KATEDRA ELEKTRICKÝCH
POHONŮ A TRAKCE

ČESKÉ VYSOKÉ UČENÍ
TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

VEKTOROVÉ ŘÍZENÍ ASYNCHRONNÍHO MOTORU S VYUŽITÍM NEURONOVÝCH SÍTÍ

DIPLOMOVÁ PRÁCE

SRPEN 2021

AUTOR: BC. JÁN ŽOLNA
VEDOUCÍ PRÁCE: ING. ONDŘEJ LIPČÁK

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Žolna** Jméno: **Ján** Osobní číslo: **453245**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrických pohonů a trakce**

Studijní program: **Elektrotechnika, energetika a management**

Specializace: **Elektrické pohony**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vektorové řízení asynchronního motoru s využitím neuronových sítí

Název diplomové práce anglicky:

Vector Control of Induction Motor Using Neural Networks

Pokyny pro vypracování:

1. Vysvětlete princip vektorové regulace asynchronního motoru společně s používanými matematickými modely.
2. Uveďte základní princip neuronových sítí a jejich klasifikaci dle vybraných kritérií.
3. Prozkoumejte možnost využití neuronových sítí v oblasti elektrických pohonů.
4. Vytvořte simulační schéma vektorového řízení asynchronního motoru v programu MATALB/Simulink využívající neuronové sítě.
5. Na vybraných průběžích demonstrujte funkčnost vašeho modelu.

Seznam doporučené literatury:

[1] ZEMAN, Karel et al. Automatická regulace pohonů s asynchronními motory. 1. vyd. Plzeň: Západočeská univerzita, 2004.
[2] CHAN, Tze-Fun a Keli SHI. Applied Intelligent Control of Induction Motor Drives.1. Aufl.;1;1st; Hoboken: Wiley, 2011.
[3] VAS, Peter. Sensorless Vector and Direct Torque Control. Oxford: Oxford University Press, 1998.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Ondřej Lipčák, katedra elektrických pohonů a trakce FEL

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

Datum zadání diplomové práce: **13.04.2021** Termín odevzdání diplomové práce: **13.08.2021**

Platnost zadání diplomové práce: **19.02.2023**

Ing. Ondřej Lipčák podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. | Datum převzetí zadání

Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

POĎAKOVANIE

Chcel by som poďakovať vedúcemu tejto práce, Ing. Lipčákovi. Bez jeho rád, pomoci a povzbudení by táto práca nevznikla. Tiež by som chcel poďakovať rodine a priateľom za podporu pri celom štúdiu.

PROHLÁŠENÍ

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

V Praze dne 13. augusta 2021

.....

ABSTRAKT

V tejto diplomovej práci budem skúmať využitie neurónových sietí pri vektorovej regulácii asynchrónneho motora a simulovať ich na modeli tohto pohonu. Neurónové siete budú najprv použité na odhadovanie elektrickej uhlovej rýchlosti rotora. Potom bude zostrojená bezsenzorová regulácia s použitím odhadnutých parametov.

V prvých kapitolách sú základné teoretické informácie o vektorovej regulácii a modeli asynchrónneho motora. Následne je priblížená štruktúra a fungovanie umelých neurónových sietí a ich učenie. Ďalej opisujem bezsenzorový odhad parametrov motora a použitie neurónových sietí v oblasti elektrických pohonov. Nasledujúce dve kapitoly opisujú implementáciu modelu vektorovej regulácie v prostredí Simulink a generovanie neurónových sietí v programe Matlab. V poslednej časti je rozbor zachytených priebehov odhadu uhlovej rýchlosti pomocou doprednej a rekurentnej neurónovej siete a následne implementované bezsenzorové riadenie pomocou odhadnutých parametrov.

Kľúčové slová: asynchrónny motor, vektorová regulácia, dopredná neurónová sieť, rekurentná neurónová sieť, bezsenzorová regulácia, model motora, odhad rýchlosti

ABSTRACT

In this diploma thesis, I will investigate the use of artificial neural networks in vector control of induction motor and simulate them on a model of this drive. At first, neural networks will be used to estimate the electrical angular velocity of the rotor. Later sensorless control will be constructed using the estimated parameters.

First chapters contain basic theoretical information about vector control and induction motor model. Subsequently, structure and principles of artificial neural networks and their training are explained. Then, I describe sensorless estimation of motor parameters and the use of neural networks in the field of electric drives. Following two chapters talk about implementation of vector control model in the Simulink environment and generation of neural networks in the Matlab program. In the last part, analysis of captured waveforms of the angular velocity using feedforward and recurrent neural network is performed and sensorless control using the estimated parameters is implemented.

Keywords: induction motor, vector control, feedforward neural network, recurrent neural network, sensorless control, motor model, speed estimator

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Elektrické pohony s asynchrónnymi motormi prešli sa posledných päťdesiat rokov rozsiahlym vývojom. Ten bol spôsobený tlakom na zvyšovanie ich účinnosti, presnejšie ovládanie a nižšiu cenu. Vektorové riadenie bolo predstavené pred viac ako štyridsiatimi rokmi a doteraz je často používané. Neskôr sa začala používať metóda priameho riadenia momentu (DTC). Obe tieto metódy potrebujú na vstup do regulátorov presne zmerané a vypočítané veličiny, či už sa jedná o uhlovú rýchlosť, magnetický tok, natočenie rotora, atď. Preto v posledných rokoch zaznamenala bezsenzorová regulácia veľký rozmach. Tá môže z jednej strany zvýšiť spoľahlivosť pohonného systému a z druhej strany znížiť cenu, a to odstránením otáčkového senzora.

Myslíme si, že bezsenzorová regulácia môže najviac pomôcť práve asynchrónnemu motoru. Oproti jednosmernému motoru je spoľahlivejší tým, že nemá komutátor a oproti synchrónnemu je lacnejší (kvôli jednoduchšej konštrukcii a lacnejším materiálom rotora), a tak odstránenie presného otáčkového senzora môže prispieť k atraktivite tohto pohonu.

Pri regulácii asynchrónneho motora používame mnoho parametrov, ktoré sú buď závislé na teplote alebo na pracovnom bode. Niektoré z nich sa nedajú merať alebo ich meranie je drahé a ťažko realizovateľné. Jedným z príkladov je rotorový odpor, ktorého hodnota sa zvyšuje spolu s tým, ako rastie teplota rotora. Neurónové siete v tejto aplikácii dokážu zefektívniť vektorovú reguláciu a tak zvýšiť účinnosť motora.

Cieľom diplomovej práce je otestovať odhadovanie otáčok asynchrónneho motora pomocou neurónových sietí v programe Matlab/Simulink. Prvé tri kapitoly slúžia na oboznámenie čitateľa s teoretickými základmi pre túto úlohu. Keďže sa asynchrónnymi motormi a neurónovými sieťami zaoberajú dva rôzne odbory, je málo ľudí, ktorí majú prehľad o oboch. Preto sme v týchto častiach priblížili obe oblasti. Vo väčšej miere sme sa ale venovali opisu neurónových sietí, keďže predpokladáme čitateľa, ktorý má viac vedomostí o pohonoch. V tretej kapitole sa zaoberáme bezsenzorovým meraním parametrov motora a využitím neurónových sietí v oblasti elektrických pohonov. Ďalšie kapitoly majú už praktický charakter. Štvrtá opisuje implementáciu vektorového riadenia a neurónových sietí v programe Simulink. V kapitole päť je opísaný postup generovania a optimalizácie neurónových sietí v Matlabe. Na tieto kapitoly potom nadväzuje šiesta, kde je opis a výsledky odhadu elektrickej uhlovej rýchlosti rotora pomocou neurónových sietí. Najprv sme pracovali s rozbehom nezaťaženého motora a odhadovali rýchlosť pomocou doprednej neurónovej siete. Neskôr sme použili rekurentnú sieť. Nakoniec sme do odhadu pridali záťaž a zmerali priebehy. V siedmej kapitole sú tieto odhadnuté parametre použité ako vstup do regulátora, čiže je tu implementovaná bezsenzorová regulácia.

V práci tiež uvádzame časti kódu z Matlabu a obrázky modelu zo Simulinku. Fungujúci projekt s týmito materiálmi je potom zahrnutý v prílohe.

KAPITOLA 1: ASYNCHRÓNNY MOTOR A JEHO REGULÁCIA

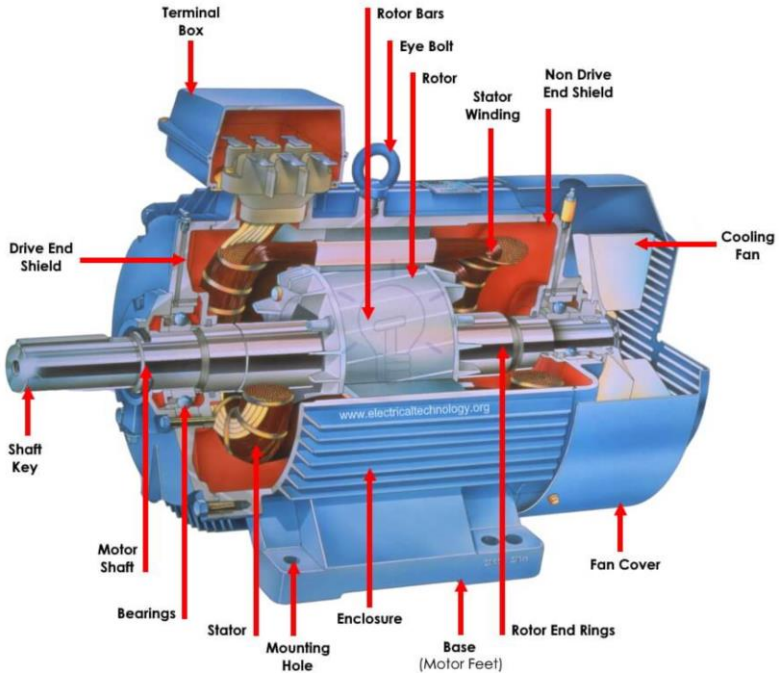
Asynchrónny motor (AM) je najviac používaný typ motora. Je to pre jednoduchosť jeho konštrukcie, ktorá je spojená s nízkymi nákladmi na výrobu, spoľahlivosť a veľký výkonový rozsah. Bol patentovaný pred viac ako 135 rokmi a za jeho vynájdením stojí Nikola Tesla. Používa sa prevažne v aplikáciách, kde potrebujeme konštantné otáčky, kvôli možnosti ho priamo pripojiť na napájaciu sieť. V prípade, že chceme vo väčšej miere a presnejšie riadiť rýchlosť, musíme AM použiť spolu s meničom.

Rozmedzie použitia je od elektrického náradia, cez ventilátory a čerpadlá, až po použitie v trakcii, kde najnovšie našiel uplatnenie v niektorých elektrických automobiloch. Táto aplikácia je jedna z najnáročnejších a hľadí sa tu mimo požadovaných výkonových parametrov aj na efektivitu, v ktorej AM zaostáva za synchrónnym motorom s permanentnými magnetmi. Ten je efektívnejší a oproti AM má aj väčší výkon na danú váhu a objem.

Hoci je vo viacerých parametroch horší ako synchrónny motor, v tejto práci sme sa rozhodli práve pre použitie AM. Dôvodom je jeho jednoduchšia regulácia a väčšie množstvo variabilných parametrov, ktoré ovplyvňujú správanie pohonu (napríklad odpor rotora). Ďalšie rozhodnutie bolo medzi typom regulácie. Najviac používané typy sú vektorové riadenie orientované na rotorový tok a DTC. Vybrali sme si vektorovú reguláciu.

1.1 Asynchrónny motor

Ako sme už spomínali, AM má jednoduchú konštrukciu. Konkrétne sa skladá z dvoch hlavných častí: stator a rotor. Stator je tvorený izolovanými plechmi, ktoré majú tvar prstenca. Ten má na vnútornej časti vyrezávané otvory na vinutie. Plechy sú uložené v konštrukcii. Na vnútornom obvode sa po spojení vytvoria drážky, do ktorých je vložené väčšinou trojfázové vinutie. To je vyvedené do svorkovnice, ktorá je navrhnutá tak, že vinutie motora môžeme zapojiť do hviezdy alebo trojuholníka (1 s. 77).



Obrázok 1 Rez asynchrónnym motorom (2)

Podľa typu rotora delíme AM na motor s klietkou a motor s rotorovým vinutím. Oba tieto vodiče sú uložené v konštrukcii z plechov. V prípade, ak je použité vinutie, to je potom napájané cez krúžky a kefy. Na rotorové vinutie tak môžeme pripojiť spúšťací odpor, ktorý obmedzuje záberný prúd a dosahujeme tak lepších spúšťacích charakteristík motora. Rotor je s konštrukciou spojený ložiskami (1 s. 77).

