pro kontrolu sestavování matic a pro semisymbolickou analýzu. Je to dvourozměrné pole řetězců.

```
TStringArray(int AColCount, int ARowCount);
```

6.8 Metody třídy TStringArray

Třída obsahuje pouze dvě metody, a sice pro načtení (metoda *GetItem*) a zapsání (metoda *SetItem*) libovolné hodnoty v matici.

GetItem

Metoda vrací řetězec typu *ansistring* zapsaný na pozici dané vstupními parametry metody. První určuje sloupec a druhý řádek v matici.

```
AnsiString GetItem(int AColIndex, int ARowIndex);
```

SetItem

Metoda zapíše řetězec typu *ansistring* na pozici dané vstupními parametry metody. První určuje sloupec a druhý řádek v matici, třetí parametr pak obsahuje vkládaný řetězec.

```
void SetItem(int AColIndex, int ARowIndex, AnsiString NewItem);
```

6.9 Seznam uzlů v obvodu TEleNodeItem

V teorii byla naznačena teoretická páce s uzly. Ve skutečnosti je v programu vytvořen seznam veškerých uzlů, kde každému uzlu obvodu je přiřazen index. Tento index dále nahrazuje v seznamech jednotlivých součástek jména uzlů. Vzhledem k této substituci je pak možné provést veškerou redukci nezávislých branových veličin pouze prací s touto strukturou *TEleNodeItem*, aniž by program zpracovával další zbytečná data.

```
typedef struct TEleNodeItem { int ID; ShortString Name;
int ModAU; int ModAI; int ModBU; int ModBI; } TFEleNodeItem;
```

6.10 Práce programu s uzly obvodu

Nejprve je deklarován nový seznam uzlů, kde jednotlivé řádky odpovídají konkrétnímu uzlu obvodu. V každém řádku je zapsáno šest údajů:

1) pořadové číslo (index),
2) název uzlu,
3) napěťový koeficient uzlu ve fázi 1,
4) proudový koeficient uzlu ve fázi 1,
5) napěťový koeficient uzlu ve fázi 2,
6) proudový koeficient uzlu ve fázi 3.

Algoritmus plnění seznamu sestává z postupného prohledávání seznamu součástek a zapisování nových neznámých uzlů, přičemž každý řádek je vyplněn číselnými hodnotami názvu uzlu. Takto vytvořený seznam uzlů je připraven pro následující transformace.

6.10.1 Transformace působením spínačů

U tohoto typu transformace je využito skutečnosti, že napěťové i proudové koeficienty jsou shodné. Uvažují-li se ideální spínače s nulovým odporem v sepnutém stavu, pak při zjištění spojení dvou uzlů se v dané fázi nahradí napěťový a proudový koeficient u uzlu s vyšším číslem nižší číselnou hodnotou uzlu. Názorně tedy pokud je zjištěno spojení ve fázi 1 uzlů *x* a *y*, pak program provede následující úkony:

1) ze seznamu uzlů si pro uzly načte hodnoty a_x* a a_y**,
2) porovná koeficienty a_x* a a_y** mezi sebou
3) přepíše všechny koeficienty u uzlu s vyšší hodnotou koeficientu nižší hodnotou

4) projde ostatní a^* uzlů a pokud se shodují s přepisovaným uzlem a nahradí je novou hodnotou.

Tímto postupem je zajištěna správnost transformace i po vícenásobné transformaci.

6.10.2 Vynechání nepřipojených uzlů

Tato transformace přiřadí každému uzlu hodnoty koeficientů referenčního uzlu, pokud je v dané fázi uzel vlivem spínání izolován. Program tedy postupně prochází seznam uzlů a porovnává jejich spojení v dané fázi s pasivními a aktivními prvky obvodu. V případě nalezení prvního spojení uzlu se součástkou, program přechází na další uzel.

6.10.3 Transformace vlivem neregulárních prvků

V programu jsou začleněny dvě součástky, které redukují počet branových veličin. Jsou to tyto dva typy neregulárních prvků:

- ideální diferenční operační zesilovač (IDOZ)
- ideální zesilovač napětí s jednotkovým zesílením (BUFFER).

Uvažuje-li se ideální diferenční operační zesilovač se vstupními svorkami x a y a výstupní z, pak proběhnou následující transformace:

1) a^* a b^* vstupních svorek x a y se přepíše stejně jako u transformace spínačů,
2) a_z^{**} a b_z^{**} výstupní svorky se přepíše na 0.

