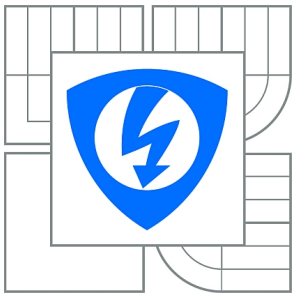

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY


FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS


# ANALÝZA AUTONOMNÍCH OBVODŮ POUŽITÍM PROGRAMU MATLAB

AUTONOMOUS CIRCUIT ANALYSIS USING MATLAB

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS


AUTOR PRÁCE — LADISLAV ZMEŠKAL

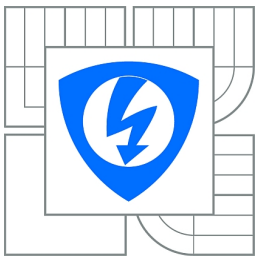
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV KOTON, Ph.D.

SUPERVISOR



BRNO 2013



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav telekomunikací**

# Bakalářská práce

bakalářský studijní obor
**Teleinformatika**

***Student:*** Ladislav Zmeškal ***ID:*** 134674
***Ročník:*** 3 ***Akademický rok:*** 2012/2013

**NÁZEV TÉMATU:**

## Analýza autonomních obvodů použitím programu Matlab

**POKYNY PRO VYPRACOVÁNÍ:**

Na základě vhodného popisu obecného autonomního obvodu s proudovými a napěťovými konvejory sestrojte jeho admitanční matici. Využitím teorie obvodů analyzujte jeho chování, zejména pak napěťové, proudové a smíšené přenosy. Celý postup analýzy algoritmizujte v programu Matlab. Na základě získaných výstupů vybrané zapojení podrobně simulacím a potvrďte tak správnost navrženého algoritmu.

**DOPORUČENÁ LITERATURA:**

[1] Doňar, B., Zaplatílek, K.: „MATLAB pro začátečníky“, BEN – technická literatura, 2003, ISBN 80-7300-175-6.
[2] Doňar, B., Zaplatílek, K.: „MATLAB – tvorba uživatelských aplikací“, BEN – technická literatura, 2004, ISBN: 80-7300-133-0.

***Termín zadání:*** 11.2.2013 ***Termín odevzdání:*** 5.6.2013

***Vedoucí práce:*** Ing. Jaroslav Koton, Ph.D.
***Konzultanti bakalářské práce:***

**prof. Ing. Kamil Vrba, CSc.**
*Předseda oborové rady*

**UPOZORNĚNÍ:**

# Abstrakt

Tato bakalářská práce se bude zabývat vytvořením a popisem programu pro symbolický výpočet přenosových funkcí obvodu, pro obvody, sestávající se z obecných proudových konvejorů a admitancí. V teoretické části budou popsány algoritmické metody analýzy obvodu společně s charakteristikou obecného proudového konvejoru a stručný popis programu MATLAB. Ve druhé části bude popsána funkce vytvořeného programu, jeho jednotlivé části, a konkrétní příklady výpočtu přenosových funkcí. V poslední části je ověřována správnost výsledků vytvořeného programu v MATLABu s výsledky z programu SNAP.

# Klíčová slova

MATLAB, obecný proudový konvejor, přenosové funkce

# Abstract

This bachelor's thesis will deal with the creation of a description of a program for symbolic calculation of transfer function circuit, circuits, consisting of base current conveyors and admittance. In the theoretical part describes algorithmic methods analysis circuit together with the general characteristics of the current conveyor and a brief description of the program Matlab. The second part will describe the function created program, its parts, and specific examples of calculation of transfer functions. The last part of the validation of the results program created in MATLAB with the results of the SNAP program.

# Keywords

MATLAB, general current conveyor, transfer functions

ZMEŠKAL, L. *Analýza autonomních obvodů použitím programu Matlab.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 42 s. Vedoucí bakalářské práce Ing. Jaroslav Koton, Ph.D..

# Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Analýza autonomních obvodů použitím programu MATLAB jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009Sb.

V Brně dne …….. ……..

podpis autora

# Poděkování

Děkuji vedoucímu práce Ing. Jaroslavu Kotonovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne …….. ……..

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007-13
OP Výzkum a vývoj pro inovace

# Obsah

# Seznam obrázků

# Úvod

Analýza chování obvodů hraje významnou roli při návrhu elektronických obvodů. Druhy analýz, jejich rozdělení, vyjádření a možnosti použití metod budou popsány v první kapitole. Bude zde uvedena i matematická charakteristika a popis obecného proudového konvejoru.

Ve druhé kapitole bude blíže popsán program MATLAB. Základní vlastnosti, kterými se MATLAB vyznačuje, jeho použití a základní přehled funkcí.

Třetí kapitola bude věnována popisu vytvořeného program pro analýzu chování obvodů, jeho jednotlivých částí a provádění výpočtů využitím teorie z první kapitoly.

Poslední kapitola bude zaměřena na porovnání správnosti výsledků, pro každou přenosovou funkci, s výsledky z programu SNAP.

# 1 Analýza obvodů

Analýzou (řešením) elektrických obvodů se rozumí zjišťování vlastností této soustavy, jejíž struktura, tj. způsob propojení jednotlivých dílčích prvků, i jejich parametry jsou dány [2].

## 1.1 Analýza

Analýzu provádíme ve snaze získat informace o určitých vlastnostech zkoumaného obvodu. Z praktických důvodů však zpravidla analýze nepodrobujeme samotný obvod, nýbrž jeho model. Jedním z dobrých důvodů může být skutečnost, že daný obvod dosud neexistuje a před jeho výrobou je vhodné ověřit, zda je navržen správně. Z hlediska matematického je model obvodu představován soustavou rovnic, které lze odvodit na základě rovnic dílčích elektrických prvků a Kirchhoffových rovnic, které reprezentují způsob zapojení součástek [1].

Všechny existující metody analýzy můžeme rozdělit na nealgoritmické (heuristické) a algoritmické. Při použití algoritmické metody dokáže řešitel obvod vyřešit prostým pochopením algoritmu a nemusí znát fyzikální podstatu. Algoritmické metody jsou určeny především pro počítačové řešení obvodů. Využívají se v simulačních a analyzačních programech. Mezi nejznámější algoritmické metody patří např. metoda smyčkových proudů (MSP), metoda uzlových napětí (MUN) nebo modifikovaná metoda uzlových napětí. Naopak při analýze heuristickou metodou volí řešitel postupy na základě svých předchozích zkušeností s využitím tvůrčího přístupu. Mezi tyto metody patří např. Théveninův a Nortonův teorém, princip superpozic, substituce nebo ekvivalence, aj [1].

Jelikož je analýza pouze věcí matematiky, vznikly aplikace jako např. Spice, WinSpice, OrCad, MicroCap, Electronic Workbench, SwitcherCAD, jejichž matematickým základem jsou numerické metody řešení rozsáhlých soustav rovnic. Objevují se ovšem i programy, které pracují na tzv. symbolickém principu. Termín symbolický označuje, že ve výsledku jsou přítomny symboly parametrů prvků obvodu. Nevyskytují se zde skutečné, číselné hodnoty parametrů. Z programů, které jsou založeny na symbolických algoritmech, jmenujme např. program SNAP [1]. Pro tyto účely je možné použít i univerzální matematický program, který umí pracovat se symbolickými výpočty, třeba MATLAB. Uživatel tak má možnost vytvořit si vlastní aplikaci, využívající např. algoritmickou metodu MUN, i s grafickým zobrazením. A právě algoritmická metoda MUN bude popsána v další podkapitole.

## 1.2 Metoda uzlových napětí

Metoda vychází z 1. Kirchhofova zákona (KZ), kdy pro obvod, který má $n$ uzlů, můžeme napsat $n - 1$ rovnic právě podle 1.KZ.

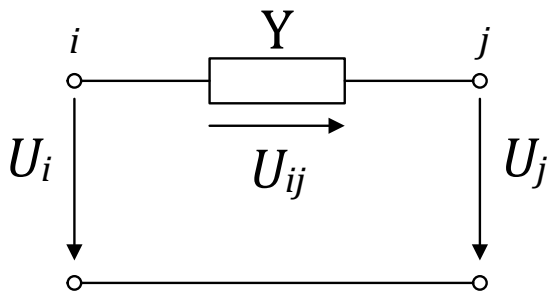
Libovolný jeden uzel v obvodu neuvažujeme. Tento uzel je vztažný, resp. referenční. Je výhodné volit za referenční ten uzel, k němuž je připojeno nejvíce větví [7], např. zem. Pro aplikaci základních zákonů elektrických obvodů je nutné určit počet nezávislých uzlů ($p$). Každému nezávislému uzlu se přisoudí nezávislé uzlové napětí a sestaví se odpovídající počet nezávislých rovnic podle 1. (KZ). Standardní maticový tvar těchto rovnic je pak

$$\mathrm{I} = \mathrm{YU}, \tag{1.1}$$

kde $\mathrm{I} \in \Re^p$ je vektor známých uzlových proudů, $\mathrm{U} \in \Re^p$ je vektor neznámých uzlových napětí a $\mathrm{Y} \in \Re^{p \times p}$ je čtvercová matice admitančních koeficientů, tzv. admitanční matice [2].

### 1.2.1 Vytvoření admitanční matice

U této metody je možné admitanční matice Y sestavit přímo podle zapojení obvodu. Prvek *i*,*i* na hlavní diagonále obsahuje součet všech admitancí, které jsou připojeny k uzlu *i*. Prvek *i*,*j* mimo hlavní diagonálu obsahuje záporně vzatý součet všech admitancí, které jsou připojeny bezprostředně mezi uzly $i$ a $j$ [1], neboť napětí mezi nimi je u souhlasně orientované soustavy dáno vždy rozdílem příslušných uzlových napětí, tj. $U_{ij} = U_i - U_j$ (Obr. 1.1) [2].




**Obr. 1.1: K definici vzájemné admitance u souhlasně orientovaných soustav [2]**

Při sestavení admitanční matice, kde je referenční uzel vně obvodu, vznikne tzv. úplná admitanční matice. Její zvláštností je, že sečteme-li všechny prvky v libovolném řádku nebo sloupci, dostaneme nulu. Pokud dodatečně prohlásíme za referenční uzel některý z uzlů v obvodu, řekněme uzel *k*, získáme příslušnou admitanční matici tak, že z úplně admitanční matice vypustíme *k*-tý řádek a *k*-tý sloupec.