Princíp činnosti AM spočíva na vzájomnom pôsobení točivého magnetického poľa statora a prúdov v rotore, ktoré indukuje. Na stator je privedené (väčšinou) trojfázové striedavé napätie. To vďaka rozmiestneniu vinutia (cievky sú rovnako usporiadané, majú rovnaký počet závitov a voči sebe sú otočené o 120°) vytvorí točivé magnetické pole, ktoré buď v klietke alebo vinutí rotora naindukuje napätie, ktoré vyvolá prúdy a tie začnú pôsobiť proti príčine ich vzniku (Lenzov zákon). Moment, ktorý tam vznikne, začne točiť rotorom. Toto sa deje za predpokladu, že uhlová rýchlosť rotora je odlišná od uhlovej rýchlosti točivého magnetického poľa statora. Keby sa rotor otáčal synchrónne s poľom statora, indukovalo by sa na ňom nulové napätie, pretekal by tam nulový prúd a stroj by mal nulový moment (1 s. 75).

1.1.1 Matematický model

Pre zjednodušenie výpočtu na odvodenie matematických rovníc a v ďalších matematických modeloch predpokladáme (3 s. 111):

- napájacia sústava je trojfázová súmerná a všetky napätia sú harmonické,
- veľkosť vzduchovej medzery je konštantná,
- vinutia jednotlivých fáz sú sínusovo rozložené pozdĺž vzduchovej medzery v drážkach statora a rotora,
- odpory a indukčnosti jednotlivých fáz statora a rotora sú zhodné,
- motor pracuje v lineárnej časti magnetizačnej charakteristiky,
- straty v železe sú zanedbateľné.

Z týchto predpokladov môžeme vidieť, že sa jedná o motor, ktorý má trojfázové vinutie na statore a rotore. Rovnice nebudeme odvodzovať. Využijeme už odvodené vzorce z (4) a (5). S uvažovaním predchádzajúcich predpokladov sú odvodené rovnice pre asynchrónny motor s kotvou nakrátko. Z nich vypočítame rotorové a statorové prúdy v súradniciach $\alpha\beta$ (5 s. 19):

$$\frac{\mathrm{d}i_{1\alpha}}{\mathrm{d}t} = \frac{1}{L_1}\left(u_{1\alpha} - R_1 i_{1\alpha} - L_m \frac{\mathrm{d}i_{2\alpha}}{\mathrm{d}t}\right),$$

$$\frac{\mathrm{d}i_{1\beta}}{\mathrm{d}t} = \frac{1}{L_1}\left(u_{1\beta} - R_1 i_{1\beta} - L_m \frac{\mathrm{d}i_{2\beta}}{dt}\right), \qquad 1\text{-}1$$

$$\frac{\mathrm{d}i_{2\alpha}}{\mathrm{d}t} = \frac{1}{L_2}\left[-R_2 i_{2\alpha} - L_m \frac{\mathrm{d}i_{1\alpha}}{\mathrm{d}t} - \omega\left(L_2 i_{2\beta} + L_m i_{1\beta}\right)\right],$$

$$\frac{\mathrm{d}i_{2\beta}}{\mathrm{d}t} = \frac{1}{L_2}\left[-R_2 i_{2\beta} - L_m \frac{\mathrm{d}i_{1\beta}}{\mathrm{d}t} - \omega(L_2 i_{2\alpha} + L_m i_{1\alpha})\right]. \qquad 1\text{-}2$$

Na výpočet momentu použijeme rovnicu, ktorá ho vypočíta zo statorových a rotorových prúdov. V prípade, že máme vypočítaný moment, môžeme z rovnice 1-4 vypočítať mechanickú uhlovú rýchlosť, z ktorej po vynásobení počtom polpárov dostaneme elektrickú uhlovú rýchlosť. Rovnice majú potom tvar (5 s. 19):

$$M = \frac{3}{2} p_p L_m \left(i_{2\alpha} i_{1\beta} - i_{2\beta} i_{1\alpha}\right), \tag{1-3}$$

$$M - M_z = J \frac{\mathrm{d}\Omega}{dt}, \tag{1-4}$$

$$\omega = p_p \Omega. \tag{1-5}$$

1.2 Potrebné transformácie

1.2.1 Clarkovej transformácie

Keďže v modeli motora počítame v *αβ* súradniciach zviazaných so statorom, ale motor je napájaný z trojfázovej sústavy *abc*, budeme na prevod medzi týmito sústavami potrebovať Clarkovej transformáciu. Tá je daná nasledujúcim vzťahom, kde transformačný koeficient *K* je rovný 2/3 (4 s. 7):

$$\begin{pmatrix} x_\alpha \\ x_\beta \end{pmatrix} = \begin{pmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \\ 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix}. \tag{1-6}$$

Na opačný prevod použijeme inverznú Clarkovej transformáciu, ktorá prevádza jednotky z *αβ* do *abc* sústavy (4 s. 8):

$$\begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix} \begin{pmatrix} x_\alpha \\ x_\beta \end{pmatrix}. \tag{1-7}$$

1.2.2 Parkova transformácia

Regulátory momentotvornej a tokotvornej zložky prúdu vo vektorovej regulácii pracujú v sústave *dq* zviazanej s rotorovým tokom. No keďže rovnice pre AM sú v sústave *αβ,* budeme na prevod potrebovať Parkovu transformáciu. Obecný vzorec na prevod zo sústavy *xy* do *αβ* je (4 s. 10):

$$\begin{pmatrix} a_x \\ a_y \end{pmatrix} = \begin{pmatrix} \cos \upsilon_k & \sin \upsilon_k \\ -\sin \upsilon_k & \cos \upsilon_k \end{pmatrix} \begin{pmatrix} a_\alpha \\ a_\beta \end{pmatrix}, \tag{1-8}$$

kde υ_k je okamžitý uhol medzi týmito dvomi systémami, do ktorého potom dosadíme transformačný uhol θ. Na opačný prevod použijeme inverznú Parkovu transformáciu (4 s. 10):

$$\begin{pmatrix} a_\alpha \\ a_\beta \end{pmatrix} = \begin{pmatrix} \cos \upsilon_k & -\sin \upsilon_k \\ \sin \upsilon_k & \cos \upsilon_k \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix}. \tag{1-9}$$

1.3 In model motora

Nakoniec budeme potrebovať In model motora. Z neho dostaneme veľkosť vektora magnetického toku rotora. Jeho zložky v súradniciach $\alpha\beta$ vypočítame z rovníc (4 s. 15):

$$\frac{d\psi_{2\alpha}}{dt} = \frac{L_m R_2}{L_2} i_{1\alpha} - \frac{R_2}{L_2}\psi_{2\alpha} - \omega\psi_{2\beta},$$
$$\frac{d\psi_{2\beta}}{dt} = \frac{L_m R_2}{L_2} i_{1\beta} - \frac{R_2}{L_2}\psi_{2\beta} - \omega\psi_{2\alpha}. \qquad 1\text{-}10$$

Môžeme vidieť, že ako vstupy budeme potrebovať statorové prúdy a elektrickú uhlovú rýchlosť. Tieto parametre budeme najprv brať z modelu asynchrónneho motora, no neskôr budeme používať odhadovanú uhlovú rýchlosť. Na veľkosť vektora toku použijeme vzorec:

$$\left|\overrightarrow{\psi_2}\right| = \sqrt{\psi_{2\alpha} + \psi_{2\beta}} \qquad 1\text{-}11$$

Z In modelu motora môžeme nakoniec vypočítať aj transformačný uhol θ. Je to uhol natočenia vektoru magnetického toku rotora a vypočítame ho zo zložiek toku pomocou goniometrickej funkcie. Vzťah pre transformačný uhol je (4 s. 20):

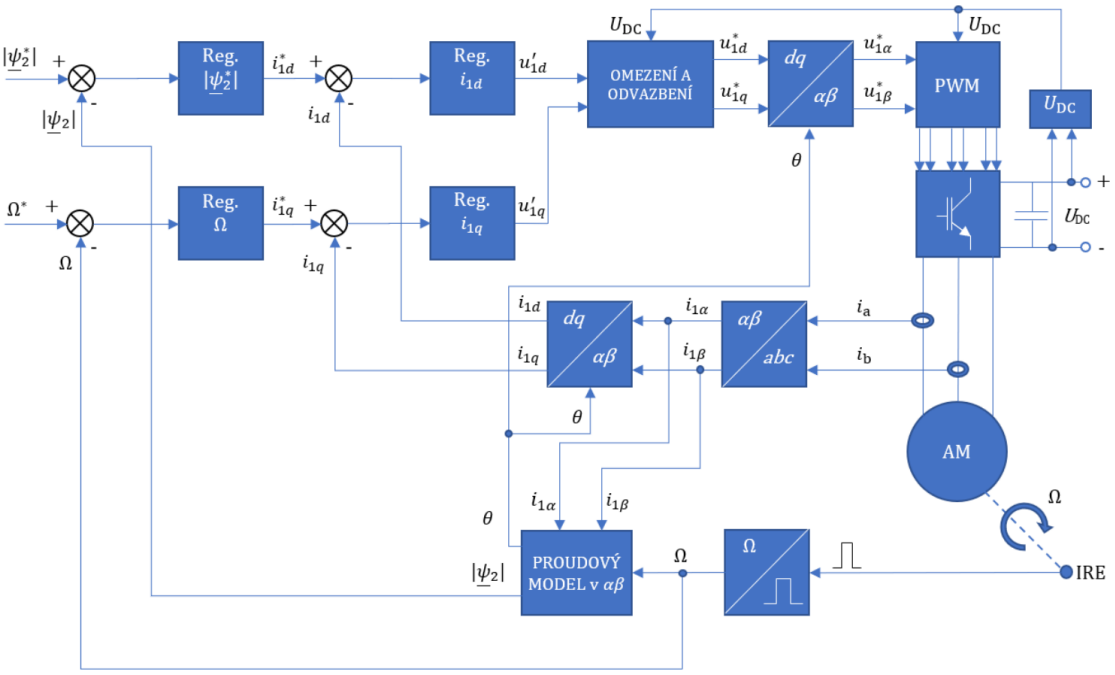
$$\theta = \arctan\frac{\psi_{2\beta}}{\psi_{2\alpha}} \qquad 1\text{-}12$$

1.4 Vektorová regulácia asynchrónneho motora

Vektorová regulácia bola navrhnutá tak, aby napodobnila reguláciu cudzo budeného jednosmerného motora. Rozkladáme tak statorový prúd na 2 zložky: tokotvornú a momentotvornú. Vlastnosti vektorovej regulácie sú (6):

- oddelené riadenie magnetického toku a momentu,
- pri konštantnom magnetickom toku by mal byť moment priamo úmerný momentotvornej zložke prúdu,
- v ustálenom stave by mali byť regulované veličiny jednosmerné,
- moment by mal byť úmerný sklzu.

Na Obrázok 2môžeme vidieť, že zvonku do vektorovej regulácie vchádzajú požadované hodnoty rotorového toku a elektrickej uhlovej rýchlosti. Výstupy z týchto regulátorov potom používame v regulátoroch statorového prúdu. Z týchto regulátorov potom vystupujú požadované hodnoty momentotvornej a tokotvornej zložky statorového napätia. Blok odväzbenia nebude implementovaný v tejto práci a na správny chod úlohy nemá vplyv. Výstupy z regulátora transformujeme z *dq* suradníc do $\alpha\beta$ pomocou inverznej Parkovej transformácie. AM potom napájame cez napäťový striedač. Pulzy doň generujeme blokom modulátora. Na AM potom meriame statorové prúdy, ktoré transformujeme z *abc* súradníc do *dq* cez Clarkovej a Parkovu transformáciu a použijeme ich v In modeli motora. Na motore tiež meriame otáčky pre In model a pre regulátor.



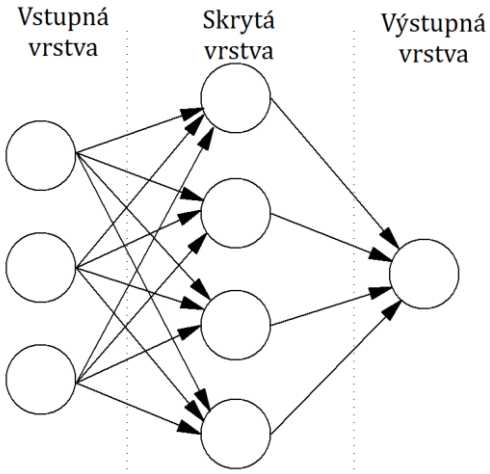
Obrázok 2 Bloková schéma vektorovej regulácie (4)

KAPITOLA 2: NEURÓNOVÉ SIETE

V tejto kapitole sa budeme venovať obecnému popisu neurónových sietí, ich typom a použitiu. Informácie mali slúžiť na pochopenie základného princípu fungovania siete a tiež sa tu definujú názvy používané v odbore neurónových sietí. Využitiu neurónových sietí v pohonoch sa potom venuje Kapitola 3:.