Obsahuje-li obvod ideální zesilovač napětí s jednotkovým zesílením, který má vstupní svorku x a výstupní svorku y, pak proběhnou podobné transformace jako v předchozím případě:

1) a^* a b^* vstupní a výstupní svorky se přepíše stejně jako u transformace spínačů,
2) a_z^{**} a b_z^{**} výstupní svorky se přepíše na 0.

6.10.4 Vzestupné přečíslování koeficientů

Program prochází seznam uzlů a postupně přepisuje každý sloupec koeficientů tak, že prvnímu nenulovému koeficientu přiřadí hodnotu 1 a dalšímu novému přiřadí hodnotu o jedna větší. Vyskytuje-li se některý koeficient v matici vícekrát, je mu též přiřazena stejná hodnota jako prvnímu přepsanému.

6.11 Funkce pracující se strukturou TEleNodeItem

Výše zmíněná algoritmizace redukce branových veličin je provedena za pomocí následujících funkcí.

Count

Metoda vrací počet uzlů nalezených v obvodě. Tato hodnota je typu *integer*.

```
int Count(void);
```

Add

Metoda přidá uzel na konec seznamu uzlů a vrací jeho index.

```
int Add(TEleNodeItem Item);
```

CompareAdd

Metoda hledá uzel podle jména v seznamu postupným procházením seznamu a porovnáváním názvu uzlu s názvy uzlů v seznamu. Pokud uzel nenajde, přidá jej na konec seznamu. Funkce vrací pozici uzlu v seznamu (index).

```
int CompareAdd(ShortString Name);
```

GetItem

Metoda načte informace o uzlu zadaného indexem v seznamu.

```
TEleNodeItem GetItem(int Index);
```

SetItem

Metoda zapíše informace o uzlu zadaného indexem do seznamu.

```
void SetItem(int Index, TEleNodeItem);
```

IndexOfID

Metoda zjistí index uzlu podle názvu uzlu.

```
int IndexOfName(ShortString AName);
```

6.12 Seznamy součástek v obvodu

V teorii byl popsán způsob práce s jednotlivými součástkami v obvodu. Pro rozdělení součástek bylo vytvořeno několik struktur. Některé jsou pro určité součástky stejné, a tedy pro lepší orientaci ve zdrojovém kódu, je zde uveden seznam struktur s popisem, pro které součástky je daná struktura používána. Plus u každé struktury jsou přiloženy funkce pro zápis a načtení součástky. Zdrojový kód je v tomto případě velice intuitivní co do významu jednotlivých částí, a proto v této části nejsou rozepsány jednotlivé části struktur ani zde nejsou uvedeny popisy funkcí plnících/načítajících položky seznamu součástek.

Struktura pro práci s odpory a kondenzátory

```
typedef struct TElePartItem {
      ShortString Name; //Název součástky
      ShortString Value;      //hodnota součástky
      int Input;  //vstupní uzel
      int Output; //výstupní uzel
                                    } TFElePartItem;
int Count(void);  //počet součástek
int Add(TElePartItem Item);   //přidání součástky
TElePartItem GetItem(int Index);    //načtení součástky
void SetItem(int Index, TElePartItem Item);    //přepis součástky
```

Struktura pro práci s cívkami

```
typedef struct TElePartItemL {
      ShortString Name; //Název součástky
      ShortString ValueLs;    //hodnota L součástky
      ShortString ValueRs;    //hodnota R součástky
      int Input;  //vstupní uzel
      int Output; //výstupní uzel
                                    } TElePartItemL;
int Count(void);  //počet součástek
int Add(TElePartItemL Item);  //přidání součástky
TElePartItemL GetItem(int Index);   //načtení součástky
void SetItem(int Index, TElePartItemL Item);    //přepis součástky
```

Struktura pro práci s jednotkovými zesilovači (BUFFER)

```
typedef struct TElePartItemB {
      ShortString Name; //Název součástky
      int Input;  //vstupní uzel
      int Output; //výstupní uzel
                                    } TElePartItemB;
int Count(void);  //počet součástek
int Add(TElePartItemB Item);  //přidání součástky
TElePartItemB GetItem(int Index);  //načtení součástky
void SetItem(int Index, TElePartItemB Item);   //přepis součástky
```

Struktura pro práci se spínači

```
typedef struct TElePartItemS {
      ShortString Name; //Název součástky
      ShortString Value;           //fáze sepnutí součástky
      ShortString ValueR;          //hodnota R v sepnutém stavu
      int Input;  //vstupní uzel
      int Output; //výstupní uzel
                                    } TElePartItemS;
int Count(void);  //počet součástek
int Add(TElePartItemS Item);  //přidání součástky
TElePartItemS GetItem(int Index);  //načtení součástky
void SetItem(int Index, TElePartItemS Item);   //přepis součástky
```