### 1.2.2 Výpočet základních obvodových funkcí z admitanční matice

Mějme lineární obvod o $N$ uzlech vyjma referenčního uzlu, který je popsán admitanční maticí $N$ x $N$ pomocí metody uzlových napětí. Pomocí algebraických doplňků této matice můžeme spočítat:

**Uzlové napětí** $U_k$, je-li obvod napájen z jediného zdroje proudu $I_i$ zapojeného mezi uzel $i$ a referenční uzel: [1]

$$U_k = \frac{\Delta_{i:k}}{\Delta} I_i. \tag{1.2}$$

**Impedanci** mezi uzlem $k$ a referenčním uzlem:

$$Z_k = \frac{\Delta_{k:k}}{\Delta}. \tag{1.3}$$

**Přenos napětí** z uzlu $i$ do uzlu $o$:

$$K_u = \frac{U_o}{U_i} = \frac{\Delta_{i:o}}{\Delta_{i:i}}. \tag{1.4}$$

**Přenosová impedance**, tj. poměr výstupního napětí a vstupního proudu vychází z (1.2) [2]

$$Z_t = \frac{1}{Y_t} = \left(\frac{U_o}{I_i}\right) = \frac{\Delta_{i:o}}{\Delta}, i \neq o. \tag{1.5}$$

Ve všech vzorcích je $\Delta$ determinant admitanční matice a $\Delta_{i;k}$ je její algebraický doplněk při vynechání $i$-tého řádku a $k$-tého sloupce. Číselně se rovná vzniklému subdeterminantu matice násobenému číslem $(-1)^{i+k}$ [1].

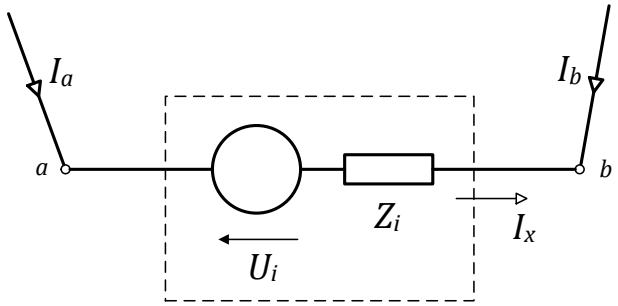
## 1.3 Modifikovaná metoda uzlových napětí

Nevýhodou metody uzlových napětí je, že neumožňuje analyzovat obvody se zdroji napětí a součástkami, které nemají admitanční matici (transformátor, operační zesilovače, konvejory). Proto vznikly její modifikace, které zachovávají výhody MUN, ale umožní i analyzovat lineární obvody bez výše uvedených omezení [1].

### 1.3.1 Metoda razítek

Každý „problémový“ prvek je popsán minimálně jednou přídavnou rovnicí a o stejný počet obohatí množinu neznámých. Současně dojde k modifikaci některých původních rovnic 1. KZ. Maticová rovnice pak získá zvláštní strukturu: k původní admitanční matici MUN přibydou řádky a sloupce, jejichž prvky obecně nemají rozměr admitancí. Jsou to tzv. razítka přídavných elektrických prvků. Celá matice se pak nazývá pseudoadmitanční [1].

Uvažujme obvod popsaný rovnicemi klasické MUN. Mezi uzly *a* a *b* obvodu dodatečně připojíme obecný dvojpól, který je popsán Théveninovým modelem podle (Obr. 1.2). Včleněním dvojpólu dojde ke změně napěťových a proudových poměrů v obvodu. Dvojpólem bude protékat proud $I_x$, který modifikuje proudové poměry v uzlech *a* a *b*. Dojde i ke změně původních uzlových napětí [1].




**Obr. 1.2: Začlenění obecného lineárního dvojpólu, popsaného Théveninovým modelem, do obvodu [1]**

Původní rovnice popisující rovnováhu proudů v uzlu *a* musí být na pravé straně doplněna o proud $I_x$, vytékající ven z uzlu, a v uzlu *b* o proud $I_x$ se záporným znaménkem, protože vtéká dovnitř uzlu *b*. Navíc uzlová napětí $U_a$ a $U_b$ jsou nyní spolu vázána podmínkou

$$Z_i I_x + U_b = U_i + U_a. \tag{1.6}$$

Všechny tyto modifikace lze zahrnout do nové soustavy matice MMUN [1], z které je pak možné spočítat admitanční a proudový přenos.

**Přenos admitance** je dán poměrem výstupního proudu a vstupního napětí

$$Y_t = \frac{1}{Z_t} = \left(\frac{I_o}{U_i}\right) = \frac{\Delta_{i:o}}{\Delta_{i:i}}, i \neq o. \tag{1.7}$$

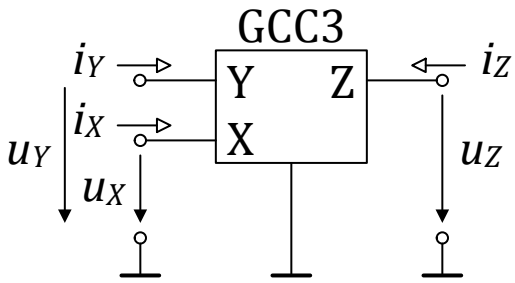
Pro výpočet **proudového přenosu** se vychází z rovnice (1.2), její úpravou dostaneme

$$K_i = \left(\frac{I_o}{I_i}\right) = Y_o \frac{\Delta_{i:o}}{\Delta}. \tag{1.8}$$

## 1.4 Obecný proudový konvejor

Mezi prvky, které nemají admitanční matici, patří i konvejor, který se používá např. při návrhu kmitočtových filtrů.

Aby bylo možné již při návrhu nových elektronických obvodů uvažovat co nejvíce typů proudových konvejorů (až dvanáct základních tříbranových konvejorů [6].) byl navrhnut zobecněný proudový konvejor (General Currend Conveyor - GCC) [8], jehož schématická značka je na Obr. 1.3.




**Obr. 1.3: Schématická značka aktivního tříbranového prvku GCC3 [5]**

GCC je možné popsat obecnými rovnicemi [8]

$$u_X = vo.u_Y, i_Y = cu1.i_X, i_Z = cu2.i_X, \qquad (1.9a, b, c)$$

nebo obecnou hybridní maticí [5]

$$\begin{bmatrix} u_X \\ i_Y \\ i_Z \end{bmatrix} = \begin{bmatrix} 0 & vo & 0 \\ cu1 & 0 & 0 \\ cu2 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_X \\ u_Y \\ u_Z \end{bmatrix}, \qquad (1.10)$$

kde *vo* je napěťový přenos mezi bránou *Y* a *X*, *cu1* proudový přenos mezi bránou *X* a *Y* a *cu2* je proudový přenos mezi bránou *X* a *Z*. Brány jsou nazývány následovně:

- X – proudová brána
- Y – napěťová brána
- Z – výstupní brána

# 2 MATLAB

MATLAB je integrovaným prostředím, s jehož pomocí lze provádět matematické výpočty, modelování, analýzu a vizualizaci dat, měření a zpracování dat, vývoj algoritmů, návrhy řídících a komunikačních systémů a mnohem další [3].

Základem je výpočetní jádro, provádějící numerické operace s maticemi reálných či komplexních čísel. MATLAB je tedy orientován maticově. Kromě matic podporuje MATLAB tzv. pole buněk. Jde o struktury podobné maticím. Na rozdíl od nich však každý prvek může být jiného typu. V prostředí MATLAB lze efektivně pracovat s vektory, které mohou představovat časové řady, signály různých typů atd. Grafický subsystém umožňuje snadné zobrazení výsledků výpočtů. MATLAB obsahuje plnohodnotný programovací jazyk čtvrté generace. Uživatel nalezne vše potřebné k programování a ladění zdrojových kódů. Systém navíc nabízí vestavěnou podporu tvorby grafických prvků (tlačítek, zatržítek, menu) a podporu, usnadňující načítání dat z jiných zdrojů [3].

MATLAB podporuje i tzv. toolboxy. Jsou to knihovny funkcí, které významně rozšiřují možnosti jádra MATLABu. Jsou orientovány na konkrétní vědní a technické obory. Uživatel si je může dokoupit jako přídavné moduly. Existují tak toolboxy pro zpracování signálů a obrazů, pro práci s neuronovými sítěmi, pro návrh filtrů, finance a ekonomiku aj [3].

## 2.1 Popis pracovního prostředí

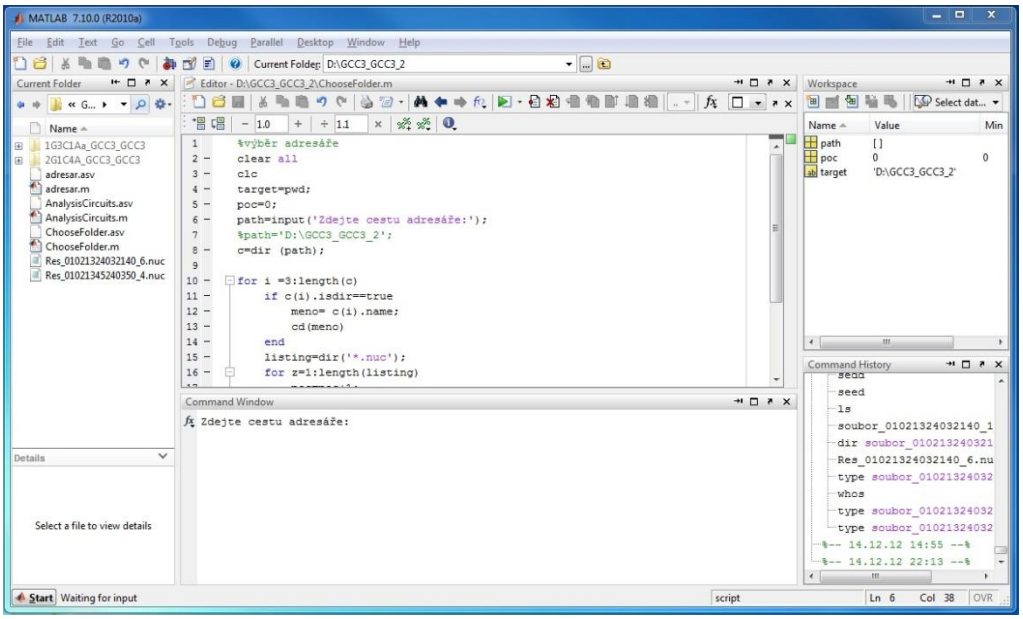
Prostředí MATLABu obsahuje několik základních oken. Standartní rozvržení oken v prostřední MATLABu pro verzi 7.10.0.499 (R2010a) je znázorněno na Obr. 2.1.

Okno Command Window - je hlavní a nejdůležitější částí pracovní plochy. Sem zapisuje uživatel své příkazy a povely, zde je vidět odezva MATLABu a zde se zobrazují systémová hlášení.

Okno Command History - je velmi užitečným oknem. Zobrazují se zde všechny příkazy a povely, zapsané a potvrzené uživatelem v hlavním okně Command Window. Je-li potřeba již jednou zapsaný a potvrzený příkaz znovu použít, stačí jej v tomto okně nalistovat a poklepáním znovu aktivovat či jej přetáhnout myší do hlavního okna.

Okno Workspace. Při používání proměnných v hlavním okně bude v tomto okénku přehled všech vámi použitých proměnných. Poklepnutím na symbol některé proměnné, zobrazí se detailní informace o ní (rozměr, struktura apod.).

Okno Current Directory ukazuje seznam souborů v aktuální (current) složce (adresáři). Poklepáním myší na některý ze zobrazených souborů jej otevřete ve vestavěném editoru [3].