Neurónová sieť (artificial neural network ANN) (NS) je výpočtový systém inšpirovaný mozgom. Jej zjednodušený model je na nasledujúcom obrázku. Je založená na zhluku prepojení a uzlov. Tie sú navzájom pospájané. Každý uzol (neurón) môže prijať signál od iného neurónu, spracovať ho a poslať ďalej. Spojenia medzi neurónmi sa nazývajú hrany. Jednotlivé neuróny a hrany môžu a typicky majú váhu, ktorá sa upravuje počas učiaceho procesu. Väčšinou sú neuróny rozdelené do vrstiev, ktoré delíme na (7):

- **Vstupná vrstva** slúži na spracovanie vstupov do NS.
- **Skrytá vrstva** berie dáta z predchádzajúcej vrstvy (vstupnej/skrytej), spracuje a posunie ich do ďalšej vrstvy (výstupnej/skrytej). Skrytá vrstva neprodukuje priamo výstupné dáta. NS môže mať aj nula skrytých vrstiev.
- **Výstupná vrstva** slúži na predikciu výsledku.



Obrázok 3 Jednoduchý model neurónovej siete

Proces práce s NS začína trénovaním siete, potom je sieť otestovaná a nakoniec sa implementuje. Na začiatku tréningu majú váhy náhodné hodnoty. Tieto hodnoty sa upravujú počas celého tréningového procesu. Implementácia NS sa skladá z nasledujúcich krokov (8):

1. Zber dát.
2. Separácia tréningových a testovacích dát.
3. Výber architektúry siete.
4. Nastavovanie parametrov a inicializácia váh.
5. Transformovanie dát.
6. Trénovanie.
7. Testovanie.
8. Implementácia.

2.1 Základné pojmy

Pre pochopenie ďalšieho textu budú potrebné tieto definície:
Neuróny: NS sa skladá z uzlov (neurónov). Každý neurón môže mať viacero vstupov, ktoré vyprodukujú jediný výstup. Ten sa môže poslať na viacero neurónov (7).
Prepojenia a váhy: NS obsahuje aj prepojenia, ktoré spájajú neuróny navzájom. Každému pripojeniu je pridelená váha, ktorá udáva dôležitosť spojenia. Každý neurón môže mať viacero vstupných a výstupných prepojení. Ak je váha nulová, daný neurón nemá žiadny vplyv na druhý neurón (7).
Perceptron: je jednoduchý neurón, ktorého výstup je 1, ak súčet vstupov presahuje danú hranicu, inak je výstup 0 (9).
Propagačná funkcia: spočíta vstup do neurónu ako vážený súčet výstupov z predchádzajúcich neurónov (7).
Hyperparameter: je konštantný parameter, ktorého hodnota je určená pred učiacim procesom. Môže to napríklad byť počet skrytých vrstiev alebo rýchlosť učenia (7).
Rýchlosť učenia (learning rate): je konštanta, ktorá ovplyvňuje čas a presnosť naučenia NS. Určuje veľkosť zmien váh. Ak sú zmeny príliš veľké, môže algoritmus oscilovať a dôjsť tak k chybe, ktorá vedie k zle naučenej sieti. V prípade, že zmenšíme rýchlosť učenia, vedie to k zvýšeniu počtu krokov na dosiahnutie požadovaného výsledku a tak k zvýšeniu času trénovania (9).
Spätné šírenie (backpropagation): metóda, ktorá sa používa na upravenie váhy prepojení, aby sa kompenzovala každá chyba nájdená počas učenia (7).
Epocha: určuje stav prechodu informácií zo vstupu na výstup NS. Počas epochy sa upravujú váhy. Počet epoch určuje čas učenia (9).

Učenie a aktivačné funkcie sú obšírnejšie pojmy, a preto sú popísané v ďalších častiach.

2.1.1 Učenie

Učenie je adaptácia NS na lepšie zvládanie danej úlohy. Nazýva sa aj trénovanie v odbore umelej inteligencie a v odbore analýzy dát to môžeme nazvať regresia. Pri učení sa upravujú váhy a pracuje sa na tom, aby sa minimalizovala výsledná chyba. Tá skoro nikdy nedosahuje nulu (záleží od komplexnosti problému). V prípade, že je chyba väčšia ako očakávaná, musí sa NS upraviť. Modely učenia môžu byť vnímané ako aplikovanie optimalizačnej teórie a štatistického odhadu (10). Rozoznávame tri typy učenia:

- **Učenie s dohľadom** (supervised learning): používa súbor spárovaných vstupov a chcených výstupov. Úlohou tohto učenia je produkovať požadované výstupy pre každý vstup. Tento typ učenia bude použitý v tejto práci (9).
- **Učenie bez dohľadu** (unsupervised learning): predstavuje typ strojového učenia, kde NS nepozná „správnu odpoveď". Na rozdiel od učenia s dohľadom nedostane páry vstupov a požadovaných výstupov, ale sieť dostane vstupy a má v nich hľadať zaujímavé vzory, pravidelnosť, alebo podobnosti medzi nimi (9).
- **Posilňované učenie** (reinforcement learning): od učenia s dohľadom sa líši v tom, že nepotrebuje páry vstupov a výstupov, a v tom, že neoptimálne výsledky nemusíme viac optimalizovať. Naopak, cieľ je nájsť balans medzi prieskumom (nepoznaných vecí) a využitím (doterajších znalostí) (11).

2.1.2 Aktivačné funkcie

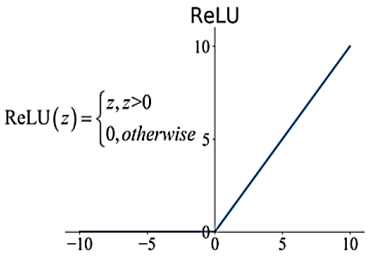
Výberom aktivačnej funkcie (AF) ovplyvňujeme kvalitu učenia NS na daných tréningových dátach. Jej voľbou vo výstupnej vrstve tiež definujeme, aký typ predikcie náš model dokáže urobiť. AF má takto veľký dopad na konečné schopnosti a výkon NS.

Funkcia určuje, ako sa váha súčtu vstupov do neurónu transformuje na výstup z neurónu v danej vrstve siete. Pri výpočte je použitá až po vnútornom spracovaní vstupov v neuróne. Typicky všetky skryté vrstvy používajú rovnakú AF a naopak výstupná vrstva väčšinou používa odlišnú. Existuje veľa druhov AF, no v praxi sa ich používa len zopár (12).

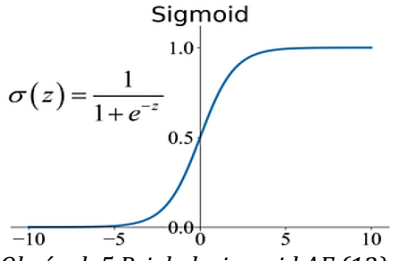
2.1.2.1 Aktivačné funkcie v skrytých vrstvách

V skrytých vrstvách väčšinou používame diferencovateľné nelineárne AF. Tie umožňujú, aby sa model natrénoval na komplexnejšie priebehy. Medzi najpoužívanejšie funkcie patria (12):

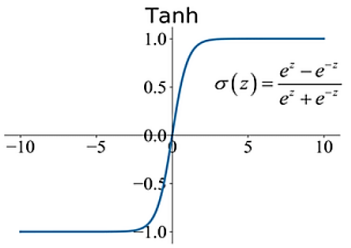
- **Relu** (Rectified Linear Activation): je pravdepodobne najpoužívanejšia funkcia. Oproti ostatným funkciám je odolnejšia voči problému miznúceho gradientu. Matematicky je daná tak, že pre záporné x vracia 0 a pre kladné vracia x.
- **Sigmoid** (Logistic): je typ funkcie, ktorá vstupy (reálne čísla) projektuje na interval 0 až 1. Čím väčší je vstup, tým bližšie bude k 1 a čím je menši tým bude bližšie k 0. Počíta sa podľa vzorca $1/(1+e^{-x})$.
- **Tanh** (hyperbolický tangens): je funkcia podobná sigmoid funkcii a má podobný S-tvar, no výstupom sú hodnoty medzi -1 a 1. Takže čím väčší je výsledok, tým je bližší k 1 a čím je menší, bude bližšie k -1. Počíta sa podľa vzorca $(e^{x}-e^{-x})/(e^{x}+e^{-x})$.



Obrázok 4 Priebeh ReLu AF (13)



Obrázok 5 Priebeh sigmoid AF (13)



Obrázok 6 Priebeh tanh AF (13)

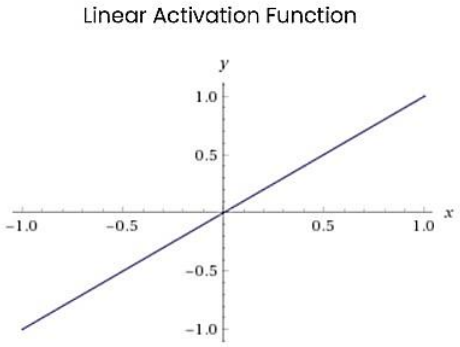
Ako už bolo spomínané, NS väčšinou používajú rovnakú AF pre všetky skryté vrstvy. V 90. rokoch minulého storočia bola štandardom sigmoid funkcia, no neskôr ju nahradila AF tanh. Problémom ale bolo, že obe tieto funkcie robia NS náchylnú na chyby počas trénovania, dôvodom je miznúci gradient. Výber AF v skrytých vrstvách väčšinou závisí od typu NS. Dopredné a konvolučné NS väčšinou používajú Relu funkciu, no v rekurentných je najviac používaná sigmoid alebo tanh funkcia. V niektorých rekurentných funkciách sa dokonca používajú obe zároveň. Napríklad LSTM (bude vysvetlená neskôr) používa sigmoid funkciu na rekurentné prepojenia a tanh na výstupy (12).

2.1.2.2 Aktivačné funkcie vo výstupných vrstvách

Keďže je výstupná vrstva priamo produkuje výstup, majú ju všetky NS. V tejto vrstve sú najpoužívanejšie AF (12):

- **Lineárna**: je funkcia, ktorá nemení hodnotu a priamo vracia vstupnú hodnotu na výstup. Takže funkcia pre x vracia x.
- **Sigmoid** (Logistic): bola priblížená v predchádzajúcom odseku.

- **Softmax**: vracia vektor hodnôt, ktorých súčet je 1. O tejto funkcii sa dá povedať, že dáva pravdepodobnosť, do ktorej triedy patrí výsledok. Takže cieľová hodnota bude blízka 1 a ostatné budú blízke 0. Počíta sa podľa vzorca $e^x/\sum(e^x)$. Táto funkcia sa nedá graficky znázorniť.



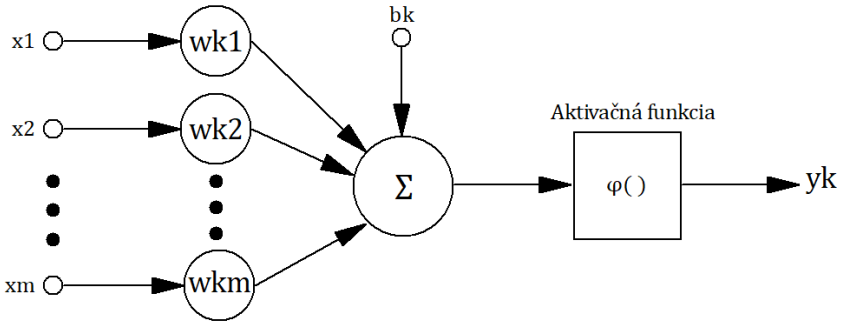
Obrázok 7 Priebeh lineárnej AF (14)

AF vo výstupnej vrstve sa vyberá podľa typu predikcie výstupnej vrstvy. V praxi riešime len dva problémy. Prvý je problém regresie (odhadujeme číselnú hodnotu) a tie odhadujeme pomocou lineárnej AF. Na problémy klasifikácie (pravdepodobnosť, binárna klasifikácia, atď.) použijeme ostatné funkcie. Keďže v týchto úlohách riešime regresný problém, všade budeme používať lineárnu AF (12).

2.2 Princíp fungovania neurónovej siete

S použitím pojmov z predchádzajúcej časti môžeme pokračovať s vysvetlením princípu fungovania NS. Ako prvé je potrebné vysvetliť fungovanie jedného neurónu, ktorý tvorí základnú časť siete.

Ten funguje tak, že vstupné hodnoty sú najprv vynásobené váhami konkrétnych spojení. Vynásobené hodnoty potom vstupujú do propagačnej funkcie, ktorá ich sčíta a pripočíta k nim prahovú hodnotu (bias). Táto prahová hodnota posúva AF pridaním konštanty, má teda analogickú úlohu ako pridanie konštanty k lineárnej funkcii. Výsledok je potom vstupom do vybranej aktivačnej funkcie, ktorá produkuje buď vstupy do neurónov v ďalších vrstvách alebo konečný výstup. V takom neuróne pri konfigurácii NS vyberáme aktivačnú funkciu na začiatku trénovania a počas neho sa upravujú váhy prepojení a prahová hodnota. Schéma neurónu je znázornená na ďalšom obrázku (15).



Obrázok 8 Schéma neurónu

Matematicky môžeme výstup takéhoto neurónu zapísať ako (16):

$$y_k = \varphi\left(\sum_{j=1}^{n}\left(w_{kj}x_j\right) + b_k\right) \qquad 2\text{-}1$$

kde y je výstupný signál, φ je AF, n je počet vstupov do neurónu, w_j je váha priradená j-tému spojeniu, x_j je j-tý vstupný signál a b je prahová hodnota.