Struktura pro práci se vzájemnou indukčností

```
typedef struct TElePartItemM {
      ShortString Name; //Název součástky
      ShortString ValueL1;         //hodnota L1
      ShortString ValueL2;         //hodnota L2
      ShortString ValueR1;         //hodnota R1
      ShortString ValueR2;         //hodnota R2
      int InputA;       //vstupní uzel A
      int OutputB;      //výstupní uzel B
      int InputC;       //vstupní uzel C
      int OutputD;      //výstupní uzel D
                                    } TElePartItemM;
```

```
int Count(void);  //počet součástek
int Add(TElePartItemM Item);     //přidání součástky
TElePartItemM GetItem(int Index);     //načtení součástky
void SetItem(int Index, TElePartItemM Item); //přepis součástky
```

Struktura pro práci s ideálními operačními zesilovači (IDOZ)

```
typedef struct TEleIDOZItem {
      ShortString Name; //Název součástky
      int Input;  //vstupní uzel
      int Output; //výstupní uzel
                                  } TEleIDOZItem;
int Count(void);  //počet součástek
int Add(TEleIDOZItem Item);   //přidání součástky
TEleIDOZItem GetItem(int Index);   //načtení součástky
void SetItem(int Index, TEleIDOZItem Item);    //přepis součástky
```

Struktura pro práci s operačními zesilovači

```
typedef struct TEleAOItem {
      ShortString Name; //Název součástky
      ShortString ValueA;          //hodnota A
      ShortString ValueFt;         //hodnota Ft
      ShortString ValueRout;       //hodnota Rout
      int InputA;       //vstupní uzel A
      int InputB;       //vstupní uzel B
      int Output;       //výstupní uzel
                                  } TEleAOItem;
int Count(void);  //počet součástek
int Add(TEleAOItem Item);    //přidání součástky
TEleAOItem GetItem(int Index);     //načtení součástky
void SetItem(int Index, TEleAOItem Item);//přepis součástky
```

6.13 Výpočet inverzní matice

Výpočet inverzní matice je vzhledem ke zvolenému způsobu řešení analýzy obvodů nejnáročnější na výpočetní čas. Tato funkce byla převzata z [6] stejně jako funkce *LUC* a *SOLVEC* a je u ní dbáno na maximální minimalizaci výpočetních úkonů. Zdrojový kód funkce je následující.

```
TComplexArray* InvMat(TComplexArray* y, int t, int deleni)
{
        int n = y->GetColCount();
        TComplexArray* indx = new TComplexArray(n,1);
        int d = 1;
        TComplexArray* col = new TComplexArray(n,1);
        TComplexArray* invmat = new TComplexArray(n,n);
        TComplexArray* matpom = new TComplexArray(n,n);
        for (int j=0;j<n;j++) {
          for(int k=0;k<n;k++) {
            matpom->SetItemReIm(k,j, y->GetItem(k,j).r, y->GetItem(k,j).i); } }
        matpom->Luc(matpom,indx,d);
        for (int j=0;j<n;j++)
        {
          col->NullAll();
          col->SetItemReIm(j,0,1.0,0.0);
          matpom->Solvec(matpom,indx,col);
          for (int i=0;i<n;i++)
          invmat->SetItem(j,i,col->GetItem(i,0));
        }
return invmat;
}
```

6.14 Algoritmus získání hodnot kmitočtových charakteristik

Algoritmus získání jednotlivých hodnot kmitočtových charakteristik lze rozdělit do několika hlavních bodů:

1) Ošetření smyslu výběru fáze vstupu a výstupu pro daný obvod.
2) Nulování řádků v „supermatici“.
3) Zápis velikosti napětí budícího zdroje do „supermatice“.
4) Výběr hodnoty z inverzní matice.

Jelikož je tento algoritmus značně obsáhlý a složitý, veškeré přípustné varianty a jejich řešení jsou zpracovány v tab. 3.

Tab. 3:Algoritmizace nulování řádků a získání výstupních hodnot.