**Obr. 2.1: Pracovní prostředí MATLABu**

## 2.2 Základní přehled funkcí

MATLAB obsahuje mnoho funkcí. Ty nejzákladnější včetně příkladu použití a jejich významu viz příloha A. Patří mezi ně používání proměnných, znakových řetězců, vektorů a matic.

## 2.3 Práce s m-soubory

Tzv. m-soubory jsou textové soubory s příponou *.*m*. Slouží k zápisu posloupnosti příkazů MATLAB a jejich uložení na disk. Jsou tedy zdrojovým kódem, který umí MATLAB vykonat [3]. Vytvořit m-soubor lze v libovolném textovém editoru, který nepřidává žádné informace typu formátování a který vám umožní zapsat si vlastní příponu *.m. M-soubory můžeme rozdělit na dva základní typy, scripty a funkce [4].

Scripty označují jednoduché m-soubory, které obsahují příkazy MATLABu. Jejich označení vychází z angličtiny. Jednoduše napsáno, neumí to, co umí funkce. Funkce jsou také m-soubory. Na rozdíl od skriptů však mohou být volány s jedním nebo více vstupními parametry a mohou předávat jeden nebo více výstupních parametrů. To je důležité např. tehdy, když v rámci jedeno m-souboru voláme jiný a jeho činnost se projeví tím, předá své výsledky (parametry) [4].

# 3 Popis vstupních dat a programu

Program je navržen pro symbolický výpočet základních přenosových funkcí obvodů, které obsahují admitanční prvky a aktivní prvky GCC3. V této kapitole bude dále popsán formát vstupních dat a jednotlivé funkce s výpisem kódu, který je důležitý pro celkové pochopení, jak je program navržen a jak pracuje.

Program se skládá celkem z pěti funkcí a jednoho scriptu. Scriptem, nazvaným AnalysisCircuit, se celý proces analýzy spouští. Zahrnuje v sobě mimo jiné i spouštění čtyř funkcí pro analýzu a funkce *save_file* pro uložení výsledku do souboru. Funkce *save_file* je podrobněji popsána v podkapitole 3.2. Zbývající funkce jsou nazvány následovně:

- Funkce pro napěťový přenos – *voltage_transfer*.
- Funkce pro proudový přenos – *current_transfer*.
- Funkce pro impedanční přenos – *impedance_transfer.*
- Funkce pro admitanční přenos – *admittance_transfer*.

Nejdříve je ovšem nutné popsat formát vstupních dat, se kterým program dokáže pracovat.

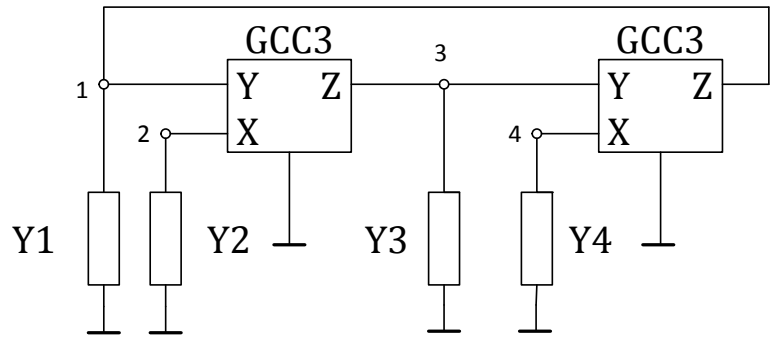
## 3.1 Formát vstupních dat

Jako zdroje analýzy slouží soubory s příponou *.nuc (dále jen soubory). Soubory obsahují vektor *M*, počet admitancí, počet a typ aktivních prvků. Ve všech případech se jedná o aktivní prvky GCC3. Obsah souboru vypadá následovně:

**Výpis kódu 3.1: Obsah vstupního souboru**

```
Number of admittances: 4
M vector is: [ 0  1  0  2  0  3  0  4  1  2  3  3  4  1]
1st active element: GCC3
2nd active element: GCC3
Determinant: Y3*Y1-Y4*cu3*vo2*Y1-Y3*cu1*vo1*Y2-
cu4*vo1*Y2*Y4*cu2*vo2+Y4*cu3*vo2*cu1*vo1*Y2
```

Vektor *M* definuje, jak jsou prvky v obvodu zapojeny. Každá admitance v obvodu je charakterizována dvojicí čísel, které představují uzly, mezi kterými je připojena. U aktivních prvků GCC3 je to obdobné. Jsou ovšem charakterizovány trojicí čísel, protože obsahují tři výstupní svorky (*y, x, z*). Berme uvedený vektor zleva. *Y1* je první admitance, která je připojena mezi uzel *0* (zem) a uzel *1*, *Y2* je připojena mezi uzel *0* a uzel *2*, atd. Pro první GCC3 platí, že výstupní svorka *y* je připojena do uzlu *1*, svorka *x* do uzlu *2* a výstupní svorka *z* do uzlu *3*. Druhý GCC3 je připojen mezi uzly *3, 4* a *1*. Celkové zapojení ve schématické podobě je na Obr. 3.1.




**Obr. 3.1: Zapojení ve schématické podobě**

## 3.2 Popis programu

AnalysisCircuit je hlavní script, který hledá relevantní vstupní soubory. Podle zadání uživatele spouští požadované funkce pro výpočty přenosu a následně funkci pro uložení výsledku do souboru. Zobrazením „Complete“ v Command Window program oznámí uživateli dokončení všech výpočtů. Zjednodušený vývojový diagram scriptu viz příloha B.

Po spuštění scriptu je uživatel vyzván k výběru přenosové funkce, která má být vypočítána. Tato volba se uloží do proměnné *quest* pod písmenem, kde každé písmeno odpovídá zvolené přenosové funkci. Pokud uživatel žádnou nevybere, je automaticky vybrána napěťová (tím se předchází chybovému hlášení, kdy program „neví“, který přenos má počítat). Může být zvoleno i více přenosových funkcí najednou, tzn. do proměnné *quest* se uloží řetězec znaků. Následně je uživatel vyzván k výběru adresáře, který obsahuje vstupní soubory. Je-li vybraný adresář prázdný, vypíše se v Command Window upozornění. Níže je uveden výpis programového řešení:

**Výpis kódu 3.2: Uživatelská část**

```
quest = input ('Chcete vypočítat Ku/Ki/Zt/Yt? [a/b/c/d]: [a]', 's');
% do proměnné quest se ukládá písmeno zvoleného přenosu
if isempty(quest)                   % je-li je quest prázdné
quest = 'a';                        % uloží do quest a
end
path_search=uigetdir('C:\');        % uloží cestu adresáře
cd(path_search);                    % vstoupí do adresáře
folder=dir (path_search);           % uloží obsah adresáře do folder
if size(folder,1) == 2              % pokud je adresář prázdný
disp('Adresář je prázdný!')         % vypíše text
else
```

Není-li vybraný adresář prázdný, program pokračuje vyhledáváním souborů. Neobsahuje-li adresář žádné soubory, jsou postupně prohledávány podadresáře. Pokud ani tentokrát žádné soubory nenalezne, dojde k jeho ukončení s výpisem „Complete“. Ovšem najde-li soubory, uloží se jejich celkový počet spolu s názvem souboru do proměnné *listing*. V případě nalezení souborů přímo ve zvoleném

adresáři vytvoří se prázdná proměnná s názvem *name_folder*. Najde-li soubory až v podadresáři, do proměnné se uloží název podadresáře.

Spuštění každé přenosové funkce je dané podmínkou. V podmínce se ověřuje, zda řetězec v proměnné *quest* obsahuje písmeno vybrané přenosové funkce:

**Výpis kódu 3.3: Podmínka spuštění přenosové funkce**

```
if ~isempty(strfind(quest, 'a'))                    % Napěťový přenos
[D, M, imitt, node]=voltage_transfer(file, path_file);
quest2='a';
save_file (D, M, imitt, node, quest2, name_folder)
end

if ~isempty(strfind(quest, 'b'))                    % Proudový přenos
[D, M, imitt, node]=current_transfer(file, path_file);
quest2='b';
save_file (D, M, imitt, node, quest2, name_folder)
end
```

Je-li podmínka splněna, spustí se příslušná funkce. Vstupními parametry funkce jsou vždy proměnné *file* a *path_file*. Proměnná *file* obsahuje jméno souboru a v *path_file* je uložena cesta k souboru, se kterým bude funkce pracovat. Po vykonání funkce se do proměnné *quest2* uloží písmeno odpovídající vypočítané přenosové funkci.

Poslední částí, o kterou se script stará, je spuštění funkce pro uložení výsledků do souboru. Vstupními parametry tedy jsou výsledky z předchozí volané funkce spolu s proměnnými *quest2* a *name_folder*. Podrobný popis této funkce bude vysvětlen v poslední části této kapitoly.

## 3.3 Vytvoření admitanční a pseudoadmitanční matice

Základem kteréhokoli výpočtu přenosu obvodu je sestavení jeho admitanční a následně pseudoadmitanční matice. Tento základ je tedy pro všechny funkce, které počítají přenosové funkce, stejný. Další část je věnována sestavování těchto matic ze zadaných vstupních souborů.

V první části jsou data ze souboru načtena do proměnných jako textový řetězec. Následuje převod, podle potřeby, z textových řetězců do číselné, vektorové nebo symbolické podoby:

**Výpis kódu 3.4: Načtení a zpracování souboru**

```
fid=fopen(file,'r');          % otevře soubor, režim čtení
while feof(fid)==0
    imitt_char=fgetl(fid);
    seed=fgetl(fid);
    act01=fgetl(fid);
    act02=fgetl(fid);
    det_x=fgetl(fid);
end
fclose(fid);

for i=1:length(imitt_char)    % převede počet pasivních prvků na číslo
    if(strfind(imitt_char, ': '))
        imitt=str2num(imitt_char(i));
    end
end

for i=1:length(seed)          % převede "seed" na vektor M
    K=strcmp ('[',seed(i));
    if K==true
        M=str2num( seed(i+1:end-1))+1;
    end
end
```

S použitím příkazu *while* se každý řádek ze souboru uloží do příslušné proměnné. Následně, jak je uvedeno ve výpisu, se převede číslo, které udává počet použitých admitancí v obvodu, z textové formy na numerickou formu a uloží se do proměnné *imitt*. Obdobně se převede i vektor *M* z textové formy na numerickou a ke každému číslu ve vektoru se přičte *1* pro snadnější sestavení matice. Uzel *1* nyní představuje zem.