Tieto neuróny sú potom usporiadané vo vrstvách, ktoré boli opísané v predchádzajúcich častiach. Dizajn neurónovej siete znamená výber počtu neurónov v každej vrstve, počtu vrstiev a typov spojení medzi nimi. Výber počtu neurónov a vrstiev je založený na intuícii a neexistuje k tomu definovaný štandard (10).

Existujú dva typy spojení medzi uzlami. Prvý typ sú jednosmerné spojenia bez spätnej väzby. Druhý sú spojenia so spätnou väzbou, kde výstup z uzlov môže byť vstupom do uzlov v rovnakej vrstve. Na základe týchto spojení môžeme NS klasifikovať do dvoch typov: dopredné a rekurentné NS. Tieto typy budú opísané v ďalších častiach (17).

Po navrhnutí NS môžeme začať tréning. Na začiatku tréningu sa váham priradia náhodné hodnoty a počas neho sú upravované na dosiahnutie požadovaného výsledku. Váhy sa nastavujú tak, aby sa minimalizovala chyba medzi výstupom z NS a správnym výsledkom.

Osobitnú pozornosť musíme dať na výber správnych tréningových dát. Ideálne tréningové dáta musia dobre reprezentovať podstaty systému. Dáta, ktoré to nesplňujú, vedú k nespoľahlivej NS. Jeden z častých problémov pri učení je „preučenie“(overfitting). To sa stane, buď keď učenie trvá príliš dlho, alebo súbor tréningových dát je príliš malý na pokrytie všetkých vzorov. Môžeme to pozorovať v prípade, keď sa výkon NS zlepšuje na tréningových dátach a zhoršuje na predtým neotestovaných testovacích dátach. Ako riešenie sa používa rozdelenie dát na tréningové a overovacie. Overovacie dáta tak ukazujú chyby nezávisle od testovacích dát. Podľa štúdie (18) sa zistilo, že ideálny pomer je použitie 80% pre tréningové dáta a 20% pre testovacie (19).

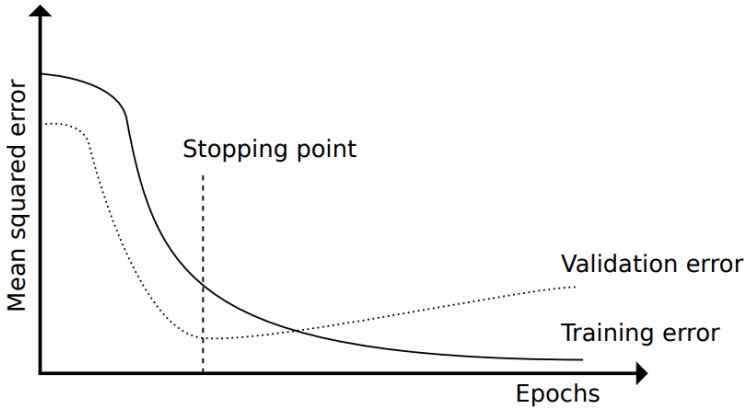
NS sa dá považovať za naučenú, keď dosiahne cieľovú chybu (target performance) na tréningových dátach. Na jej zistenie sa používa mnoho metrík, no v tejto práci priblížime a budeme používať metódu strednej kvadratickej chyby (mean squared error). Na ňu potrebujeme rovnicu pre chybu jedného neurónu (10):

$$e_k = y_{k,n} - y_k^* \qquad 2\text{-}2$$

kde e_k je chyba, $y_{k,n}$ je výstup z NS a y_k^* je požadovaná hodnota. Z nej vypočítame strednú kvadratickú chybu podľa vzorca (17):

$$E = \frac{1}{n}\sum_{k=1}^{n}(e_k)^2 \qquad 2\text{-}3$$

kde E je stredná kvadratická chyba a n je počet neurónov vo výstupnej vrstve. Táto chyba sa počíta pri každej epoche. Výpočet sa zastaví, keď sa dosiahne cieľová chyba, alebo keď chyba na overovacích dátach je na minime.



Obrázok 9 Závislosť strednej kvadratickej chyby na počte epoch (17)

Čím rýchlejšie sa dostaneme k výslednej chybe, tým kratšie bude trvať trénovanie NS. Preto boli na čo najoptimálnejšiu minimalizáciu chyby navrhnuté algoritmy. Medzi najpoužívanejšie patria: gradientový zostup s momentom, Levenberg-Marquardtov algoritmus (použitý v tejto práci) a Bayesova regularizácia (20 s. 1247).

2.3 Typy neurónových sietí

Existuje veľa typov neurónových sietí, a keďže je tento odbor stále vo vývoji, v budúcnosti ich bude ešte viac. Medzi jedno zo základných rozdelení patrí rozdelenie podľa typu prepojení v sieti. Tu existujú dva hlavné druhy, a to: dopredné NS, ktoré majú len dopredné prepojenia a rekurentné NS, ktoré obsahujú aj spätné väzby. Potom existuje aj modulárna NS (modular neural network), v ktorej viacero nezávislých NS prispieva k celkovému výsledku (7).

2.3.1 Dopredná neurónová sieť

Dopredná NS (feed-forward neural network) je taká sieť, v ktorej signály sa šíria len smerom dopredu od vstupnej po výstupnú vrstvu. Gradient chyby sa tu počíta cez algoritmus spätného šírenia (backpropagation algorithm). Tieto siete dobre zvládajú prípady s veľkým počtom vstupov a tiež dobre odolávajú šumu.

Jednovrstvová dopredná NS (Single-Layer Feedforward Network) je sieť, ktorá neobsahuje skrytú vrstvu. „Jednovrstvová“ odkazuje na výstupnú vrstvu, pretože vstupnú vrstvu, kvôli tomu, že sa tam nedeje žiaden výpočet, nepočítame (21).

Mnohovrstvová dopredná NS (Multilayer Feedforward Network) obsahuje aspoň jednu skrytú vrstvu. Takéto siete sa označujú m-h_1- h_2-...- h_n-q, kde m označuje počet vstupov, h_i počet neurónov v i-tej skrytej vrstve z n a q označuje počet výstupných neurónov (21).

Výhody:

- menej komplexné, jednoduché na dizajn a údržbu,
- rýchle (propagácia jedným smerom),
- veľká odozva na zašumené dáta.

Nevýhody

- nemôžu byť použité na hĺbkové učenie (deep learning) (kvôli tomu, že nemajú husté vrstvy a spätné učenie) (22).

Jedným z ďalších typov zložitejšej doprednej NS je konvolučná NS (convolutional neural network), ktorá bola navrhnutá na spracovanie obrazu, no nebudeme ju používať v tejto práci.

2.3.2 Rekurentná neurónová sieť

Rekurentná NS (recurrent neural network) je taká sieť, ktorá obsahuje neuróny so spätnou väzbou do predchádzajúcej vrstvy. Neuróny tak slúžia ako malé pamäťové bunky (21).
Výhody:

- dobré modelovanie sekvenčných dát, kde predpokladáme, že každá vzorka je závislá na predchádzajúcich.

Nevýhody:

- problém miznúceho a explodujúceho gradientu,
- trénovanie rekurentných NS môže byť náročné,
- zložité spracovanie dlhých sekvencií v prípade, že používame ReLu AF (22).

Jedným z príkladov zložitejšej rekurentnej NS je NS typu LSTM (long short-term memory neural network), ktorá rieši problém miznúceho gradientu v prípade týchto sietí. Majú oveľa zložitejšiu štruktúru a používa sa hlavne v prípadoch, kde si potrebujeme pamätať kontext (napríklad umelá inteligencia, ktorá rozumie zmyslu textu v prekladačoch).

2.4 Aplikácie neurónových sietí

NS sa používajú vo veľkom množstve oblastí. Sú ideálne v aplikáciách, kde máme málo vedomostí o výsledku a z veľkého množstva často zašumených dát musíme vybrať informácie, ktoré budú viesť k presnému výsledku. NS sú najrozšírenejšie v rozpoznávaní hlasu a obrazu. Tiež sú schopné vytvárať simulácie a predikcie pre komplexne systémy a vzťahy, ako napríklad predpoveď počasia a medicínske predpovede. Používajú sa napríklad pre (23):

- rozpoznávanie obrazu,
- rozpoznávanie hlasu,
- rozpoznávanie vzorov,
- syntéza hlasu,
- riadenie komplexných procesov,
- predpovede pre komplexné systémy,
- analýza časových dejov,
- preklad textu,
- simulácia komplexných systémov,
- biometrické systémy,
- ekonomické modely, atď.

KAPITOLA 3: BEZSENZOROVÉ MERANIE A MOŽNOSTI VYUŽITIA NEURÓNOVÝCH SIETÍ V OBLASTI ELEKTRICKÝCH POHONOV

3.1 Bezsenzorové meranie

Výraz bezsenzorová vektorová regulácia AM môžeme interpretovať ako vektorová regulácia bez otáčkového senzora. Pomenovanie nie je celkom presné, pretože bezsenzorové pohony používajú senzory. Tie merajú statorové prúdy a ovládacie napätia, ktoré sa potom používajú v regulačných algoritmoch. Sú však umiestnené mimo motora najčastejšie pri výkonovej elektronike (24).

Otáčkový senzor je naopak pripevnený na hriadeli motora (väčšinou inkrementálny senzor optického typu alebo resolver) a do regulátora posiela spätnú väzbu, či už o otáčkach alebo pozícii rotora. Výrobcom pohonov môže pomôcť odstránenie tohto senzora, pretože zvyšuje cenu a pridáva problémy so spoľahlivosťou, naviac sa pre jeho montáž musí predĺžiť hriadeľ a skonštruovať uloženie v motore. (25 s. 388)

Medzi hlavné techniky bezsenzorovej regulácie asynchrónneho motora patria (25) (26):

- výpočet sklzu (otáčky zo sklzovej uhlovej rýchlosti),
- odhad s použitím sledovaných statorových prúdov/napätí,
- model reference adaptive system (MRAS),
- rozšírený Kalmanov filter (Extended Kalman filter EKF),
- otáčkovo adaptovateľný pozorovateľ toku (Luenberger observer),
- injekcia pomocného signálu na vystupujúci rotor,
- odhad s použitím umelej inteligencie (neurónové siete, systémy založené na fuzzy logika, atď.).

Z predchádzajúcich príkladov môžeme vidieť, že k bezsenzorovému meraniu pristupujeme z dvoch smerov. Prvým je odhad, ktorý počíta danú veličinu v doprednej forme. Druhou technikou je pozorovateľ. Ten je viac sofistikovaný a má určité funkcie, ktoré sa dokážu samé prispôsobovať (24 s. 179).

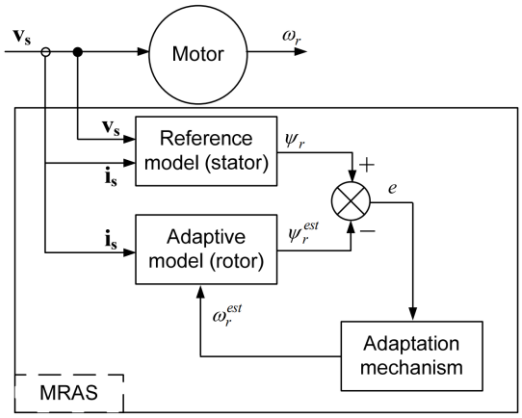
Detailný popis týchto metód je mimo rozsahu tejto práce, no v ďalších častiach priblížime zopár najpožívanejších metód a detailnejšie sa budeme venovať použitiu neurónových sietí, ktoré sú predmetom tejto práce.

3.1.1 MRAS

MRAS alebo model reference adaptive system (adaptívny pozorovateľ s referenčným modelom) v spojení s odhadom otáčok je založený na princípe, kde sa počíta chybový vektor z výstupov dvoch modelov. Každý z týchto modelov je závislý na rôznych parametroch motora. Upravovaním parametrov, ktoré ovplyvňujú jeden z modelov, je chyba zmenšovaná k nule. Základná schéma MRAS je na Obrázok 10 Schéma MRAS. Oproti Kalmanovým filtrom a ostatným pozorovateľom má výhodu v jednoduchosti použitých modelov. Odhad otáčok pomocou MRAS schémy môže byť napríklad založený na (28):

1. chybe rotorového toku,
2. spätnej elektromotorickej sile (back EMF),
3. chybe statorových prúdov.

Táto metóda teda funguje na princípe dvoch modelov. Medzi referenčným a nastaviteľným modelom sa vypočíta chyba, ktorá potom cez adaptačný mechanizmus (väčšinou PI regulátor) upravuje nastaviteľný model.