Typ buzení	Fáze výstupního uzlu	Uzemněný vstupní uzel ve fázi 1	Uzemněný vstupní uzel ve fázi 2	Nulované řádky	Pozice jedničky (sloupec, řádek)	Pozice výsledku v inverzní matici (sloupec, řádek)
Fáze 1	Fáze 1	NE	ANO	In *a***	In *a**, In a**	In *a***, Out *a**
	Fáze 2	NE	ANO			In *a***, Out *b**
	Fáze 1	NE	NE	In *a***; In *b***	In *a**, In *a***; In *b**, In *b***	In *a***, Out *a**
	Fáze 2	NE	NE			In *a***, Out *b**
Fáze 2	Fáze 1	ANO	NE	In *b***	In *b**, In *b***	In *b***, Out *a**
	Fáze 2	ANO	NE			In *b***, Out *b**
	Fáze 1	NE	NE	In *a***; In *b***	In *a**, In *a***; In *b**, In *b***	In *b***, Out *a**
	Fáze 2	NE	NE			In *b***, Out *b**
Fáze 1 a 2	Fáze 1	ANO	NE	In *b***	In *b**, In *b***	In *b***, Out *a**
	Fáze 2	ANO	NE			In *b***, Out *b**
	Fáze 1	NE	ANO	In *a***	In *a**, In *a***	In *a***, Out *a**
	Fáze 2	NE	ANO			In *a***, Out *b**
	Fáze 1	NE	NE	In *a***; In *b***	In *a**, In *a***; In *b**, In *b***	In *a***, Out *a** + In *b***, Out *a**
	Fáze 2	NE	NE			In *a***, Out *b** + In *b***, Out *b**

*a** napěťový koeficient v 1. spínací fázi,

*a*** proudový koeficient v 1. spínací fázi,

*b** napěťový koeficient ve 2. spínací fázi,

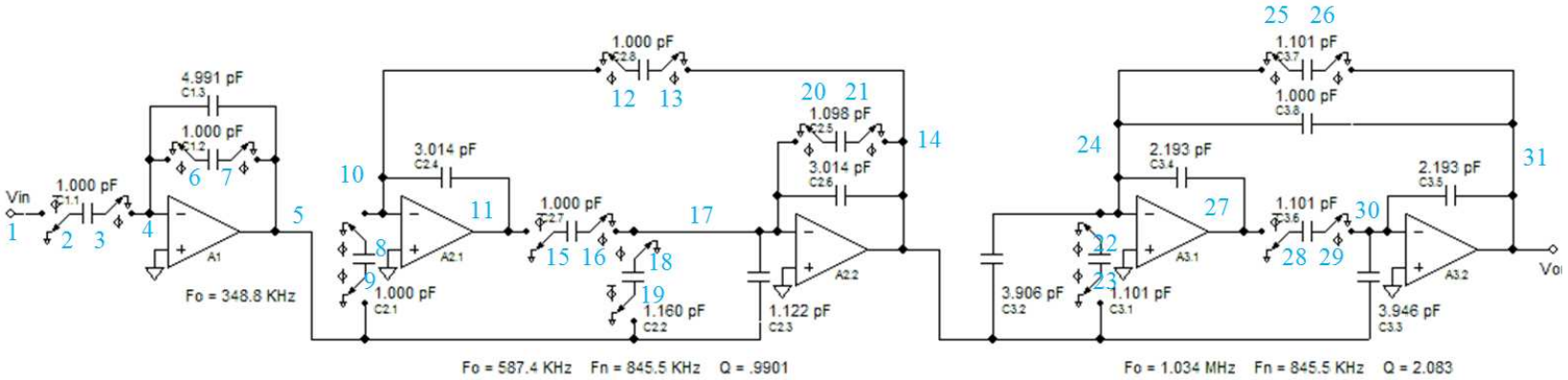
*b*** proudový koeficient v 2. spínací fázi.

7 Praktické ukázky programu

V této části práce je uveden podrobný výpis jednotlivých fází analýzy v programu. Jako vzorový příklad byl zvolen obvod Chebyshevova filtru typu dolní propust 5. řádu na obr. 15. Nejprve musí být pro tento obvod vytvořen popisný soubor obvodu. Z netlistu je patrné, že obvod lze rozdělit do tří hlavních částí a že jsou uvažovány ideální spínače a reálné operační zesilovače s jedním kmitočtem lomu.

NetList:

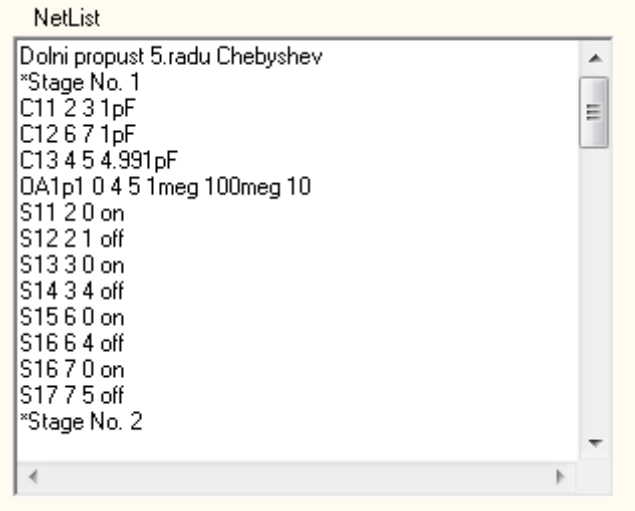
```
Dolni propust 5.radu Chebyshev
*Stage No. 1
C11 2 3 1pF
C12 6 7 1pF
C13 4 5 4.991pF
OA1p1 0 4 5 1meg 100meg 10
S11 2 0 on
S12 2 1 off
S13 3 0 on
S14 3 4 off
S15 6 0 on
S16 6 4 off
S16 7 0 on
S17 7 5 off
*Stage No. 2
C21 8 9 1pF
C22 18 19 1.16pF
C23 5 17 1.122pF
C24 10 11 3.014pF
C25 20 21 1.098pF
C26 17 14 3.014pF
C27 15 16 1pF
C28 12 13 1pF
OA1p21 0 10 11 1meg 100meg 10
OA1p22 0 17 14 1meg 100meg 10
S21 9 0 on
S22 9 5 off
S23 8 0 on
S24 8 10 off
S25 15 0 on
S26 15 11 off
S27 16 0 on
S28 16 17 0ff
S29 18 0 on
S210 18 17 off
S211 19 0 on
S212 19 5 off
S213 12 0 on
S214 12 10 off
S215 13 0 on
S216 13 14 0ff
S217 20 0 on
S218 20 17 off
S219 21 0 on
S220 21 14 off
*Stage No. 3
C31 22 23 1.101pF
C32 14 24 3.906pF
C33 14 30 3.946pF
C34 24 27 2.193pF
C35 30 31 2.193pF
C36 28 29 1.101pF
C37 25 26 1.101pF
C38 24 31 1pF
OA1p31 0 24 27 1meg 100meg 10
OA1p32 0 30 31 1meg 100meg 10
S31 23 0 on
S32 23 14 off
S33 22 0 on
S34 22 24 off
S35 28 0 on
S36 28 27 off
S37 29 0 on
S38 29 30 off
S39 25 0 on
S310 25 24 off
S311 26 0 on
S312 26 31 off
```



Obr. 15: Dolní propust 5. řádu Chebyshev [7].

Po zadání výše zmíněného netlistu do programu, obr. 16 a po stisknutí tlačítka „Roztřiď“, program vyplní příslušné seznamy (obr. 17, 18, 19, 20). Na každém obrázku je vidět, o jakou součástku se jedná a je zde uvedena i informace o počtu výskytu součástky v obvodu.

Obrázek 20 ukazuje speciální pole, které obsahuje nerozeznané syntaxe v popisném soubor obvodu. V tomto případě obsah nerozpoznal řádky rozčleňující netlist do částí podle hlavních bloků obvodu.



Obr. 16: Výstřižek z okna programu obsahující popisný soubor obvodu.

Kondenzátory C — Celkem: 19

Název součástky	Vstup	Výstup	Hodnota	Komentář
C11	2	3	1e-12	
C12	6	7	1e-12	

Obr. 17: Výstřižek z okna programu obsahující seznam všech cívek v obvodu.

Operační zesilovače OA1p — Celkem: 5

Název součástky	in+	in-	out	A	ft	Rout	Komentář
OA1p1	0	4	5	1e+6	100e+6	10	
OA1p21	0	10	11	1e+6	100e+6	10	

Obr. 18: Výstřižek z okna programu obsahující seznam všech operačních zesilovačů v obvodu.

Spínače S — Celkem: 40

Název součástky	Vstup	Výstup	1/2	Odpor v sepnutém stavu	Komentář
S11	2	0	on		
S12	2	1	off		

Obr. 19: Výstřižek z okna programu obsahující seznam všech spínačů v obvodu.

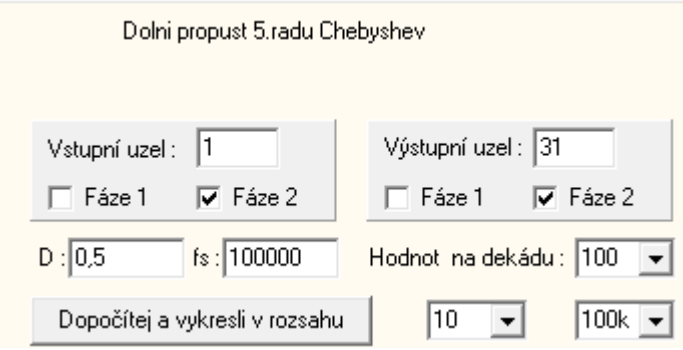
Obr. 20: Výstřižek z okna programu obsahující nerozpoznané syntaxe popisného souboru obvodu.