Ve druhé části se vytváří úplná admitanční matice *Y_p*, která obsahuje pouze admitance. Matice *Y_p* má rozměry dané součtem hodnot v proměnné *node* a *rowcols*. Proměnná *node* obsahuje celkový počet uzlů, který je dán největším číslem ve vektoru *M*. V proměnné *rowcols* je uveden počet aktivních prvků. Na obvod je nyní nahlíženo tak, že zvolený referenční uzel se nachází vně obvodu. Viz Vytvoření admitanční matice. Z této skutečnosti vychází zapisování jednotlivých admitancí na příslušné místo v matici. Programové řešení je následovné:

**Výpis kódu 3.5: Vytváření úplně admitanční matice**

```
Y_p=sym(zeros(node+rowcols));                % vytvoří pasivní matice
for i=1:imitt
    Y_p(M(2*i-1),M(2*i-1))=Y_p(M(2*i-1),M(2*i-1)) + Y(i)    %admitance
připojené k zemi
    Y_p(M(2*i),M(2*i))=Y_p(M(2*i),M(2*i)) + Y(i)            %admitance
připojené k zemi
    Y_p(M(2*i-1),M(2*i))=Y_p(M(2*i-1),M(2*i)) - Y(i)        %admitance
připojené mezi dvěma uzly
    Y_p(M(2*i),M(2*i-1))=Y_p(M(2*i),M(2*i-1)) - Y(i)        %admitance
připojené mezi dvěma uzly
end
```

Mějme zapojení uvedené na Obr. 3.1. Výsledná úplná admitanční matice odpovídající tomuto zapojení je na Obr. 3.2.

| | | | | | | |
|---|---|---|---|---|---|---|
| Y1+Y2+Y3+Y4 | -Y1 | -Y2 | -Y3 | -Y4 | 0 | 0 |
| -Y1 | Y1 | 0 | 0 | 0 | 0 | 0 |
| -Y2 | 0 | Y2 | 0 | 0 | 0 | 0 |
| -Y3 | 0 | 0 | Y3 | 0 | 0 | 0 |
| -Y4 | 0 | 0 | 0 | Y4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Obr. 3.2: Úplná admitanční matice *Y_p***

Pro zjednodušení programu je matice *Y_p* již rozšířena o počet řádků a sloupců odpovídající počtu aktivních prvků. Jak již bylo popsáno v kapitole 1.4, každý aktivní prvek GCC3 je popsán třemi přenosovými koeficienty tzn., první aktivní prvek charakterizují koeficienty *vo1*, *cu1* a *cu2*. Druhý potom *vo2*, *cu3* a *cu4*. Zapisování těchto koeficientů na příslušné místo v matici *Y_a*, která je duplicitní s maticí *Y_p*, vychází z metody razítek.

**Výpis kódu 3.6: Pseudoadmitanční matice Y_a**

```
for i=1:cur_op      %Generování správného počtu proudových koeficientů
    cu(i)=sym(sprintf('cu%d',i));
end
for i=1:vol_op      %Genereting propper number of voltage coefficients
    vo(i)=sym(sprintf('vo%d',i));
end

for i=1:length(acts)                % vytvoří pseudoadmitanční matici
    if (strfind(acts{i},'GCC3'))    %GCC3 - three-port GCC
        ports=ports+3;
        rowcols=rowcols+1;
        cur_op=cur_op+2;
        vol_op=vol_op+1;
        %modifing "rowcols" column
        Y_a(M(2*imitt+ports-2),max(M)+rowcols)=Y_a(M(2*imitt+ports-
2),max(M)+rowcols)+cu(cur_op-1);              %Y ~ M(2*imitt+ports-2)

Y_a(M(2*imitt+ports),max(M)+rowcols)=Y_a(M(2*imitt+ports),max(M)+rowco
ls)+cu(cur_op);                                 %Z ~ M(2*imitt+ports)
        Y_a(M(2*imitt+ports-1),max(M)+rowcols)=Y_a(M(2*imitt+ports-
1),max(M)+rowcols)+1;                         %X ~ M(2*imitt+ports-1)
        %modifing "rowcols" row
        Y_a(max(M)+rowcols,M(2*imitt+ports-
2))=Y_a(max(M)+rowcols,M(2*imitt+ports-2))+vo(vol_op);
        Y_a(max(M)+rowcols,M(2*imitt+ports-
1))=Y_a(max(M)+rowcols,M(2*imitt+ports-1))-1;
    end
end

Y_a(1,:)=[];
Y_a(:,1)=[];
```

Aby byla výsledná matice pseudoadmitanční, je ještě nutné smazat první řádek a první sloupec. Tímto prohlásíme uzel *1*, který představuje zem, za referenční. Výsledná pseudoadmitanční matice je zobrazena na Obr. 3.3.

$$\begin{bmatrix} Y1 & 0 & 0 & 0 & cu1 & cu4 \\ 0 & Y2 & 0 & 0 & 1 & 0 \\ 0 & 0 & Y3 & 0 & cu2 & cu3 \\ 0 & 0 & 0 & Y4 & 0 & 1 \\ vo1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & vo2 & -1 & 0 & 0 \end{bmatrix}$$

**Obr. 3.3: Pseudoadmitanční matice Y_a**

## 3.4 Popis výpočtu přenosových funkcí

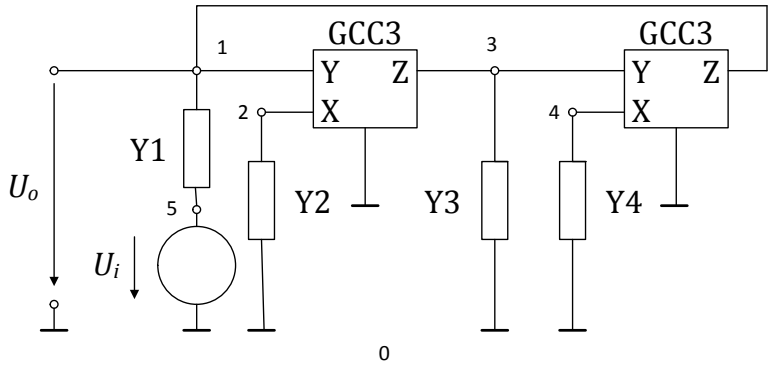
Pseudoadmitanční matice *Y_a* ovšem popisuje pouze obvod, který není buzen z žádného zdroje. Abychom byli schopni analyzovat chování obvodu, pomocí přenosových funkcí, musíme připojit napěťový nebo proudový zdroj. To znamená, je nutné upravit matici *Y_a*. Pro každou přenosovou funkci je úprava jiná.

Při výpočtu napěťového a admitančního přenosu připojíme zdroj napětí mezi zem a admitanci, popř. výstupní svorku konvejoru (*y*), která je uzemněná. Při výpočtu zbylých dvou přenosů tj. proudového a impedančního, připojíme zdroj proudu do uzlu. Postupným připojováním těchto zdrojů mezi zem a admitance, konvejory popř. do uzlů, jsme schopni analyzovat chování při různých zapojeních.

Dále budou popsány úpravy pseudoadmitanční matice *Y_a*, pro konkrétní přenosovou funkci.

### 3.4.1 Napěťový přenos

Mějme situaci, kdy připojíme napěťový zdroj mezi admitanci *Y1* a zem, výstupní uzel bude uzel *1* a chceme vypočítat napěťový přenos, viz Obr. 3.4.




**Obr. 3.4: Zapojení zdroje napětí pří výpočtu napěťového přenosu**

Pomocí metody razítek rozšíříme matici *Y_a* o jeden řádek a jeden sloupec, vytvoří se nový, vstupní uzel *5*. Matice *Y_a* nyní bude vypadat následovně:

| | | | | | | |
|---|---|---|---|---|---|---|
| Y1 | 0 | 0 | 0 | cu1 | cu4 | -Y1 |
| 0 | Y2 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | Y3 | 0 | cu2 | cu3 | 0 |
| 0 | 0 | 0 | Y4 | 0 | 1 | 0 |
| vo1 | -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | Vo2 | -1 | 0 | 0 | 0 |
| -Y1 | 0 | 0 | 0 | 0 | 0 | Y1 |

**Obr. 3.5: Rozšířená matice *Y_a***

Výpočet napěťového přenosu provedeme podle rovnice (1.4). V čitateli tedy bude algebraický doplněk při vynechání sedmého řádku a prvního sloupce. Ovšem algebraický doplněk ve jmenovateli, který vznikne vynecháním sedmého řádku a sedmého sloupce, je shodný s determinantem původní nerozšířené matice, čehož je v programu využíváno. Programové řešení pro všechny kombinace zapojení je rozděleno na dva cykly. Základem prvního je připojení napěťového zdroje mezi admitanci a zem. U druhého se napěťový zdroj připojuje mezi svorku *y* konvejoru a zem tzn., nejdříve je kontrolována první část vektoru *M* a následně druhá. Řešení je ve Výpis kódu 3.7 a Výpis kódu 3.8.

**Výpis kódu 3.7: Napěťový přenos – připojení zdroje mezi admitanci a zem**

```
for x = 1:2:(2*imitt)
    if M(x)==1
        Y_a2=Y_a1;
        Y_a2(a(1),i)=-Y(i);
        Y_a2(i,a(2))=-Y(i);
        Y_a2(a(1),a(1))=Y(i);
        fprintf('\nZdroj napětí je připojený mezi zem a Y%d.\n', i);
        for z= 1:node                              % Napěťové přenosy
            Y_a3=Y_a2;
            Y_a3(a(1),:)=[];
            Y_a3(:,z)=[];
            D(d,z)=((-1)^(a(1)+z)*(det(Y_a3)))/(det(Y_a));
            fprintf(' Ku = U%d/Uvst = %s\n', z, char(D(d,z)));
        end
        d=d+1;
    end
    i=i+1;
end
```

Pro rychlejší modifikaci matic je využito pomocných matic *Y_a1*, *Y_a2* a *Y_a3*. Matice *Y_a1* je, na rozdíl od matice *Y_a,* rozšířena o nulový řádek a sloupec. Pomocí cyklu *for* a podmínky se hledají uzemněné admitance ve vektoru *M*. Je-li podmínka splněna, uloží se do pomocné matice *Y_a2* matice *Y_a1*. Do matice *Y_a2* se podle metody razítek doplní admitance. Ve vnořeném cyklu *for* se uloží matice *Y_a2* do *Y_a3*. Pro každý vstupní a výstupní uzel se z matice *Y_a3* maže příslušný

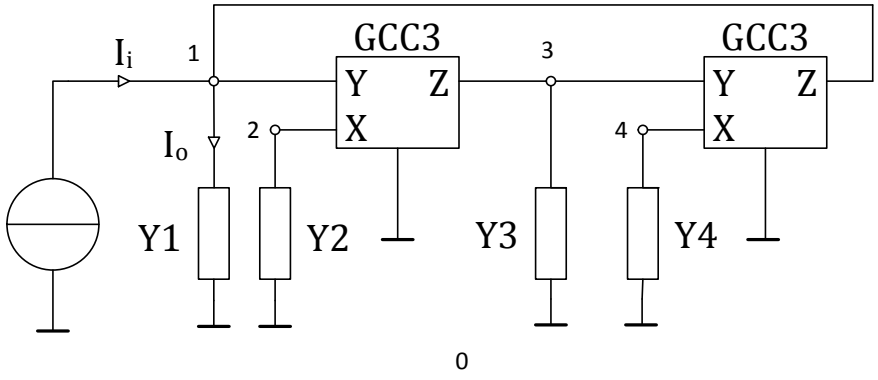
řádek a sloupec viz rovnice (1.4). Výsledky napěťových přenosů se ukládají do matice *D*, kde každý řádek představuje jinou pozici zapojení napěťového zdroje v obvodu a sloupce jednotlivé přenosy pro výstupní uzly.