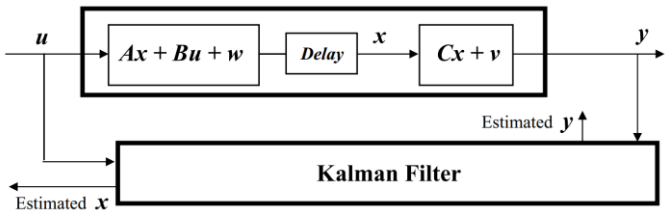


Obrázok 10 Schéma MRAS (27)

3.1.2 Rozšírený Kalmanov filter

Rozšírený Kalmanov filter (Extended Kalman filter EKF) sa používa na odhad parametrov motora len meraním statorových prúdov a napätí. Počas odhadovacieho procesu nie sú tieto signály filtrované. Presnosť odhadu otáčok závisí na tom, ako sa menia parametre motora, a na presnosti zmeraných prúdov a napätí. Princíp EKF spočíva na odhade stochastického stavu, ktorý je rekurzívne optimálny (recursive optimum stochastic state estimator) na nelineárnom dynamickom systéme v reálnom čase použitím zašumených signálov. Hlavné kroky k odhadu pomocou nespojitého EKF sú (29):

1. Výber modelu AM v časovej oblasti.
2. Diskretizácia tohto modelu.
3. Určenie stavových a kovariačných matíc.
4. Implementácia diskrétneho EKF algoritmu.
5. Nastavovanie kovariačných matíc.



Obrázok 11 Schéma základného Kalmanového filtra (30)

3.2 Neurónové siete v pohonoch

V tejto časti sa budeme venovať metódam využitia umelej inteligencie v elektrických pohonoch. Tieto riešenia môžu priniesť zvýšenie výkonu, širší rozsah nastavení a sú schopné sa prispôsobovať podmienkam. Príklady využitia sú (26):

- Nahradenie otáčkových, pozičných a iných regulátorov regulátormi založenými na umelej inteligencii (UI) (hybridné riešenia).
- Generovanie spínacích impulzov a spínacích vektorových schém pomocou UI.
- Odhadovanie rýchlosti, pozície, toku a momentu pomocou UI.
- Sledovanie stavu a diagnostika pomocou UI.
- Odhad harmonických založený na UI.

- Optimalizácia účinnosti založená na UI.
- Dizajn pohonných jednotiek s pomocou UI.

V klasických regulovaných systémoch je nutné mať informácie o systéme vo forme algebrických a diferenciálnych rovníc, ktorý analyticky spojujú vstupy s výstupmi. Lenže tieto modely môžu byť veľmi komplexné, spoliehať sa na veľa predpokladov a môžu obsahovať parametre, ktoré sa ťažko merajú alebo sa zásadne menia počas fungovania. (20) Tieto problémy rieši použitie systémov založených na inteligencii. Metódy založené na NS sú jednoducho aplikovateľné a nie sú závislé na zmene parametrov stroja (neplatí to pre túto úlohu, pretože to testujem na matematickom modeli). Medzi výhody pohonov založených na neurónových sieťach patria (31):

- NS sú rýchlejšie ako iné algoritmy vďaka ich paralelnej štruktúre.
- NS nepotrebujú výsledok žiadneho matematického modelu.
- Porucha jedného neurónu má len malý efekt na chod systému.
- V prípade, že je NS správne naučená, produkuje výsledok s minimálnou chybou.
- NS nie sú závislé na parametroch, takže zmeny v parametroch neovplyvňujú výsledok.
- NS sa dajú jednoducho aplikovať na ostatné stroje so striedavým napájaním.

Všeobecne na odhadovanie parametrov a reguláciu pohonov používame tieto systémy založené na inteligencii (20):

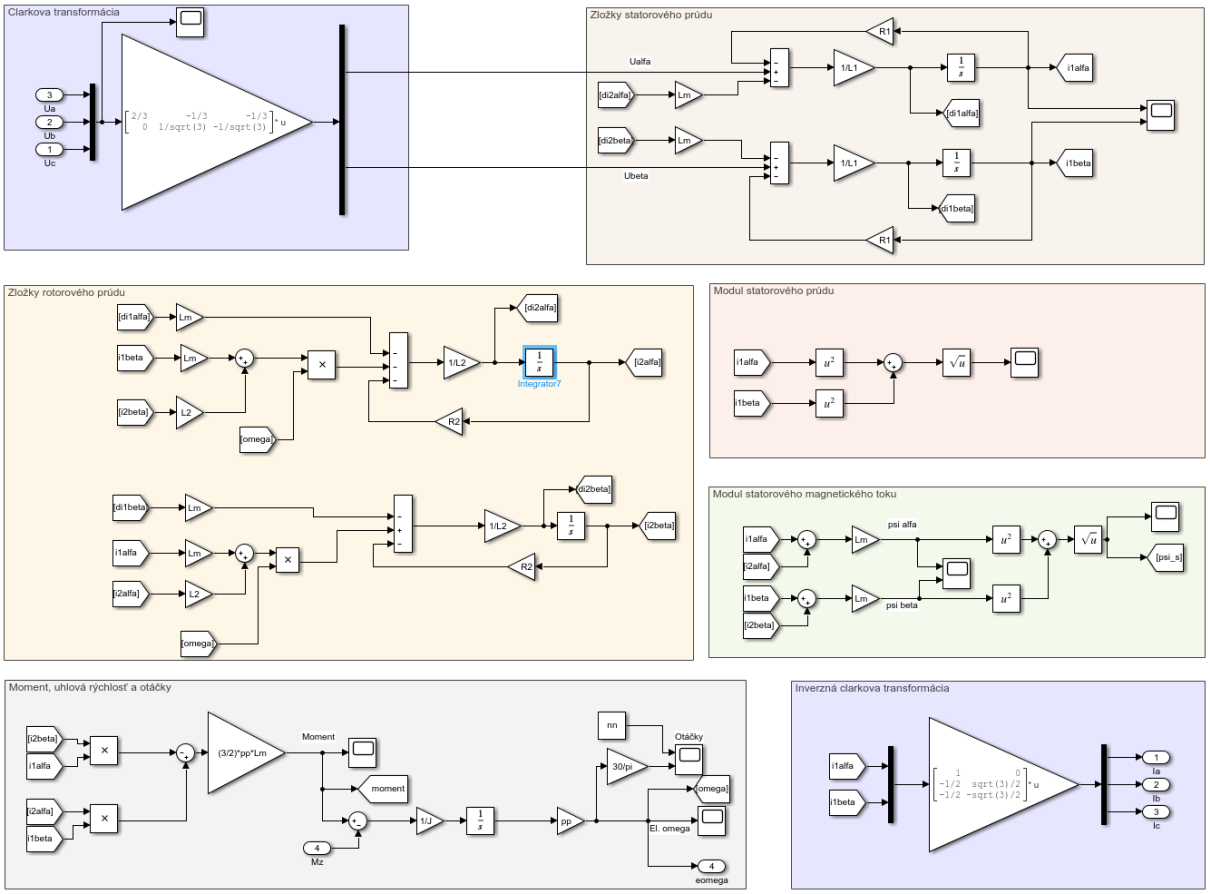
- neurónové siete (NS),
- systémy s fuzzy logikou.

Ako už bolo spomínané, v tejto práci budeme používať neurónové siete. Informácie o použitej NS budú opísané pri konkrétnej aplikácii v ďalších častiach.

KAPITOLA 4: MODEL POHONU ASYNCHRÓNNEHO MOTORA V SIMULINKU

4.1 Model motora

Rovnice pre AM z 1.1.1 Matematický model sme implementovali v programe Simulink. Na nasledujúcom obrázku môžeme vidieť, že najprv vstupné napätia u_a, u_b a u_c z meniča transformujeme Clarkovej transformáciou na u_α a u_β. Tieto napätia potom používame v modeli. Ako prvé vypočítame statorové zložky prúdu podľa rovníc *1-1* (prúdy budeme v ďalších kapitolách používať ako jedny zo vstupov do NS). Vypočítané prúdy a ich derivácie slúžia na výpočet rotorových prúdov z rovníc *1-2*. Všetky prúdy potom používame na výpočet modulu statorového toku a momentu (rovnica *1-3*), z ktorého spočítame mechanickú uhlovú rýchlosť v pohybovej rovnici *1-4*. Z tejto rýchlosti nakoniec vypočítame podľa vzorca *1-5* elektrickú uhlovú rýchlosť. Táto uhlová rýchlosť bude slúžiť ako cieľový výstupný parameter na naučenie NS. V modeli tiež počítame modul statorového prúdu, ktorý nikde nepoužívame. Ten slúži na kontrolu maximálnych hodnôt prúdu k tomu, aby sme neprekročili štítkové parametre motora a v reálnych aplikáciách nezničili motor.

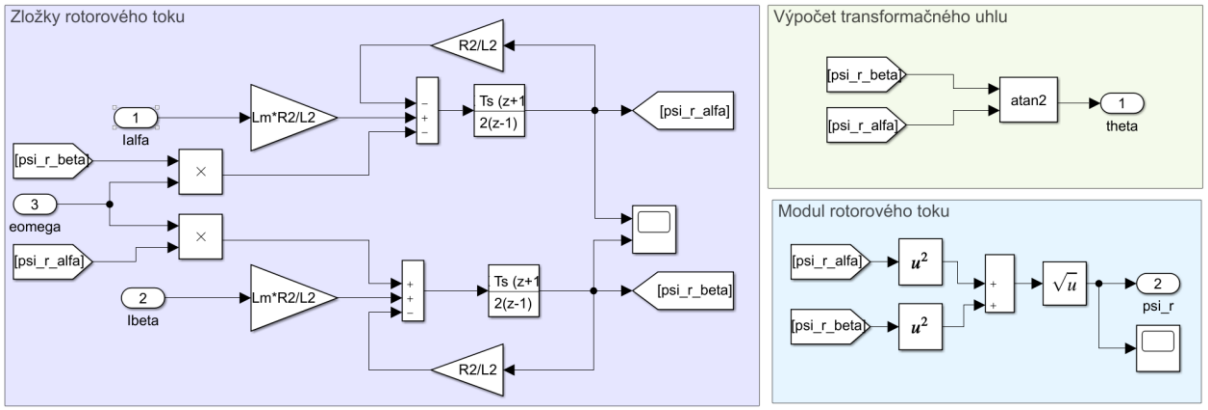


Obrázok 12 Model AM v Simulinku

4.2 In model motora

Na zistenie rotorového toku, ktorý sa používa v tokovom regulátore, existujú dva najpoužívanejšie modely motora: U-I a In model. V 0 bol ukázaný In model a ten sme implementovali v Simulinku. Vstupom do modelu sú statorové prúdy v $\alpha\beta$ súradniciach a elektrická uhlová rýchlosť. Tu podľa rovníc *1-10* vypočítame zložky rotorového magnetického toku. Môžeme si všimnúť, že sú použité diskrétne integrátory, pretože tento blok bude fungovať nespojite. Dôvod použitia týchto

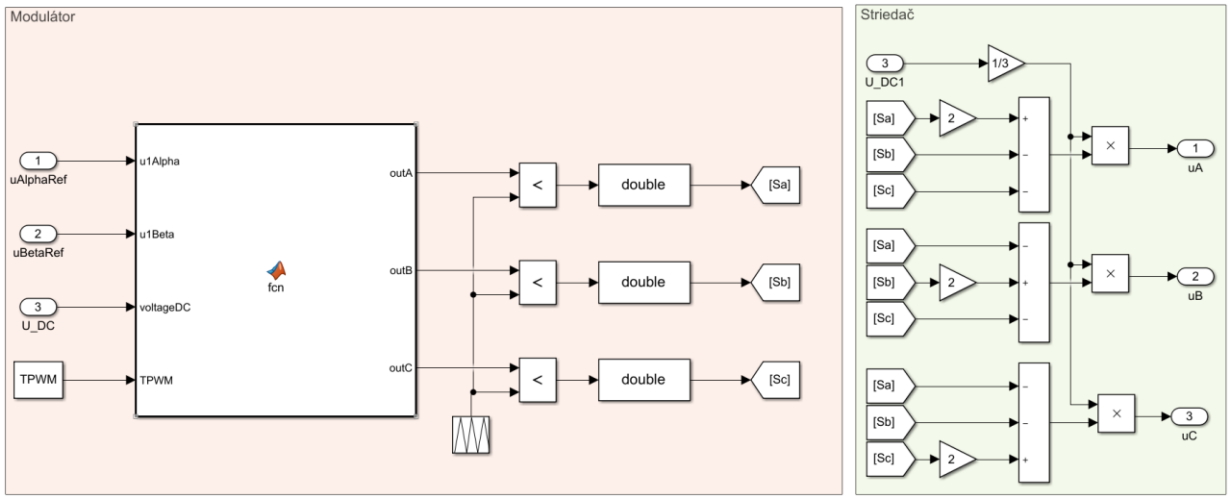
integrátorov bude vysvetlený v ďalších častiach. V tomto bloku ešte z rotorových magnetických tokov počítame modul rotorového toku podľa rovnice *1-11* a transformačný uhol θ (rovnica *1-12*).



Obrázok 13 Náhradný In model motora v Simulinku

4.3 Model meniča

Motor je napájaný trojfázovým PWM napätím z meniča, ktorého schéma je na ďalšom obrázku. Na jeho vstupe sú požadované napätia $u_\alpha{}^*$ a $u_\beta{}^*$. Menič je napájaný jednosmerným napätím U_{DC}. PWM modulácia má frekvenciu 10000 Hz. Princíp jeho fungovania je vysvetlený v (4 s. 23).