Dalším kliknutím tentokrát na tlačítko „Transformuj“, program provede hledání nezávislých branových veličin, včetně vzestupného přečíslování uzlů. Na obr. 21 je vidět výsek ze skryté záložky „Nezávislé uzly obvodu“ v programu, kde jsou rozepsány veškeré potřebné věci pro redukci počtu uzlů. Na zvýrazněném řádku na obrázku je vidět, že obvod nejvyšší číslo uzlu je 31 a že vlivem algoritmů pro redukci nezávislých branových veličin bude mít admitanční matice pro fázi 1 rozměr 10 x 10 a pro fázi 2 rozměr 11 x 11. Tedy časová úspora výpočtu je v tomto příkladě více než 60%.

ID	Uzel	U fáze 1	I fáze 1	U fáze 2	I fáze 2
29	25	0	0	10	10
30	26	0	0	11	11
25	27	8	8	12	12
27	28	0	0	12	12
28	29	0	0	13	13
2	3	0	0	2	2
24	30	9	9	13	13
26	31	10	10	11	11
5	4	1	1	2	2
6	5	2	2	3	3
3	6	0	0	2	2
4	7	0	0	3	3
7	8	0	0	4	4

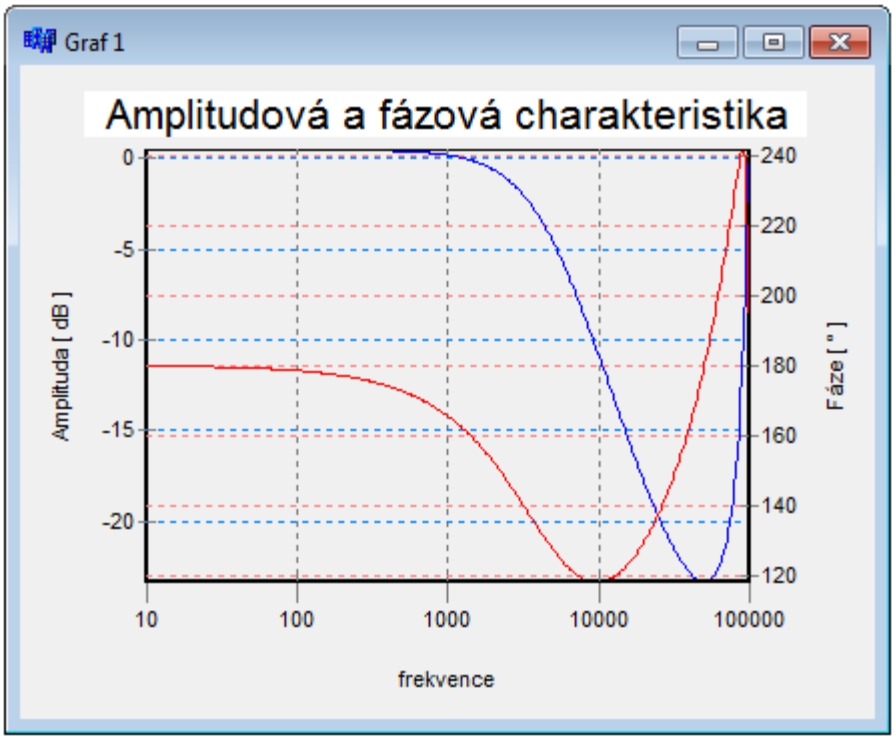
Obr. 21: Výstřižek ze skryté záložky programu obsahující finální stav uzlů obvodu po algoritmech hledajících nezávislé branové veličiny.

Po stisku tlačítka „Obvodové matice“ je vytvořena „supermatice“ pro analyzovaný obvod, která vzhledem ke své velikosti zde nebude uvedena a v této ukázce se přejde přímo na zadání posledních parametrů analýzy, jak je možné vidět na obr. 22.

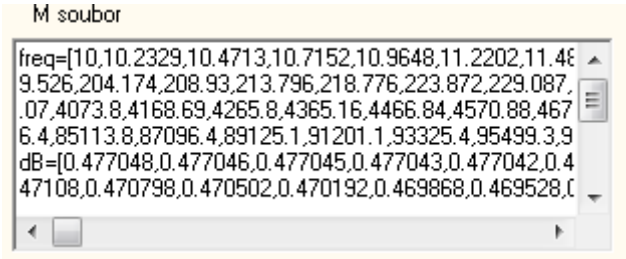


Obr. 22: Výstřižek okna programu obsahující pole pro zadání nebo výběr parametrů analýzy obvodu.