**Výpis kódu 3.8: Napěťový přenos – připojení zdroje mezi konvejor a zem**

```
for x = 9:3:12
    if M(x)==1
        y=node+1;
        Y_a2=Y_a1;
        Y_a2(a(1),y)=vo(i);
        Y_a2(y,a(1))=vo(i);
        Y_a2(a(1),a(1))=vo(i);
        fprintf('\nZdroj napětí je připojený ke vstupu "y"
%d.konvejoru\n', i);

        for z= 1:(node)
            Y_a3=Y_a2;
            Y_a3(a(1),:)=[];
            Y_a3(:,z)=[];
            D(d,z)=((-1)^(a(1)+z)*(det(Y_a3)))/(det(Y_a));
            fprintf(' Ku = U%d/Uvst = %s\n', z, char(D(d,z)));
        end
    end
    i=i+1;
end
```

### 3.4.2 Proudový přenos

Pro proudový přenos je postup podobný. Mějme zapojení, kdy je zdroj proudu připojený do uzlu *1* a chceme vypočítat proudový přenos mezi zdrojem proudu a proudem protékajícím admitancí *Y1*, viz Obr. 3.6. Pro výpočet této přenosové funkce se pseudoadmitanční matice nerozšiřuje. Jednoduše prohlásíme, že uzel *1* je vstupní. Při výpočtu vycházíme z rovnice (1.8) tzn., v čitateli dostaneme algebraický doplněk, při vynechání prvního řádku a prvního sloupce, násobený admitancí *Y1*. Jmenovatel bude obsahovat determinant pseudoadmitanční matice *Y_a*. Programové řešení viz Výpis kódu 3.9.




**Obr. 3.6: Zapojení zdroje proudu pří výpočtu proudového přenosu**

**Výpis kódu 3.9: Proudový přenos**

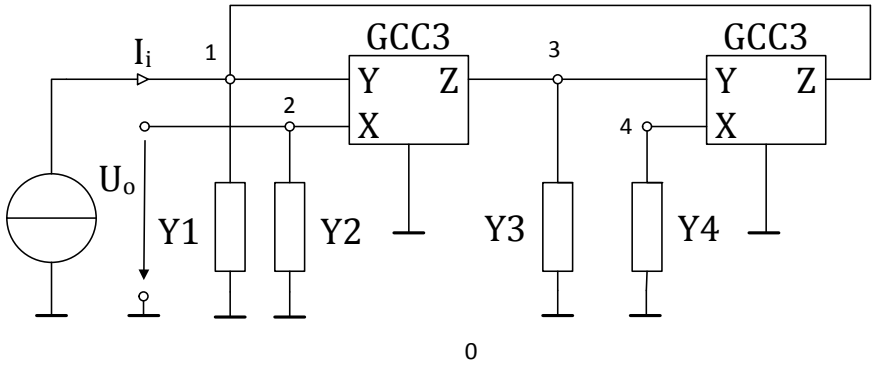
```
for z = 1:(node-1)
    fprintf('\nZdroj proudu je připojený do %d. uzlu\n', z);
    x=1;
    for i = 1:(node-1)                              % Proudové přenosy
        if M(x)==1
        Y_a1=Y_a;
        Y_a1(z,:)=[];
        Y_a1(:,i)=[];
        D(d,i)=Y(i)*(-1)^(z+i)*det(Y_a1)/det(Y_a);
        fprintf(' Ki = I%d/Ivst = %s\n', i, char(D(d,i)));
        end
        x=x+2;
    end
    d=d+1;
end
```

Cyklus *for* prohlašuje postupně každý uzel za vstupní. Vnořený cyklus *for* společně s podmínkou určuje výstupní proud tj., výstupní proud prochází uzemněnou admitancí. Stejně jako u napěťového přenosu je použita pomocná matice *Y_a1*, ve které se mažou příslušné řádky a sloupce. Výsledky se opět ukládají do matice *D*, kde každý řádek představuje uzel, do kterého je připojen zdroj proudu, a sloupce představují jednotlivé přenosy pro výstupní proudy.

### 3.4.3 Impedanční přenos

Stejně jako u proudového přenosu se i při výpočtu impedančního přenosu připojuje zdroj proudu do uzlů. Pseudoadmitanční matice *Y_a* se tedy nerozšiřuje. Výstupní uzel ovšem musí být jiný než ten, do nějž je připojen zdroj proudu. Pokud by tato podmínka nebyla splněna, jednalo by se o výpočet vstupní impedance namísto impedančního přenosu.

Mějme zapojení zobrazené na Obr. 3.7. Pro výpočet impedančního přenosu použijeme rovnici (1.5). V čitateli dostaneme algebraický doplněk při vynechání prvního řádku a druhého sloupce z matice *Y_a* a ve jmenovatel pak determinant pseudoadmitanční matice *Y_a*. Programové řešení pro všechny kombinace zapojení je ve Výpis kódu 3.10.




**Obr. 3.7: Impedanční přenos**

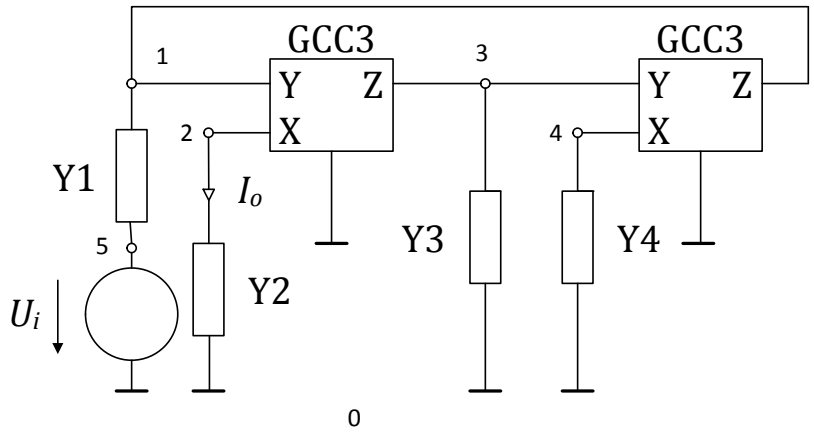
**Výpis kódu 3.10: Impedanční přenos**

```
for z = 1:(node-1)
    fprintf('\nZdroj proudu je připojený do %d. uzlu\n', z);
    for i = 1:(node-1)                          % Impedanční přenosy
        if z~=i
            Y_a1=Y_a;
            Y_a1(z,:)=[];
            Y_a1(:,i)=[];
            D(d,i)=(-1)^(z+i)*det(Y_a1)/det(Y_a);
            fprintf(' Zt = U%d/Ivst = %s\n', i, char(D(d,i)));
        end
    end
    d=d+1;
end
```

Cyklus *for* opět prohlašuje postupně každý uzel za vstupní. Vnořený cyklus společně s podmínkou určuje výstupní uzel, který nesmí být stejný jako uzel vstupní. Následně je použita pomocná matice *Y_a1*, ve které se mažou příslušné řádky a sloupce odpovídající vstupnímu a výstupnímu uzlu. Výsledek se uloží do matice *D*, podle stejného algoritmu jako v případě proudového přenosu, tj. řádek představuje vstupní uzel a sloupce potom impedanční přenosy pro jednotlivé výstupní uzly.

### 3.4.4 Admitanční přenos

Admitanční přenos je dán poměrem výstupního proudu a vstupního napětí. Při výpočtu přenosu připojujeme zdroj napětí mezi admitanci, popř. svorku *y* konvejoru, a zem. Za výstupní proud se považuje proud procházející uzemněnou admitancí. Mějme zapojení podle Obr. 3.8. Protože je použit zdroj napětí, musíme pseudoadmitanční matici *Y_a* rozšířit podle metody razítek. Při výpočtu přenosu vycházíme ze vzorce (1.7). V čitateli bude algebraický doplněk při vynechání sedmého řádku, tedy vstupního uzlu, a druhého sloupce. Jmenovatel bude obsahovat determinant matice *Y_a*, který je shodný s algebraickým doplněk při vynechání sedmého řádku a sedmého sloupce.




**Obr. 3.8: Admitanční přenos**

Programové řešení pro všechny kombinace zapojení, je stejně jako u napěťového přenosu, prováděno pomocí dvou cyklů (viz Výpis kódu 3.11 a Výpis kódu 3.12). První z nich připojuje zdroj napětí mezi admitance a zem, druhý mezi svorku *y* konvejoru a zem. Pro správnou interpretaci admitancí připojených k zemi je v prvním zanořeném cyklu dána podmínka. Tato podmínka zabraňuje označení výstupního proudu pro admitanci, ke které je připojen zdroj napětí.

**Výpis kódu 3.11: Admitanční přenos – připojení zdroje mezi admitanci a zem**

```
for x = 1:2:(2*imitt)
    if M(x)==1
        Y_a2=Y_a1;
        Y_a2(a(1),i)=-Y(i);
        Y_a2(i,a(2))=-Y(i);
        Y_a2(a(1),a(1))=Y(i);
        fprintf('\nZdroj napětí je připojený mezi zem a Y%d.\n', i);
        q=1;
        for z= 1:node                                % Admitanční přenos
            if M(q)==1 && i~=z
                Y_a3=Y_a2;
                Y_a3(a(1),:)=[];
                Y_a3(:,z)=[];
                D(d,z)=-Y(z)*(-1)^(a(1)+z)*det(Y_a3)/(det(Y_a));
                fprintf(' Yt = I%d/Uvst = %s\n', z, char(D(d,z)));
            end
            q=q+2;
        end
        d=d+1;
    end
    i=i+1;
end
```

**Výpis kódu 3.12: Admitanční přenos – připojení zdroje mezi konvejor a zem**

```
for x = 9:3:12
    if M(x)==1
        y=node+1;
        Y_a2=Y_a1;
        Y_a2(a(1),y)=vo(i);
        Y_a2(y,a(1))=vo(i);
        Y_a2(a(1),a(1))=vo(i);
        fprintf('\nZdroj napětí je připojený ke vstupu "y"
%d.konvejoru\n', i);
        q=1;
        for z= 1:(node)
            if M(q)==1
                Y_a3=Y_a2;
                Y_a3(a(1),:)=[];
                Y_a3(:,z)=[];
                D(d,z)=-Y(z)*(-1)^(a(1)+z)*det(Y_a3)/(det(Y_a));
                fprintf(' Yt = I%d/Uvst = %s\n', z, char(D(d,z)));
            end
            q=q+2;
        end
    end
    i=i+1;
end
```

Ve výpisu si můžete všimnout, že první část, upravování matice *Y_a*, je shodná jako u napěťového přenosu. Tato úprava vychází z připojování napěťových zdrojů. Druhá část potom představuje podobnost s proudovým přenosem tj., označení výstupního proudu příslušnou admitancí. Tímto byl celý algoritmus zjednodušen.

## 3.5 Ukládání výsledků do souboru

Poslední funkcí je funkce pro uložení výsledků do textového souboru. Název souboru obsahuje zkratku, která značí typ vypočítaného přenosu, a vektor *M*, který popisuje obvod. Zkratka VM odpovídá napěťovému přenosu, CM proudovému, IM impedančnímu a AM admitančnímu.