Obrázok 14 Schéma meniča v Simulinku

4.4 Model regulátora

Výstupom z regulátora je požadovaná momentotvorná a tokotvorná zložka napätia. Celkovo tento regulátor obsahuje štyri PI regulátory, z toho dva regulujú tokotvornú zložku a dva patria momentotvornej. Všetky regulátory sú nespojité a majú vnútorné anti-windup zapojenie. Anti-windup metóda bola zvolená „clamping“.

Na vstupe do regulácie tokotvornej zložky je požadovaná hodnota rotorového toku. Jej hodnotu sme vypočítali podľa vzorcov uvedených v (32).Od nej potom odčítame aktuálnu hodnotu rotorového magnetického toku vypočítanú v In modeli motora. Výsledná hodnota potom vstupuje do tokového regulátora. Horný limit saturácie je nastavený na 1,5 násobok nominálneho prúdu I_{dn}, ktorý sme vypočítali podľa vzorca :

$$I_{dn} = \frac{\psi_{2n}}{L_m} \qquad 4\text{-}1$$

Výstupom regulátora je požadovaná hodnota tokotvornej zložky prúdu. Tú odčítame od zmeranej hodnoty, ktorá bola vypočítaná transformáciou statorového prúdu. Výsledok vstupuje do regulátora tokotvornej zložky prúdu. Saturácia je nastavená na U_{fnmax} (horná hranica) a -U_{fnmax} (spodná hranica). Hodnotu U_{fnmax} sme vypočítali podľa vzťahu:

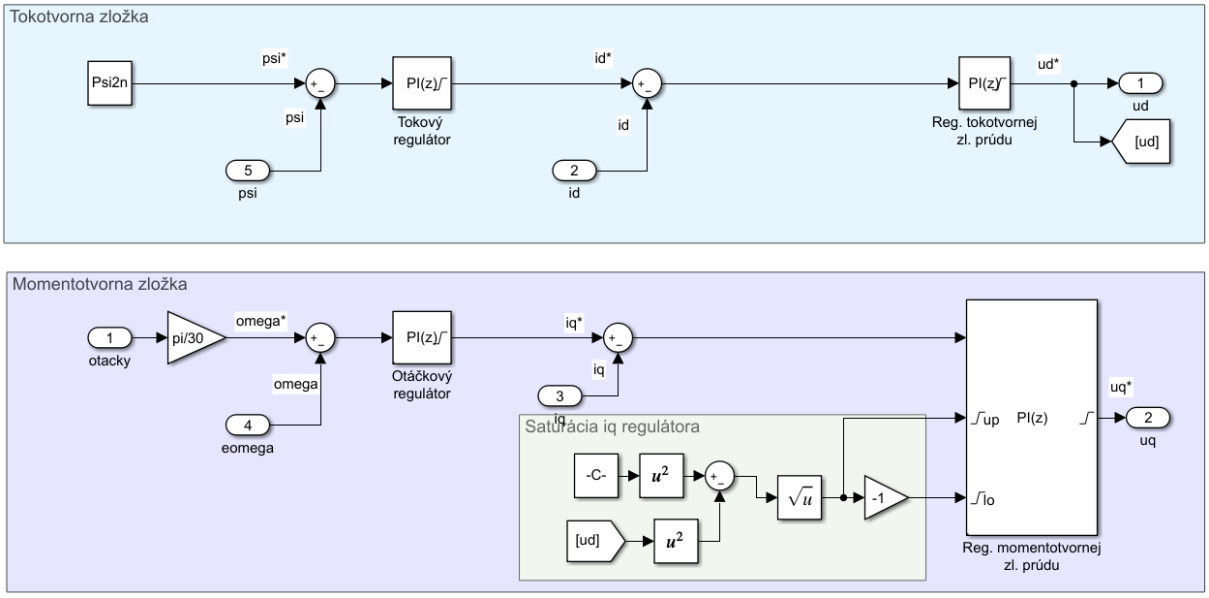
$$U_{fnmax} = U_n \frac{\sqrt{2}}{\sqrt{3}} \qquad 4\text{-}2$$

Výstupom z regulátora je potom už spomínaná požadovaná hodnota tokotvornej zložky napätia.

Na vstupe do regulácie momentotvornej zložky je požadovaná hodnota otáčok (tú prepočítavame na elektrickú uhlovú rýchlosť). Ako zdroj tam v prípade, že chceme motor rozbehnúť na určité otáčky, používame „step“ funkciu. Ďalej v projekte budeme používať aj zdroj „repeating sequence“ z dôvodu učenia NS na viac úrovní otáčok. Po odčítaní od aktuálnej uhlovej rýchlosti výsledok vstupuje do regulátora otáčok (uhlovej rýchlosti). Tu je saturácia nastavená na kladnú (horná hranica) a zápornú (spodná hranica) nominálnu hodnotu momentotvornej zložky statorového prúdu I_{sqn}. Tú vypočítame podľa vzorca:

$$I_{sqn} = \frac{(2 \cdot L_2 \cdot M_n)}{\left(3 \cdot p_p \cdot L_m \cdot \psi_{2n}\right)} \qquad 4\text{-}3$$

Výstupom je požadovaná hodnota momentotvornej zložky prúdu. Túto hodnotu odčítame od aktuálnej a výsledok vstupuje do jej regulátora. Tu je riešená saturácia dynamicky. Je navrhnutá tak, že súčet vektorov momentotvornej a tokotvornej zložky napätia nemôže byť nikdy väčší ako veľkosť vektora U_{fnmax}. Výstupom z tohto regulátora je nakoniec požadovaná hodnota momentotvornej zložky statorového napätia.

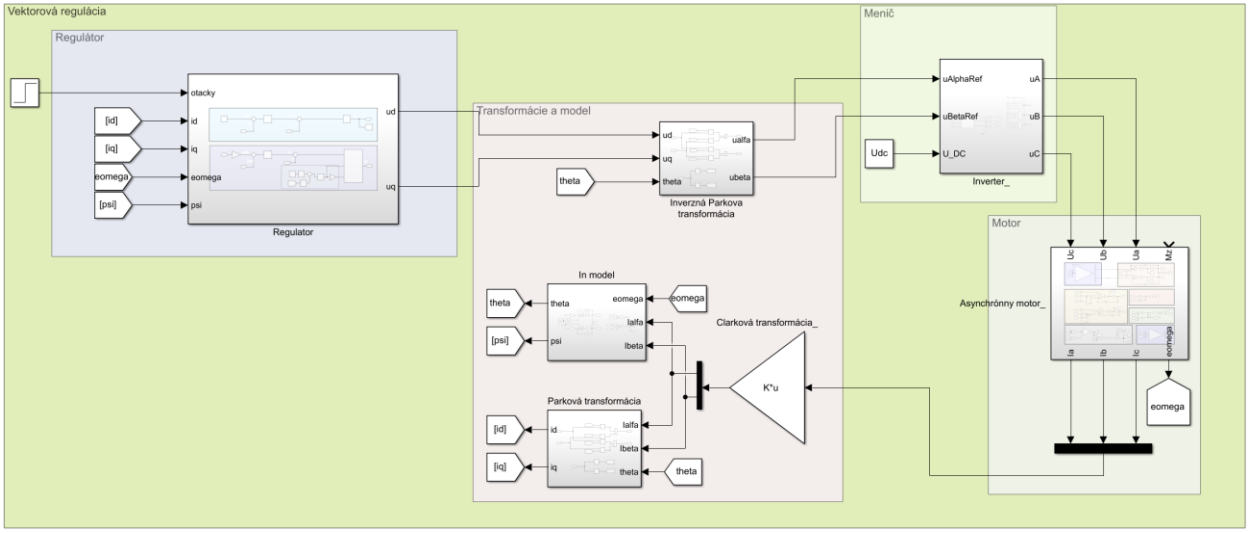


Obrázok 15 Schéma regulátora v Simulinku

4.5 Náhradný model pohonu

Bloky z predchádzajúcich častí spojíme do celkového modelu vektorovej regulácie. Schéma vektorovej regulácie potom vyzerá tak, že požadovaná hodnota statorového napätia je transformovaná pomocou inverznej Parkovej transformácie z *dq* do *αβ* súradníc a z αβ do abc súradníc pomocou inverznej Clarkovej transformácie (zakomponovaná v kóde pre modulátor). Tieto napätia potom vstupujú do bloku meniča, ktorý pracuje s frekvenciou 10000 Hz. Tieto PWM napätia potom vstupujú do modelu AM. V modeli AM meriame statorové prúdy i_a, i_b a i_c, ktoré

pomocou Clarkovej transformácie zmeníme na αβ, a tieto nakoniec transformujeme na dq prúdy pomocou Parkovej transformácie. Rotorový tok a uhol θ vypočítame s použitím In modelu AM.



Obrázok 16 Schéma pohonu v Simulinku

4.6 Celkový testovací model v Simulinku

Na zjednodušenie testovania sme v Simulinku vytvorili schému, ktorá dovoľuje testovať viac NS súčasne. Je tiež navrhnutá tak, aby zber dát na trénovanie siete bol čo najviac automatizovaný. Rozdelili sme ju do troch blokov. V prvom je upravený model vektorovej regulácie z predchádzajúcej časti. Druhý blok obsahuje prípravu na testovanie troch typov NS a posledný je navrhnutý na zber dát.

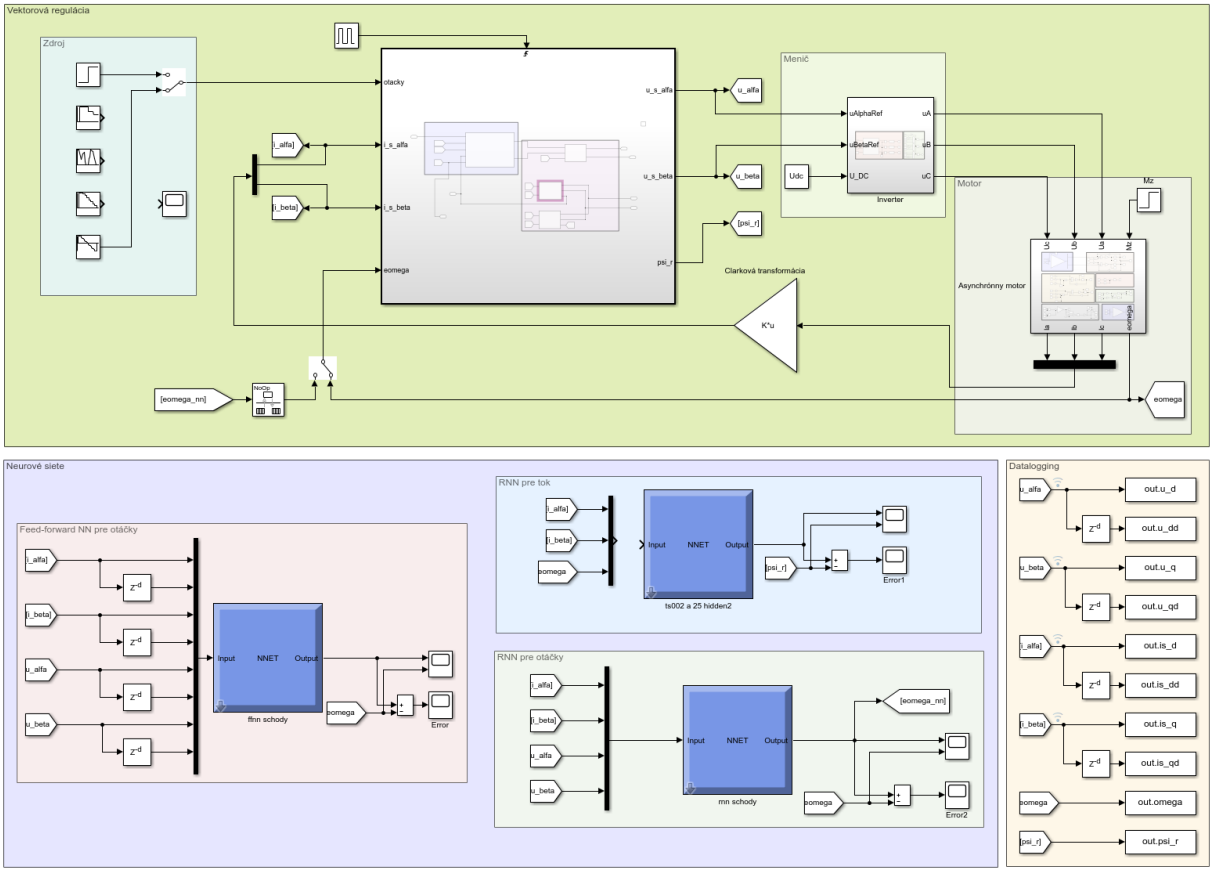
V module zdroj je definovaných 5 zdrojových funkcií na ovládanie AM. Je to z dôvodu, že čím je zložitejší priebeh, tým lepšie dokáže NS odhadnúť parametre. Na jednoduché testovanie sme používali blok „step“ a po overení funkčnosti návrhu NS sme ju potom naučili na zložitejšie schodové priebehy. V tejto práci sme chceli, aby sa simulácia čo najviac podobala reálnej aplikácii. Preto sme upravili blok regulátora (spolu ďalšími potrebnými blokmi) tak, aby fungoval nespojito. Simuluje sa tak reálny systém, kde pohon je ovládaný cez digitálny signálový procesor (DSP). Tento „triggered subsystem“ je spínaný PWM signálom s frekvenciou 10000 Hz, ktorá odpovedá skutočným frekvenciám, ktorými sú ovládané pohony. Keďže sme zmenili tento subsystém na diskrétny, museli sme v ňom zároveň zmeniť aj pár blokov (boli popísané v predchádzajúcich častiach). Zvyšné moduly potom fungujú spojito ako v skutočných podmienkach.