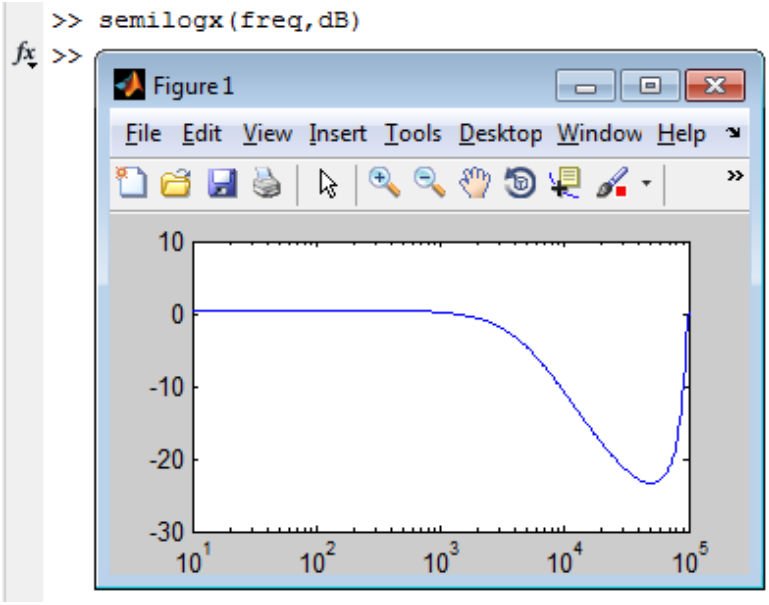
Po zadání parametrů a každém stisku tlačítka „Dopočítej a vykresli v rozsahu“ je otevřeno nové okno (obr. 23), které obsahuje kmitočtové charakteristiky podle zadaných parametrů na obr. 22. Je také vyplněn blok pole „M soubor“ (obr. 24), kde je náhled matic výsledných hodnot analýzy, které lze odtud přímo nakopírovat do programu MATLAB, kde příkazem *semilogx(freq,dB)* vykreslíme semilogaritmický graf amplitudové charakteristiky (obr. 25) a příkazem *semilogx(freq,ph)* graf fázové charakteristiky (obr. 26).



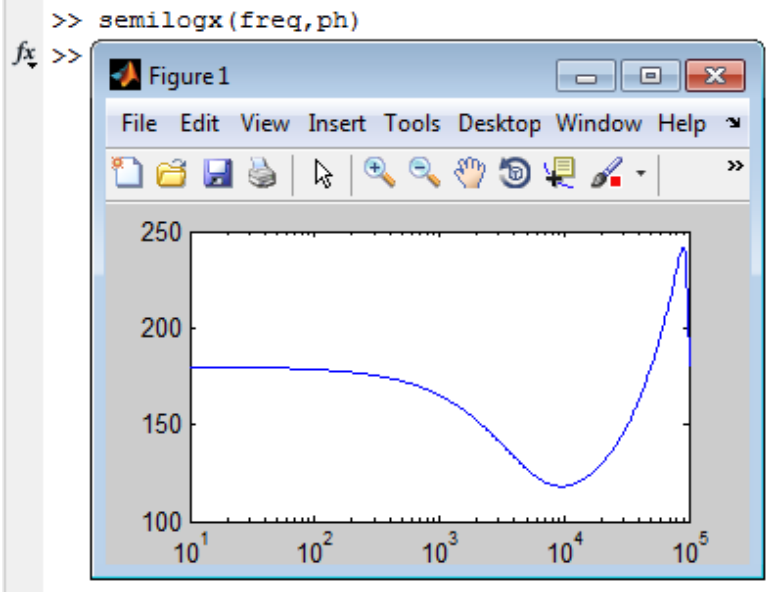
Obr. 23: Okno s kmitočtovou charakteristikou obvodu.



Obr. 24: Výstřižek okna programu obsahující náhled výstupního M souboru.



Obr. 25: Výstřižek okna programu MATLAB obsahující amplitudovou charakteristiku.



Obr. 26: Výstřižek okna programu MATLAB obsahující fázovou charakteristiku.

8 Závěr

Diplomová práce shrnuje a završuje znalosti nabyté při tvorbě bakalářské práce a semestrálních projektů. Jejím cílem bylo vytvořit program pro analýzu filtrů se spínanými kapacitory. Ačkoliv podobné simulační programy v dnešní době existují, lze jimi řešit pouze idealizované obvody obsahující ideální spínače s nulovým odporem v sepnutém stavu, kapacitory a ideální operační zesilovače. Výstupem těchto programů je obvykle kmitočtová charakteristika obvodů, která se počítá z jeho systémové funkce v oblasti z.