Funkce *save_file* využívá šest vstupních proměnných a to *D*, *M*, *imitt*, *node*, *quest2* a *name_folder*, pomocí kterých se zajišťuje správné uložení.

V první části se ověřuje zda, je proměnná *name_folder* prázdná. Není-li prázdná, je v kořenovém adresáři (ve kterém se nachází celý program) vytvořen podadresář s názvem, který je obsahem proměnné. Pokud podadresář s tímto názvem již existuje, nový se nevytvoří, a výsledky jsou ukládány do něj. Programové řešení je následující:

**Výpis kódu 3.13: Ukládání do podadresářů**

```
if ~isempty(name_folder)
    if exist(name_folder,'dir')
        cd (name_folder)
    else
        mkdir (name_folder)
        cd(name_folder)
    end
end
```

Ve druhé části se provádí ukládání výsledku do textového souboru. Uložení správného přenosu je řešeno podmínkou. Tzn., podle písmene uloženého v proměnné *quest2* je spuštěn proces ukládání pro příslušný přenos. Algoritmus pro ukládání výsledků do souboru je téměř stejný jako při ukládání výsledku z přenosové funkce do matice *D*. Programové řešení pro uložení proudového přenosu znázorňuje Výpis kódu 3.14. Pro jiné přenosy je postup obdobný.

**Výpis kódu 3.14: Uložení výsledku proudového přenosu do souboru**

```
elseif quest2 == 'b'    % Proudový přenos
    clear fid
    fid=fopen(['CM_' seed_c '.txt'],'w');
    for z = 1:(node-1)
        fprintf(fid,'\nZdroj proudu je připojený do %s.uzlu\n',
num2str(z));
        x=1;
        for i = 1:(node-1)
            if M(x)==1
                fprintf(fid,' Ki = I%s/Ivst = %s\n', num2str(i),
char(D(z,i)));
            end
            x=x+2;
        end
    end
    fclose(fid);
    clear fid
```

Obsahuje-li proměnná *quest2* písmeno *b* je spuštěn proces ukládání výsledků proudového přenosu z matice *D*. Je vytvořen textový soubor, který má v názvu zkratku CM a příslušný vektor popisující obvod. Cyklus následně prochází všechny uzly, do kterých byl připojen zdroj proudu, čemuž odpovídá příslušný řádek z matice *D*. Ve vnořeném cyklu, společně s podmínkou, jsou hledány výstupní proudy, tzn., uzemněné admitance z vektoru *M*. Tímto způsobem funkce přesně určuje umístění buňky v matici *D*, která má být uložena. Následně je obsah buňky společně s popisem, o který přenos jde, uložen na samostatný řádek v souboru. Část výstupního soubor pro proudový přenos je znázorněn ve Výpis kódu 3.15.

**Výpis kódu 3.15: Část výpisu výsledného souboru pro impedanční přenos**

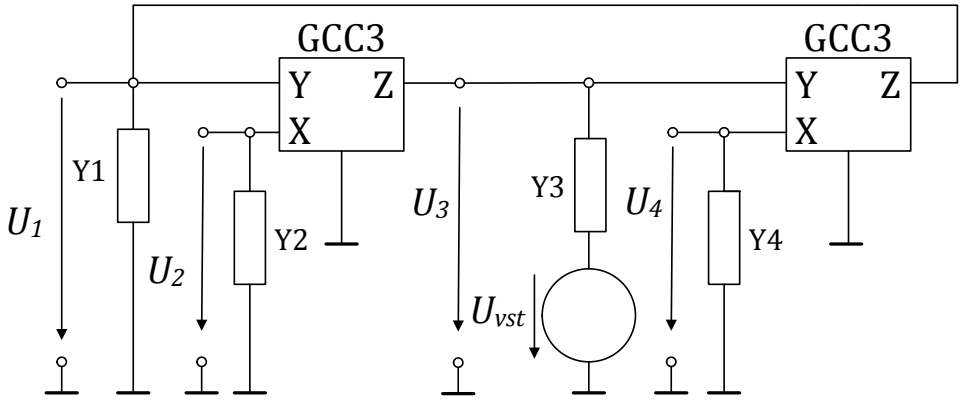
```
Zdroj proudu je připojený do 1. uzlu
 Ki = I1/Ivst = -(Y1*(Y3 - Y4*cu3*vo2))/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I2/Ivst = -(Y2*(Y3*vo1 - Y4*cu3*vo1*vo2))/(Y2*Y3*cu1*vo1 - Y1*Y3
+ Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I3/Ivst = -(Y2*Y3*cu2*vo1)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I4/Ivst = -(Y2*Y4*cu2*vo1*vo2)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)

Zdroj proudu je připojený do 2. uzlu
 Ki = I1/Ivst = (Y1*(Y3*cu1 - Y4*cu1*cu3*vo2 +
Y4*cu2*cu4*vo2))/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2 -
Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I2/Ivst = (Y2*(Y3*cu1*vo1 - Y4*cu1*cu3*vo1*vo2 +
Y4*cu2*cu4*vo1*vo2))/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2 -
Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I3/Ivst = (Y1*Y3*cu2)/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2 -
Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I4/Ivst = (Y1*Y4*cu2*vo2)/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2
- Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
```

Po každém správném uložení výsledku do souboru je tato skutečnost oznámena vypsáním „OK“ v Command Window.

# 4 Porovnání výsledků z programu SNAP

Pro porovnání správnosti výsledků byl zvolen, jako referenční, program SNAP.

Pro napěťový přenos byl zvolen obvod, který je zobrazen na Obr. 4.1.




**Obr. 4.1: Porovnání - napěťový přenos**

**Výpis kódu 4.1: Výsledky napěťového přenosu - MATLAB**

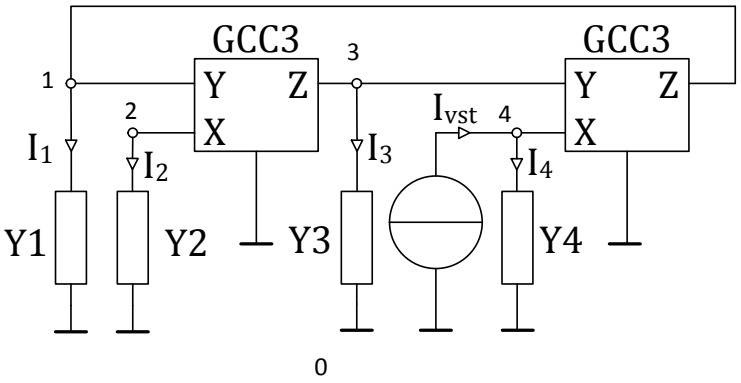
```
Zdroj napětí je připojený mezi zem a Y3.
 Ku = U1/Uvst = -(Y3*Y4*cu4*vo2)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ku = U2/Uvst = -(Y3*Y4*cu4*vo1*vo2)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ku = U3/Uvst = -(Y1*Y3 - Y2*Y3*cu1*vo1)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ku = U4/Uvst = -(Y1*Y3*vo2 - Y2*Y3*cu1*vo1*vo2)/(Y2*Y3*cu1*vo1 -
Y1*Y3 + Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
```

**Výpis kódu 4.2: Výsledky napěťového přenosu - SNAP**

```
Ku = U1/Uvst:
Y4*Y3*X2.VO2*X2.CU4
-------------------------------------
Y2*Y4*X1.VO1*X1.CU1*X2.VO2*X2.CU3 -Y2*Y4*X1.VO1*X1.CU2*X2.VO2*X2.CU4 -
Y2*Y3*X1.VO1*X1.CU1 -Y4*Y1*X2.VO2*X2.CU3 +Y3*Y1

Ku = U2/Uvst:
Y4*Y3*X1.VO1*X2.VO2*X2.CU4
-------------------------------------
Y2*Y4*X1.VO1*X1.CU1*X2.VO2*X2.CU3 -Y2*Y4*X1.VO1*X1.CU2*X2.VO2*X2.CU4 -
Y2*Y3*X1.VO1*X1.CU1 -Y4*Y1*X2.VO2*X2.CU3 +Y3*Y1

Ku = U3/Uvst:
-Y2*Y3*X1.VO1*X1.CU1 +Y3*Y1
-------------------------------------
Y2*Y4*X1.VO1*X1.CU1*X2.VO2*X2.CU3 -Y2*Y4*X1.VO1*X1.CU2*X2.VO2*X2.CU4 -
Y2*Y3*X1.VO1*X1.CU1 -Y4*Y1*X2.VO2*X2.CU3 +Y3*Y1

Ku = U4/Uvst:
-Y2*Y3*X1.VO1*X1.CU1*X2.VO2 +Y3*Y1*X2.VO2
-------------------------------------
 -Y2*Y3*X1.VO1*X1.CU1 +Y2*X1.VO1*X1.CU1*Y4*X2.VO2*X2.CU3 -
Y2*X1.VO1*X1.CU2*Y4*X2.VO2*X2.CU4 +Y3*Y1 -Y1*Y4*X2.VO2*X2.CU3
```

Z výpisu výsledků z programu MATLAB, viz Výpis kódu 4.1, a programu SNAP, vizVýpis kódu 4.2, můžeme říct, že výsledky jsou stejné. Rozdíl je pouze v matematickém znázornění, kdy je jmenovatel vynásoben číslem $-1$. Tím se změní i znaménko v čitateli. Koeficienty, které mají ve výsledku z programu SNAP, *X1* popř. *X2* označují, že se jedná o konvejor.

Pro proudový přenos byl zvolen obvod, který je zobrazen na Obr. 4.2.