V bloku „Neurónové siete“ potom testujeme NS vygenerované v Matlabe. Celkovo sme v práci testovali dva typy NS v troch rôznych zapojeniach. Prvá bola dopredná sieť na odhad uhlovej rýchlosti, ktorej vstupy boli statorové prúdy a napätia a ich hodnoty z predchádzajúceho záznamu. Ďalšia sieť bola rekurentná NS na odhad uhlovej rýchlosti, kde vstupy boli len statorové prúdy a napätia. Posledný typ testovanej siete bola rekurentá NS na odhad magnetického toku rotora. Vstupmi do nej boli statorové prúdy a uhlová rýchlosť. Výstupy z týchto sietí som meral na osciloskopoch a porovnávali s reálnou uhlovou rýchlosťou. Tiež sme počítali rozdiel reálnej a odhadovanej hodnoty a zobrazovali ju na osciloskope. NS (hlavne rekurentná) je generovaná s definovanou periódou. Najlepšie výsledky generuje NS s periódou rovnakou, ako je vzorkovacia perióda vstupných signálov. Keďže je táto frekvencia odlišná od frekvencie, s ktorou pracuje regulátor, musíme použiť blok „Rate Transition“, ktorý prevedie parametre medzi frekvenciami.

Ako už bolo spomínané, v poslednom module sa zberajú dáta na naučenie NS. Tu je definovaná vzorkovacia perióda na 5 ms. S touto periódou tak vygenerujeme 200 vzoriek na sekundu z každého parametra. Metódou pokus-omyl sme tak zistili, že táto perióda dáva

optimálny pomer medzi časom na naučenie NS (čím viac vzoriek tým dlhší čas) a presnosťou výsledného priebehu.

Periódu simulácie sme zvolili 5 µs. Pri výbere tohto parametra sme sa snažili nájsť rovnováhu medzi časom simulácie, kde je motivácia ho mať čo najkratší (kvôli tomu, že sme museli otestovať desiatky NS) a presnosťou výsledkov. Časy simulácie majú rozsah od 3 sekúnd do 23, kde na otestovanie rozbehu stačí čas 3 sekundy, ale na zložitejšie priebehy so zaťaženým motorom potrebujeme dlhšie časy. Posledným simulačným parametrom, ktorý bolo potrebné nastaviť, bola voľba riešiča (solver). Po otestovaní viacerých metód sme zvolili riešič ode4 (Runge-Kutta), ktorý produkoval najoptimálnejšie výsledky.

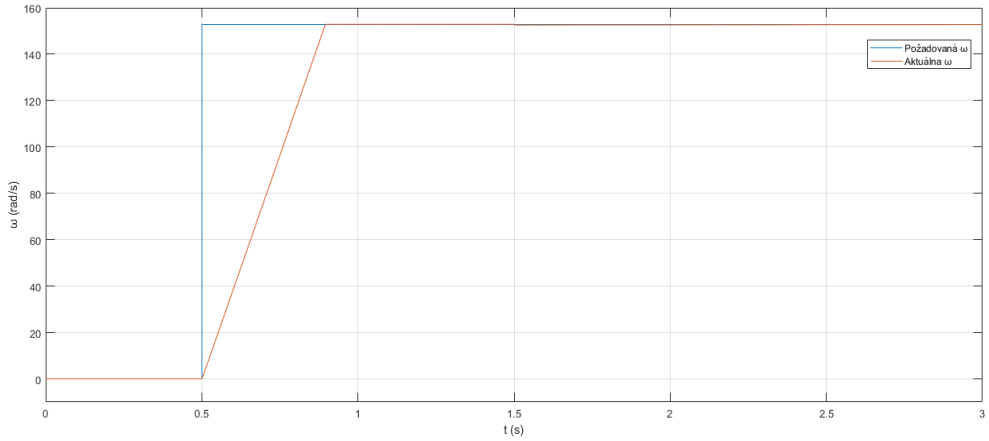


Obrázok 17 Celkový model v Simulinku

4.7 Priebehy vybraných premenných

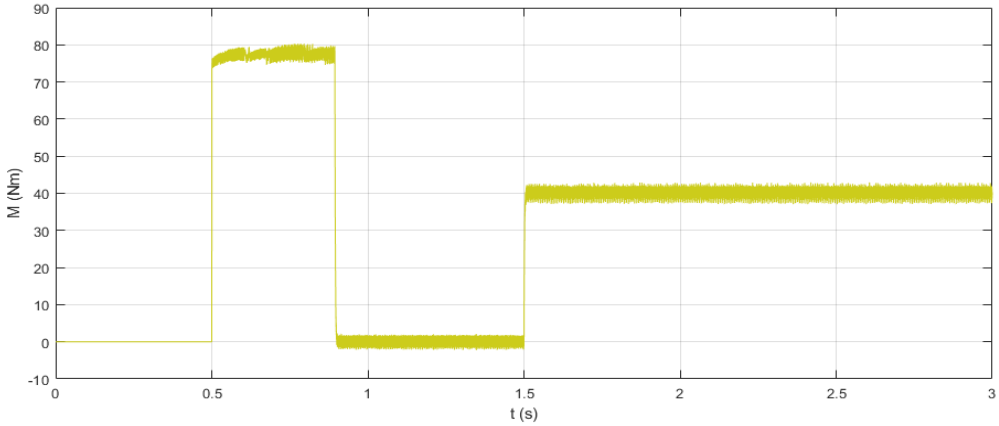
Po zostrojení tohto modelu môžeme po nastavení parametrov PI regulátorov spustiť simuláciu modelu. V tejto časti je uvedených zopár priebehov najpodstatnejších veličín. Pri realizácii úlohy bolo potrebné najprv dobre naladiť vektorovú reguláciu AM, aby dáta pre NS odpovedali reálnemu motoru. Informácie o týchto veličinách sú teda vhodné na správne prvotné naladenie regulácie.

Na prvom priebehu je zobrazená elektrická uhlová rýchlosť pri rozbehu motora na nominálne otáčky. Pomocou tohto priebehu môžeme ladiť regulátor otáčok. Je v ňom porovnanie požadovanej a reálnej rýchlosti. Vidíme, že požiadavka na rozbehnutie motora príde v čase 0,5 s. Dôvod odkladu je to, že čakáme, pokiaľ sa motor nabudí na nominálny magnetický tok. Motor sa potom rozbehne na nominálne otáčky za približne 0,4 s. Ďalej v priebehu môžeme vidieť, že v čase 1,5 s sa pripojí záťaž a otáčky sa skoro nezmenia.



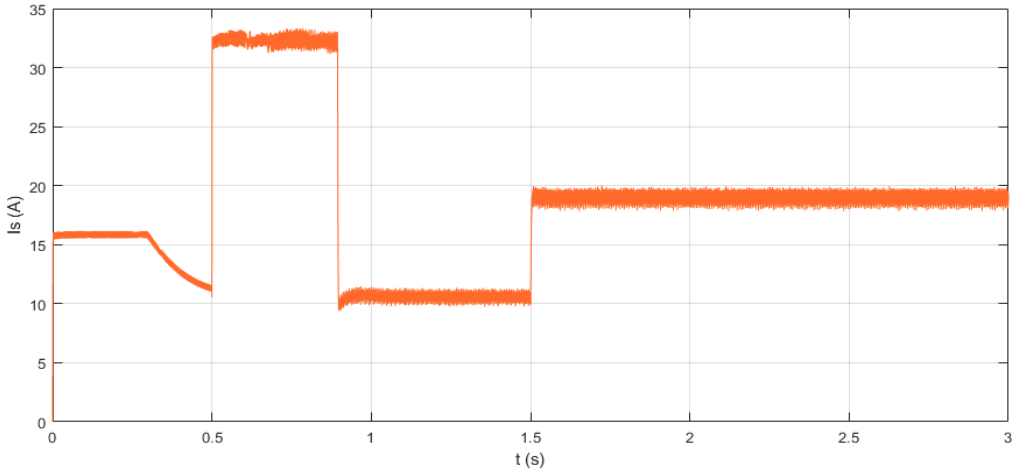
Obrázok 18 Priebeh elektrickej uhl. rýchlosti pri rozbehu AM

Na ďalšom obrázku môžeme vidieť priebeh momentu. Pri rozbehu motora jeho hodnota stúpne na 75 Nm. Po ustálení rýchlosti sa veľkosť ustáli na hodnote blízkej nule a v čase 1,5 s vidíme pripojenie záťaže 40 Nm.



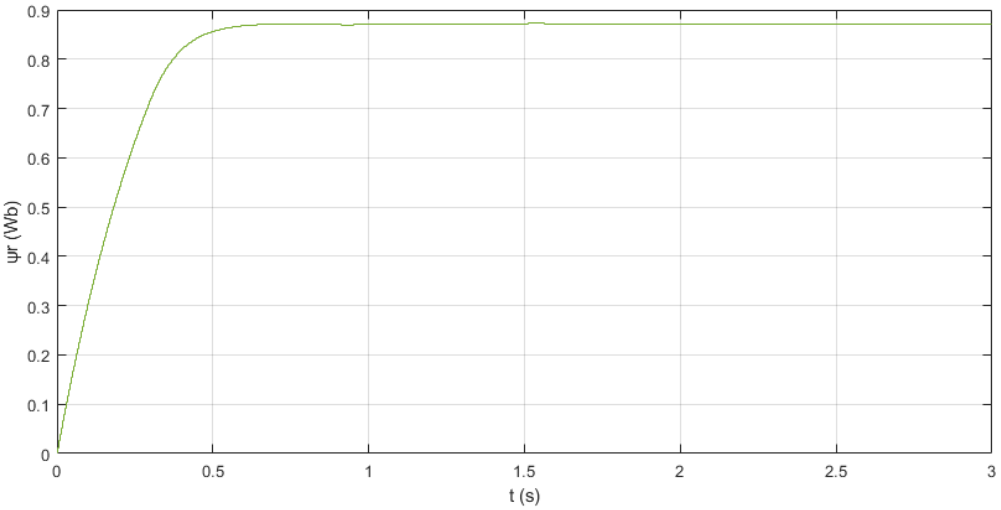
Obrázok 19 Priebeh momentu pri rozbehu AM

Nasledujúci priebeh zobrazuje veľkosť statorového prúdu. Pri rozbehu motora vidíme, že jeho hodnota je približne 32 A. Táto hodnota približne odpovedá nominálnej efektívnej hodnote statorového prúdu. Prúd sa ustáli na hodnote 10 A a po pripojení záťaže narastie na približne 19 A, čo je menej ako nominálna hodnota.



Obrázok 20 Priebeh statorového prúdu pri rozbehu AM

Na poslednom obrázku je priebeh statorového toku. Motor sa nabudí na jeho nominálnu hodnotu za približne 0,4 s. Pomocou tohto priebehu môžeme nastavovať regulátor toku.



Obrázok 21 Priebeh rotorového mag. toku pri rozbehu AM

Priebehy ďalších premenných v modeli (zložky požadovaných napätí a statorových prúdov) budú ukázané v Kapitola 6:Odhad elektrickej uhlovej rýchlosti rotora pomocou neurónových sietí. V nej sú potrebné na ilustráciu výberu vstupných dát do NS.

KAPITOLA 5: NEURÓNOVÉ SIETE V MATLABE

V tejto kapitole sa budeme venovať implementácii neurónových sietí v programe Matlab. Dôvod vzniku tejto kapitoly je ten, že na internete je málo ukážok ako v Matlabe implementovať NS, ktoré predpovedajú časové deje. Práve to bol najväčší problém pri práci na tejto úlohe. Mala by tak poskytnúť návod, ako pracovať s knižnicou NN Toolbox. Knižnica NN Toolbox prešla za posledné roky mnohými vylepšeniami a bolo tak aktualizovaných mnoho funkcií. Niektoré základné príkazy, ktoré sú uvedené v odborných publikáciách hlavne pred rokom 2010, už nie sú aktuálne a nenachádzajú sa tiež v oficiálnom manuáli od Mathworks. Aj keď sú staré príkazy ešte funkčné, z mojich skúseností sa osvedčilo používať najnovšie príkazy, pretože sú lepšie zadokumentované a majú viac vstavaných funkcií. Podrobný popis hlavných príkazov je v Príloha B:.

5.1 Spracovanie vstupných dát

Jedna zo základných vecí na zostrojenie funkčnej NS, ktorá vie správne reagovať na predpokladané vstupy, je dobrý výber meraných parametrov. Tieto parametre musia dobre odzrkadľovať správanie systému v každom očakávanom stave.