První část práce upřesňuje tvorbu popisného souboru obvodu, tzv. netlistu. Tento soubor je vstupem do programu CIRNAM. Je tedy nutné dodržet syntaxe zápisu dle uvedené teorie, neboť vzhledem k uvažování reálných spínačů a součástek bylo nutné některé syntaxe zápisu modifikovat nebo dokonce vytvořit nové, a tudíž nekorespondují se zápisem, na který mohou být uživatelé tohoto programu zvyklí z programů typu PSpice.

V další krátké kapitole „Osnova řešení zpracování dat“ je popsán způsob, jakým se v této práci ubírá analýza obvodů, u nichž jsou uvažovány dvě spínací fáze s rozdílnou dobou trvání, které na sebe navzájem navazují a opakují se. Je zde také definováno hlavní rozdělení postupu počítačové simulace těchto obvodů.

Čtvrtá obsáhlá kapitola „Sestavení obvodových rovnic“ obsahuje teorii přípravy vstupních matic pro hlavní algoritmy programu pro počítačovou analýzu obvodů. Podrobně je zpracována teorie zabývající se hledáním nezávislých obvodových veličin a sestavení admitančních a přechodových matic obvodu, které jsou submaticemi „supermatice“. Ke konci kapitoly jsou uvedeny možnosti buzení obvodu a veškerá popsaná teorie je ukázána na vzorovém příkladě obvodu spínaného filtru typu dolní propusti.

Kapitola „Program CIRNAM“ je pro uživatele programu nejzajímavější. Ačkoliv je práce v programu velmi intuitivní, je zde popsána práce v programu a možnosti ukládání vstupních a výstupních dat.

Jelikož na vývoji podobných aplikací pracují většinou celé týmy programátorů a ne pouze jeden člověk, jsou možnosti tohoto programu z pohledu analýzy obvodu se spínanými kapacitory pouze v základním rozsahu. Algoritmy a členění programu jsou však koncipovány v mnohem širším záběru a je tedy možné program případně rozšiřovat. Pro tyto účely je v práci zpracován popis implementace teorie do programu, kde případný programátor nalezne popis struktur, metod a stěžejních funkcí programu. Na konci této kapitoly je tabulka,

která přehledně shrnuje algoritmus získání výsledných hodnot ze „supermatice“ podle typu buzení obvodu, které jsou ve stávající verzi programu zakomponovány.

Při ověřování správné funkce výpočtu programu byly jednak testovány jednotlivé metody struktur a funkce programu a také proběhla simulace různých obvodů se spínanými kapacitory. Jako příklad funkčnosti obvodu byl použit obvod typu dolní propust 5. řádu.

V této práci byla tedy popsána teorie počítačové analýzy spínaných obvodů využitá ke tvorbě vlastního programu CIRNAM v programovacím jazyce C++, jehož výstupem jsou kmitočtové charakteristiky. Program vznikl za velkého příspěvku pana prof. Dalibora Biolka, který udal směr tvorby programu CIRNAM a k tomu dodal odpovídající teoretickou základnu, čímž se otevírá možnost dalšího rozšíření programu vytvořeného v rámci této diplomové práce.

9 Použitá literatura

[1] BIOLEK, D. Modelování a simulace v mikroelektronice. Elektronické učební texty, 136 stran, ÚMEL FEKT VUT Brno, 2005.

[2] BIOLEK, D. Modeling of Periodically Switched Networks by Mixed s-z Description. IEEE Trans. on CAS-I, Vol. 44, No. 8, August 1997, pp. 750-758.

[3] BIOLEK, D. Počítačová analýza reálných spínaných obvodů. Dílčí výzkumná zpráva úkolu "Syntetické prvky vyššího řádu" Grantové agentury ČR, č. 102/93/2079, ÚTEL FEI Brno, 1994.

[4] PECH, V. *Program pro analýzu filtrů-bakalářská práce*. Brno, 2008. 42 s. Vedoucí bakalářské práce prof. Ing. Dalibor Biolek, CSc. FEKT VUT v Brně.

[5] Základní dokumentace k MATLABu na www.humusoft.cz

[6] William H. Press, Numerical recipes in C: The art of scientific computing, Vol. 2, Cambridge university press, 1992, 994 stran, ISBN 0-521-43108-5.

[7] Obvod vzorového příkladu na http://www.filter-solutions.com/switched.html