**Obr. 4.2: Porovnání - proudový přenos**

**Výpis kódu 4.3: Výsledky proudového přenosu - MATLAB**

```
Zdroj proudu je připojený do 4.uzlu
 Ki = I1/Ivst = (Y1*Y3*cu4)/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2 -
Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I2/Ivst = (Y2*Y3*cu4*vo1)/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2
- Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I3/Ivst = (Y3*(Y1*cu3 - Y2*cu1*cu3*vo1 +
Y2*cu2*cu4*vo1))/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2 -
Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Ki = I4/Ivst = (Y4*(Y1*cu3*vo2 - Y2*cu1*cu3*vo1*vo2 +
Y2*cu2*cu4*vo1*vo2))/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2 -
Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
```

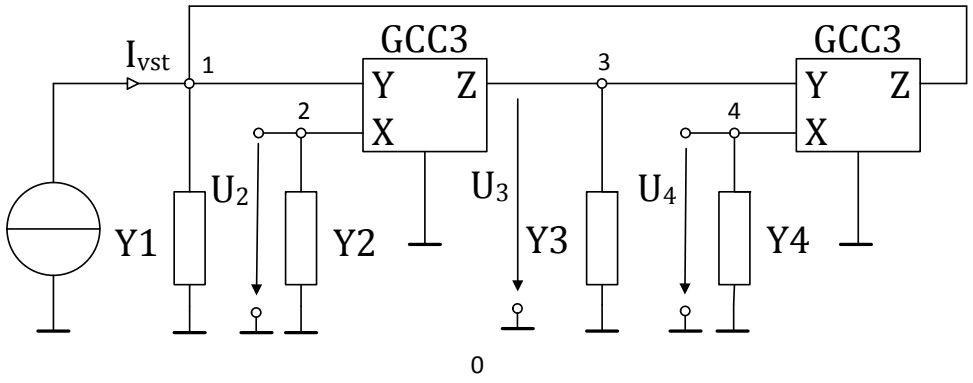
**Výpis kódu 4.4: Výsledky proudového přenosu - SNAP**

```
Ki = I1/Ivst:
X2.CU4*Y3*Y1
-------------------------------------
 -Y4*X2.VO2*X2.CU3*Y1 +Y3*Y1 +X1.VO1*X1.CU1*Y4*X2.VO2*X2.CU3*Y2 -
X1.VO1*X1.CU1*Y2*Y3 -X1.VO1*X1.CU2*Y4*X2.VO2*X2.CU4*Y2

Ki = I2/Ivst:
X1.VO1*X2.CU4*Y3*Y2
-------------------------------------
 -Y4*X2.VO2*X2.CU3*Y1 +Y3*Y1 +X1.VO1*X1.CU1*Y4*X2.VO2*X2.CU3*Y2 -
X1.VO1*X1.CU1*Y3*Y2 -X1.VO1*X1.CU2*Y4*X2.VO2*X2.CU4*Y2

Ki = I3/Ivst:
X2.CU3*Y1*Y3 -X1.VO1*X1.CU1*X2.CU3*Y2*Y3 +X1.VO1*X1.CU2*X2.CU4*Y2*Y3
-------------------------------------
 -Y4*X2.VO2*X2.CU3*Y1 +Y1*Y3 +X1.VO1*X1.CU1*Y4*X2.VO2*X2.CU3*Y2 -
X1.VO1*X1.CU1*Y2*Y3 -X1.VO1*X1.CU2*Y4*X2.VO2*X2.CU4*Y2
```

```
Ki = I4/Ivst:
X2.VO2*X2.CU3*Y1*Y4 -X1.VO1*X1.CU1*X2.VO2*X2.CU3*Y2*Y4
+X1.VO1*X1.CU2*X2.VO2*X2.CU4*Y2*Y4
-------------------------------------
Y1*Y3 -X2.VO2*X2.CU3*Y1*Y4 -X1.VO1*X1.CU1*Y2*Y3
+X1.VO1*X1.CU1*X2.VO2*X2.CU3*Y2*Y4 -X1.VO1*X1.CU2*X2.VO2*X2.CU4*Y2*Y4
```

Z výpisu výsledků z programu MATLAB, viz Výpis kódu 4.3, a programu SNAP, viz Výpis kódu 4.4Výpis kódu 4.2, je zřejmé, že výsledky nejsou úplně stejné, což je způsobené definicí proudového přenosu. Při výpočtu proudového přenosu bylo vycházeno z literatury [2], kde je přenos dán kladným poměrem výstupního a vstupního proudu, viz (1.8).

Pro impedanční přenos byl zvolen obvod, který je zobrazen na Obr. 4.3.




**Obr. 4.3: Porovnání - impedanční přenos**

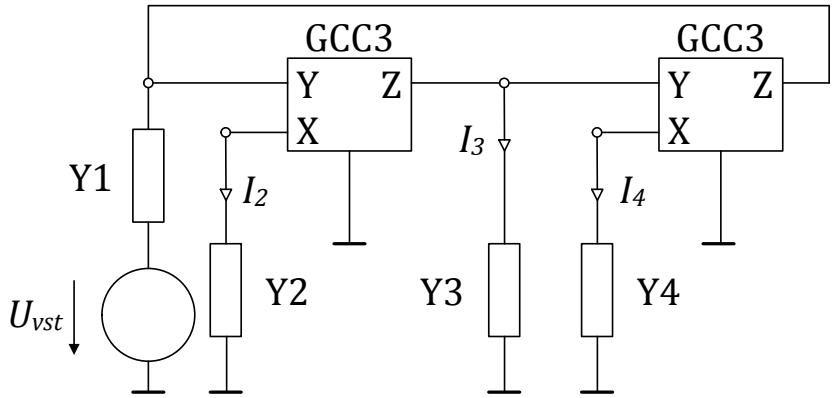
**Výpis kódu 4.5: Výsledky impedančního přenosu - MATLAB**

```
Zdroj proudu je připojený do 1. uzlu
 Zt = U2/Ivst = -(Y3*vo1 - Y4*cu3*vo1*vo2)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Zt = U3/Ivst = -(Y2*cu2*vo1)/(Y2*Y3*cu1*vo1 - Y1*Y3 + Y1*Y4*cu3*vo2 -
Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Zt = U4/Ivst = -(Y2*cu2*vo1*vo2)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
```

**Výpis kódu 4.6: Výsledky impedančního přenosu - SNAP**

```
Zt = U2/Ivst:
-Y4*X1.VO1*X2.VO2*X2.CU3 +Y3*X1.VO1
-------------------------------------
Y2*Y4*X1.VO1*X1.CU1*X2.VO2*X2.CU3 -Y2*Y4*X1.VO1*X1.CU2*X2.VO2*X2.CU4 -
Y2*Y3*X1.VO1*X1.CU1 -Y4*Y1*X2.VO2*X2.CU3 +Y3*Y1

Zt = U3/Ivst:
Y2*X1.VO1*X1.CU2
-------------------------------------
Y2*Y4*X2.VO2*X2.CU3*X1.VO1*X1.CU1 -Y2*Y4*X2.VO2*X2.CU4*X1.VO1*X1.CU2 -
Y2*X1.VO1*X1.CU1*Y3 -Y4*Y1*X2.VO2*X2.CU3 +Y1*Y3
Zt = U4/Ivst:
Y2*X1.VO1*X1.CU2
-------------------------------------
Y2*Y4*X2.VO2*X2.CU3*X1.VO1*X1.CU1 -Y2*Y4*X2.VO2*X2.CU4*X1.VO1*X1.CU2 -
Y2*X1.VO1*X1.CU1*Y3 -Y4*Y1*X2.VO2*X2.CU3 +Y1*Y3
```

Pro admitanční přenos byl zvolen obvod, který je zobrazen na Obr. 4.4.




**Obr. 4.4: Porovnání - admitanční přenos**

**Výpis kódu 4.7: Výsledky admitančního přenosu - MATLAB**

```
Zdroj napětí je připojený mezi zem a Y1.
 Yt = I2/Uvst = (Y2*(Y1*Y3*vo1 - Y1*Y4*cu3*vo1*vo2))/(Y2*Y3*cu1*vo1 -
Y1*Y3 + Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Yt = I3/Uvst = (Y1*Y2*Y3*cu2*vo1)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
 Yt = I4/Uvst = (Y1*Y2*Y4*cu2*vo1*vo2)/(Y2*Y3*cu1*vo1 - Y1*Y3 +
Y1*Y4*cu3*vo2 - Y2*Y4*cu1*cu3*vo1*vo2 + Y2*Y4*cu2*cu4*vo1*vo2)
```

**Výpis kódu 4.8: Výsledky admitančního přenosu - SNAP**

```
Yt = I2/Uvst:
-X1.VO1*Y3*Y1*Y2 +X1.VO1*X2.VO2*X2.CU3*Y1*Y2*Y4
-------------------------------------
Y3*Y1 -X2.VO2*X2.CU3*Y1*Y4 -X1.VO1*X1.CU1*Y3*Y2
+X1.VO1*X1.CU1*X2.VO2*X2.CU3*Y2*Y4 -X1.VO1*X1.CU2*X2.VO2*X2.CU4*Y2*Y4

Yt = I3/Uvst:
-X1.VO1*X1.CU2*Y1*Y3*Y2
-------------------------------------
Y1*Y3 -X2.VO2*X2.CU3*Y1*Y4 -X1.VO1*X1.CU1*Y3*Y2
+X1.VO1*X1.CU1*X2.VO2*X2.CU3*Y4*Y2 -X1.VO1*X1.CU2*X2.VO2*X2.CU4*Y4*Y2

Yt = I4/Uvst:
X2.VO2*X2.CU3*Y1*Y4 -X1.VO1*X1.CU1*X2.VO2*X2.CU3*Y2*Y4
+X1.VO1*X1.CU2*X2.VO2*X2.CU4*Y2*Y4
-------------------------------------
Y1*Y3 -X2.VO2*X2.CU3*Y1*Y4 -X1.VO1*X1.CU1*Y2*Y3
+X1.VO1*X1.CU1*X2.VO2*X2.CU3*Y2*Y4 -X1.VO1*X1.CU2*X2.VO2*X2.CU4*Y2*Y4
```

U zbývajících přenosů, tj. impedančního a admitančního, se výsledky shodují. Objevuje se u nich ovšem stejné matematické znázornění jako u napěťového přenosu.

# 5 Závěr

V této bakalářské práci byl vytvořen a popsán program pro analýzu chování autonomních obvodů, pomocí napěťových, proudových, impedančních a admitančních přenosů v symbolickém vyjádření.

V první kapitole byly popsány druhy analýzy a jejich metody, na kterých byl postaven základ programu.

Druhá kapitola je zaměřena okrajově na MATLAB. Obsahuje stručnou charakteristiku programu, jeho využití a základní práci.

Třetí kapitola obsahuje popis vstupních dat, který program zpracovává. Prvky, které jsou v analyzovaném obvodu použity. Vytvoření úplné admitanční matice obvodu, ze které se při výpočtu přenosových funkcí vychází. Dále jsou popsány přenosové funkce, které provádí povolené kombinace zapojení napěťových a proudových zdrojů a následné výpočty přenosů. V závěru kapitoly je popsáno ukládání těchto výsledků do textového souboru.

V poslední kapitole je prováděno srovnání výsledků s výsledky z programu SNAP.

Vytvořený program pracuje správně. Téměř všechny výsledky jsou shodné s výsledky ze SNAPu. Pouze u proudového přenosu se výsledky nepatrně liší, což je způsobeno nejasnou definicí proudového přenosu. Program lze dále rozšířit popř. upravit pro jiné obvody obsahující např. konkrétní typy konvejorů, setrvačné prvky (cívka, kondenzátor), nebo pro číselný výpočet. Jelikož jsou výsledky uloženy v textovém souboru, mohou být dále zpracovány.

Bakalářská práce byla tímto splněna.

# Literatura

[1] BIOLEK, Dalibor. *Řešíme elektronické obvody, aneb, Kniha o jejich analýze*. 1. vyd. Praha: BEN - technická literatura, 2004, 519 s. ISBN 80-730-0125-X.

[2] POSPÍŠIL, Jiří a Bohuslav DOŇAR. *Teorie elektronických obvodů*: *tvorba uživatelských aplikací*. 1. vyd. Brno: VUT, 1997, 200 s. ISBN 80-214-0936-3.

[3] ZAPLATÍLEK, Karel. *MATLAB pro začátečníky*. 2. vyd. Praha: BEN - technická literatura, 2005, 151 s. ISBN 80-730-0175-6.

[4] ZAPLATÍLEK, Karel a Bohuslav DOŇAR. *MATLAB*: *tvorba uživatelských aplikací*. 1. vyd. Praha: BEN, 2004, 215 s. ISBN 80-730-0133-0.

[5] KOTON, Jaroslav a Kamil VRBA. Zobecněné metody návrhu kmitočtových filtrů. *Zobecněné metody návrhu kmitočtových filtrů* [online]. 2008, roč. 2008, č. 26, s. 17 [cit. 2013-05-26]. Dostupné z: http://elektrorevue.cz/cz/download/zobecnene-metody-navrhu-kmitoctovych-filtru/.

[6] POLÁCH, Petr. *RC oscilátory pro pásmo vyšších kmitočtů: Oscillators RC for higher frequency range* [online]. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2008 [cit. 2013-05-26]. 1 elektronický optický disk [CD-ROM / DVD]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle/11012/7291/Diplomov%C3%A1%20pr%C3%A1ce.pdf?sequence=1. Diplomová práce. VUT.

[7] ČAJKA, Josef. *Teorie lineárních obvodů. Analýza lineárních a linearizovaných elektrických obvodů: analýza lineárních a linearizovaných elektrických obvodů*. 1. vyd. Praha;Bratislava: SNTL;Alfa, 1979, 355 s.

[8] KOTON, J a M MINARČÍK. Elektrorevue. *Využití grafů signálových toků pro analýzu obvodů s proudovými konvejory* [online]. 2006 [cit. 2013-05-30]. Dostupné z: http://www.elektrorevue.cz/clanky/06039/index.html

[9] ČAJKA, J a T DOSTÁL. Elektrorevue. *Nové názvosloví a sjednocující pohled na proudové konvejory* [online]. 2001 [cit. 2013-05-30]. Dostupné z: http://www.elektrorevue.cz/clanky/01024/index.html

# Seznam zkratek, veličin a symbolů

GCC – obecný proudový konvejor

KZ – Kirchhoffův zákon

MSP – metoda smyčkových proudů

MUN – metoda uzlových napětí

MMUN – modifikovaná metoda uzlových napětí

# Seznam příloh

# A Popis funkcí a práce s proměnnými v MATLABu

## A.1 Použití proměnných [3]

| Příkaz | Příklad použití | Význam |
|---|---|---|
| who | who | stručný přehled o použitých proměnných |
| whos | whos | detailní přehled o použitých proměnných |
| | whos A | detailní přehled o proměnné A (struktura aj.) |
| clear | clear<br>clear all | nenávratné vymazání všech proměnných z paměti |
| | clear A | nenávratné vymazání pouze proměnné A z paměti |
| save | save c:\Pokus A cislo | uložení proměnných *A* a *cislo* na disk do souboru se jménem *Pokus.mat*, pokud není specifikována cesta, jsou proměnné uloženy do implicitní složky, nejsou-li uvedena jména proměnných, uloží se všechny proměnné |
| load | load c:\Pokus | načtení dříve uložených proměnných v souboru *Pokus.mat*, lze také psát *load c:\Pokus A cislo* |

## A.2 Řetězce znaků [3]

| Příkaz | Příklad použití | Význam |
|---|---|---|
| zápis znakové proměnné | jmeno='Karel'<br>jmeno1=char('Bohuslav')<br>a='Ahoj Bamao';<br>b='ahoj Cvachto';<br>S=char(a,b)<br>B2=double(jmeno) | proměnné *jmeno* a *jmeno1* jsou typu *char array*, o čemž se lze přesvědčit pomocí *whos jmeno* |
| char | char(B1) | zpětný převod ASCII |
| str2num | text='123.456';<br>Z=str2num(text) | převod čísla, zapsaného jako řetězec, na číslo (double) |
| num2str | num2str(Z) | převod čísla (double) zpět na řetězec |
| strcmp | strcmp(text1,text2) | srovnání dvou proměnných, obsahující textové řetězce, vysledek je buď *0*, pokud řetězce nejsou zcela shodné nebo *1*, jsou-li shodné |
| porovnání pomocí relačních operátorů | Text1==Text2<br><br>Text~=Text2 | porovnání, zda jsou dva textové řetězce shodně a zda jsou rozdílné, výsledkem je vektor jedniček nebo nul podle toho, zda je daný znak shodný či nikoli |
| findstr | Nadpis='Mila maminko';<br>findstr(Nadpis,'m')<br>ans=6 8 | dotaz, zda se v textovém řetězci nalézá znak *m*, odezvou jsou pořadová místa znaku |
| find | find(Nadpis=='m') | obecnější příkaz na nalezení se stejnou odezvou |

## A.3 Práce s maticemi [3]

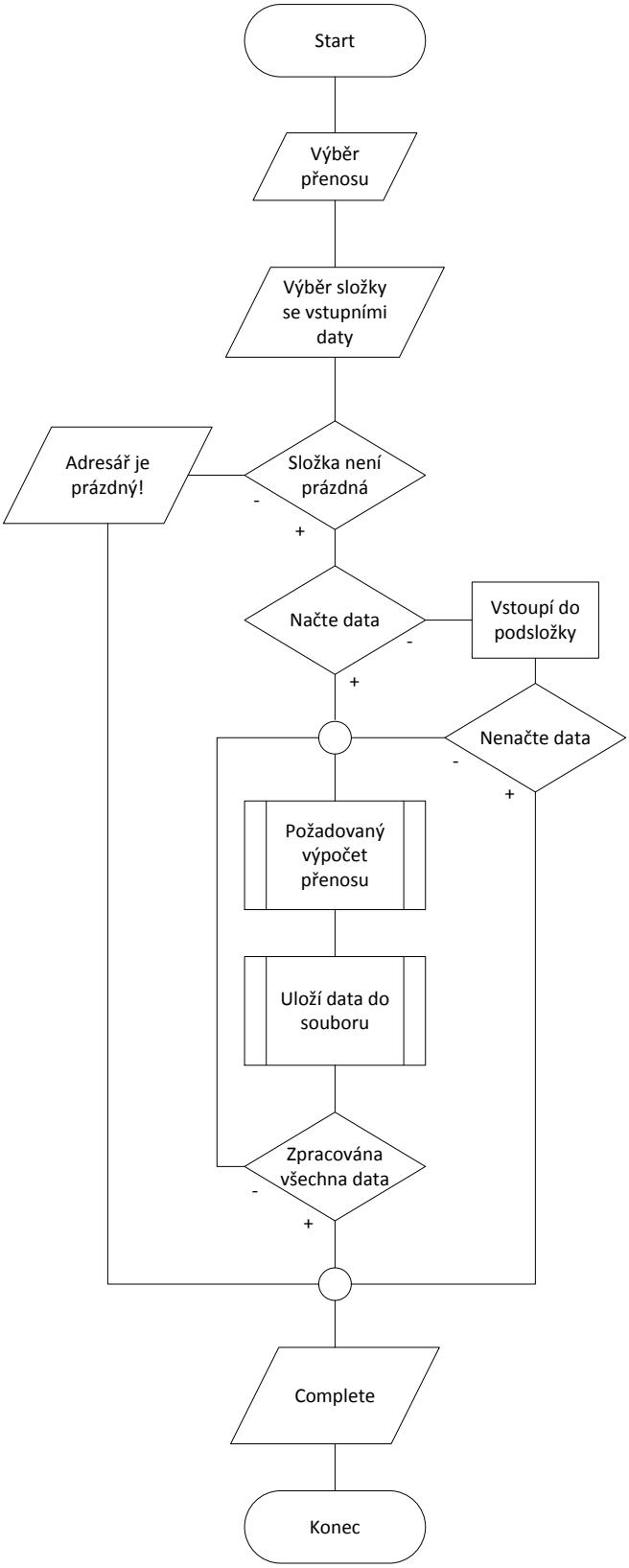
| Příkaz | Příklad použití | Význam |
|---|---|---|
| zápis matice | A=[1 2;3 4]<br>Vektor=[4 8 9 2.5 1+3i]<br>R22=[1 10;55 log10(100)] | matici zapisujeme po řádcích, jež jsou odděleny středníky, jednotlivé prvky v řádku oddělujeme mezerou nebo čárkou |
| tvorba delších vektorů či matic | t=0:4*pi/100:4*pi | vytvoření 101 hodnot (100 úseků) vektoru *t* |
| sdružování vektorů | V1=0:5;<br>V2=6:11;<br>A=[V1;V2] | tvorba matic *A* a *B* pomocí dvou vekotrů, *A* je matice, jejíž řádky jsou tvořeny vektory *V1* a *V2*, *B* je vektor, v němž jsou dva dílčí vektory v řádku za sebou |
| základní maticové funkce | A1=[1 2 3 ;4 5 6; 11 12 130]; | |
| det | det(A1) | determinant čtvercové matice |
| eig | D=eig(A1) | vrací vektor vlastních čísel čtvercové matice |
| max | G=max(A1) | vrací vektor, jenž obsahuje největší prvky každého sloupce původní matice, u vektorů vrací přímo jejich největší prvek |
| size | H=size(A1) | vrací vektor dvou čísel, jež udávají rozměr matice či vektoru (počet řádků a sloupců) |
| diag | I=diag(A1) | vrací vektor prvků na hlavní diagonále matice *A1* |
| isempty | J=isempty(A1) | vrací jedno číslo a to *0*, je-li matice *A1* neprázdná nebo *1*, je-li prázdná |
| ones | M=ones(3,2) | tvorba matice *M* daných rozměrů ze samých jedniček |
| zeros | Matnul=zeros(3,4) | tvorba matice samých nul zadaných rozměrů |

## A.4 Indexování matic a vektorů [3]

| Příkaz | Příklad použití | Význam |
|---|---|---|
| indexování vektorů | v=[16 5 9 4 2 11 7 14] | definice vektoru *v* |
| | v(3) | hodnota prvku vektoru *v* s pořadovým číslem 3 |
| | v2=v([3:7]) | tvorba vektoru *v2* tak, že bude obsahovat prvky vektoru *v* s pořadovými čísly *3* až *7* |
| | v3=v(1:2:end) | výpis prvků vektoru *v* s lichými indexy |
| | v(:) | výpis všech prvků vektoru *v* ve formě sloupcového vektoru |
| | v(:)' | výpis všech prvků vektoru v ve formě řádkového vektoru |
| indexování matic | A=magic(4) | definice matice |
| | A(2,4) | výpis prvku matice *A* se souřadnicemi *2,4* (řádek, sloupec) |
| | A(2:4,1:2) | výpis části matice *A* (řádek *2-4*, sloupec *1-2*) |

| | | |
|---|---|---|
| | A(:,end) | výpis posledního sloupce matice *A* (libovolný řádek, poslední sloupec) |
| | A(:,[2 4])=[] | "vymazání" *2.* a *4.* sloupce matice *A* (libovolný řádek, sloupec *2* a *4*), vzniká nová matice se čtyřmi řádky a jen dvěma sloupci |

# B Zjednodušený vývojový diagramu scriptu AnalysisCircuit



# C Obsah CD

CD obsahuje program (jeden script a pět funkcí) spolu se stručným popisem použití a jeho charakteristikou (uživatelská příručka.pdf).