V prípade, že sme vybrali tie správne veličiny a dostatočný počet vzoriek, môžeme ich importovať do programu. Pri tomto kroku si musíme dať pozor, aby sme mali rovnaký počet očakávaných vstupov a výstupov a aby tieto údaje boli na rovnakých miestach v maticiach. Časové priebehy dát nahraných zo Simulinku majú opačne stĺpce a riadky, ako ich spracováva knižnica. Musíme teda tieto dáta transponovať pri načítaní. Ďalší krok je zmena formátu vektorov zo súbežných na sekvenčné cez príkaz *con2seq*. S takto upravenými dátami môžeme ďalej pracovať v programe.

5.2 Dopredné neurónové siete v Matlabe

Nový príkaz pre generovanie základnej doprednej NS je *feedforwardnet* (starý je *newff*). Pomocou neho sa inicializuje nová sieť. V príkaze určujeme dva parametre. Prvým je množstvo skrytých vrstiev a počet neurónov v nich. Táto štruktúra sa definuje v hranatých zátvorkách, kde čísla oddelené čiarkou odpovedajú počtu neurónov v konkrétnej vrstve. Druhým parametrom je tréningový algoritmus. Ako základný je tu nastavený Levenberg-Marquardtov algoritmus a tento bude dostačujúci pre naše aplikácie (33).

Po výbere parametrov učenia (bude opísané v ďalšej časti) spustíme trénovanie NS pomocou príkazu *train*, kde vstupnými argumentmi sú nakonfigurovaná NS, vstupné požadované dáta a výstupné požadované dáta (34).

V Matlabe tiež môžeme generovať nové typy dopredných sietí, ako napríklad kaskádovú NS, a to pomocou príkazu *cascadeforwardnet* (33).

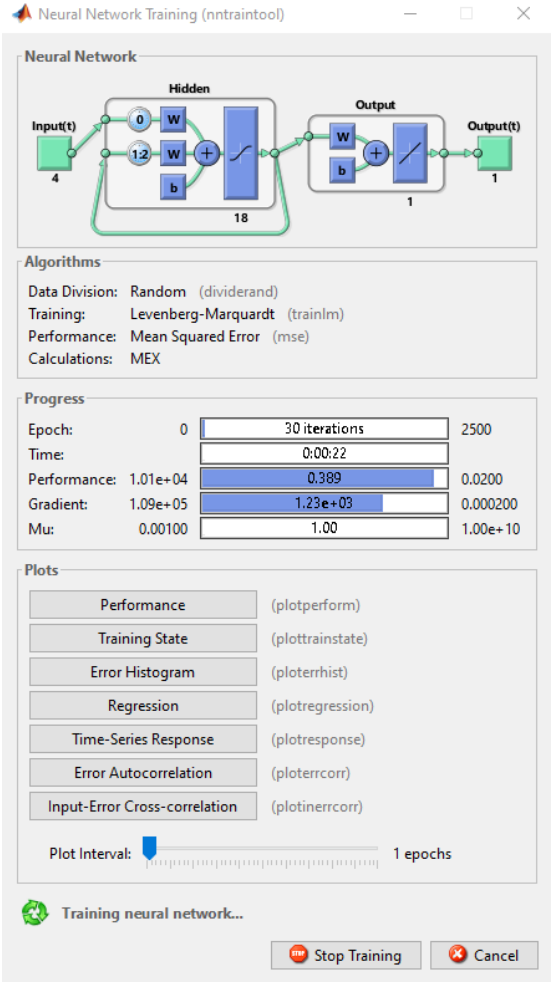
5.3 Rekurentné neurónové siete v Matlabe

Rekurentné siete sa definujú pomocou príkazu *layrecnet* (starý ekvivalent neexistuje, používala sa Elmanova sieť, ktorá je predchodcom rekurentnej NS). Vygeneruje sa tak objekt siete, v ktorom určujeme tri parametre. Keďže rekurentné siete majú spätnú väzbu, v prvom argumente vyberáme, o koľko sa výstup zo spätnej väzby oneskorí. Druhým parametrom je počet skrytých vrstiev, a ten sa určuje rovnako ako v prípade doprednej NS. Posledným argumentom je výber tréningového algoritmu a aj tu je základný Levenberg-Marquardtov algoritmus (35).

Po vygenerovaní objektu siete musíme na rozdiel od predchádzajúcej NS upraviť dáta tak, aby bol formát vhodný na učenie. To má na starosti príkaz *preparets*. Ten zo vstupných dát vygeneruje nové premenné, ako napríklad posunuté vstupy alebo počiatočné stavy odkladov (podrobný popis je v časti B.5) (36).

Objekt siete spolu s preformátovanými hodnotami sú potom vstupnými argumentmi pre učenie. To tiež spúšťame pomocou príkazu *train*.

Na nasledujúcom obrázku je príklad prostredia, ktoré sa otvorí po spustení kódu. V časti „Neural Network" je obrázok navrhnutej siete. Môžeme sa tak uistiť, že sme NS nastavili správne. Ďalej v časti „Algorithms" sú uvedené vybrané výpočtové algoritmy, či už na trénovanie alebo výpočet chyby. V časti „Progress" sú zobrazené aktuálne hodnoty hlavných parametrov učenia. Podľa nich môžeme sledovať, ako je na tom aktuálny stav tréningu a rozhodnúť o ukončení alebo pokračovaní procesu. V časti „Plots" sú tlačidla na generovanie grafov o priebehu učenia. Niekoľko príkladov je uvedených v ďalšej kapitole. Tréning môžeme zastaviť manuálne pomocou tlačidla „Stop Training", aj v tomto prípade sa vygeneruje sieť.



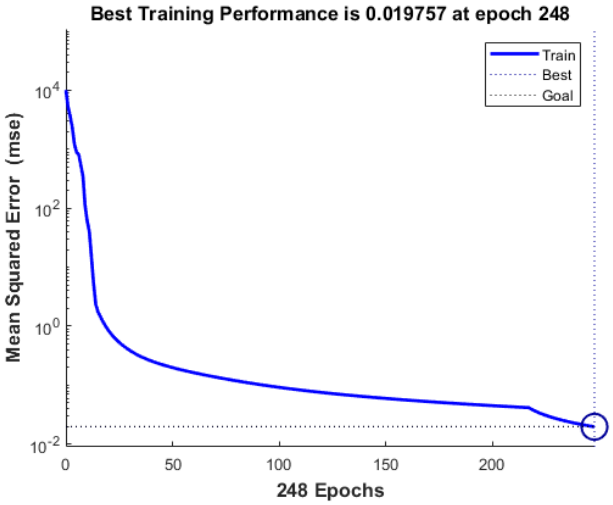
Obrázok 22 Prostredie NN Toolbox

5.3.1 Optimalizácia učenia

Správne nastavenie parametrov učenia nám umožňuje ušetriť množstvo času a výpočtového výkonu. K trénovaniu NS môžeme pristupovať z dvoch uhlov. Prvý je predimenzovanie NS a použitie veľkého výpočtového výkonu na jej naučenie. Druhý je čo najlepšia optimalizácia učiacich parametrov a tak skrátenie celkového času. V našom prípade, keďže sme bol obmedzení možnosťou simulácie len na osobnom počítači, sme si vybrali cestu čo najlepšie optimalizovaných parametrov.

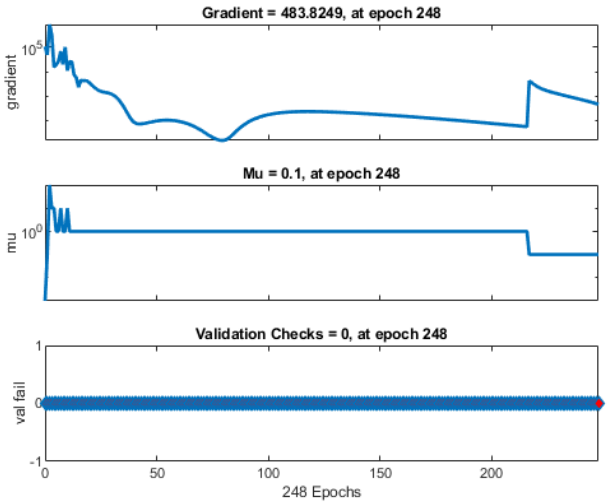
Ako už bolo spomínané v predchádzajúcich kapitolách, tento proces je založený na intuícii. Výpočet sa zastaví, pokiaľ dosiahne jeden z nasledujúcich parametrov, alebo kontrolný algoritmus zhodnotí, že navrhnutá NS nie je dostatočná na dosiahnutie definovanej chyby na vstupných dátach. Učenie sa tiež môže zastaviť dosiahnutím maximálneho počtu epoch a maximálneho času. Príkazy na nastavenie konkrétnych parametrov sú uvedené v prílohe B.4.

Hlavným parametrom na nastavenie je učiaci výkon, teda stredná kvadratická chyba. Jej hodnotu vyberáme podľa toho, ako presne chceme odhadovať priebehy. Tiež musíme brať do úvahy štruktúru dát. Takže v prípade, že máme zložité priebehy s množstvom dát, vyberáme väčšiu chybu ako v prípade použitia jednoduchších vstupov. Takéto priebehy je buď nemožné odhadnúť presne, alebo by sme na to potrebovali veľkú NS, ktorá by potrebovala veľa času a výpočtového výkonu na tréning. Snažíme sa teda nájsť rovnováhu medzi uspokojivým výsledkom a časom učenia. V nasledujúcom priebehu môžeme vidieť NS, ktorá dosiahla požadovanú strednú kvadratickú chybu.



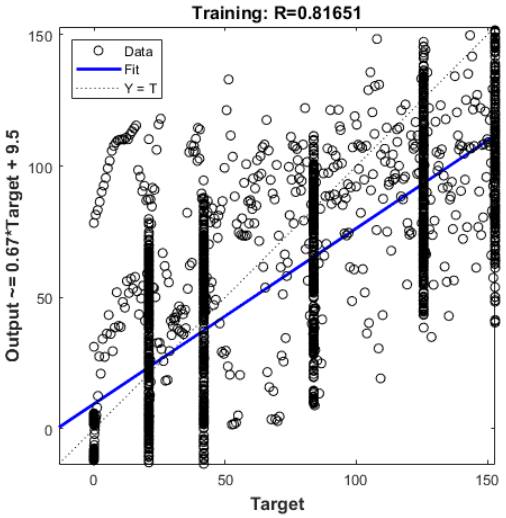
Obrázok 23 Priebeh strednej kvadratickej chyby pri úspešnom učení

Ďalším z hlavných parametrov tréningu, ktoré určujeme, je minimálny gradient. Ten zjednodušene udáva, po akých krokoch sa má algoritmus blížiť k výsledku. Takže čím bude menší krok, teda menší gradient, tým pomalšie sa dostaneme k výsledku, no tento výsledok bude presnejší.

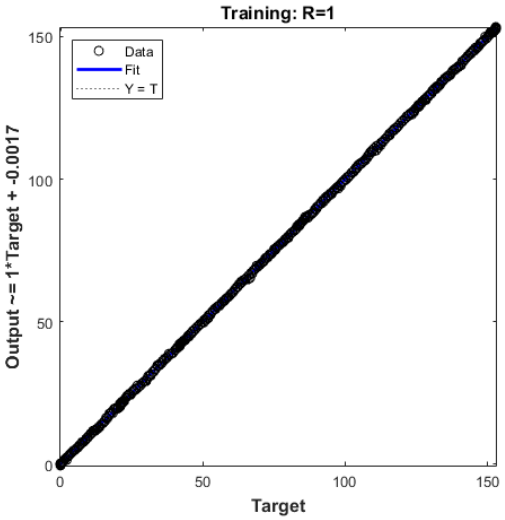


Obrázok 24 Prehľad ďalších premenných pri učení

Na posledných obrázkoch sú grafy regresie. Na týchto grafoch kontrolujeme, či je NS naučená správne. Podoba tohto grafu závisí od aktivačnej funkcie výstupnej vrstvy. Pre prípad na obrázkoch bola vybraná lineárna AF. Prvý obrázok ukazuje stav regresie v prvých epochách učenia. Vidíme, že ešte nie sú dobre nastavené váhy prepojení, a tak sa výstupy vygenerované NS neprojektujú na predpokladané výstupy. Na obrázku vpravo je regresný graf správne naučenej NS a môžeme vidieť, že výstupy z NS odpovedajú predpokladaným, a tak má graf lineárny charakter.



Obrázok 25 Regresia počas prvých epoch



Obrázok 26 Regresia po úspešnom konci učenia

5.4 Generovanie naučenej neurónovej siete

Po dokončení učenia sa v projekte vygeneruje objekt s NS. S týmto objektom môžeme pracovať podobne ako s inými premennými v Matlabe. Môžeme ho teda uložiť do mat súboru alebo použiť v ďalších programoch. Na použitie v Simulinku musíme z tohto objektu vygenerovať blok. Ten sa generuje pomocou príkazu *gensim*. Vstupnými parametrami sú NS a perióda, s ktorou boli vzorkované vstupné dáta (základná hodnota je -1).

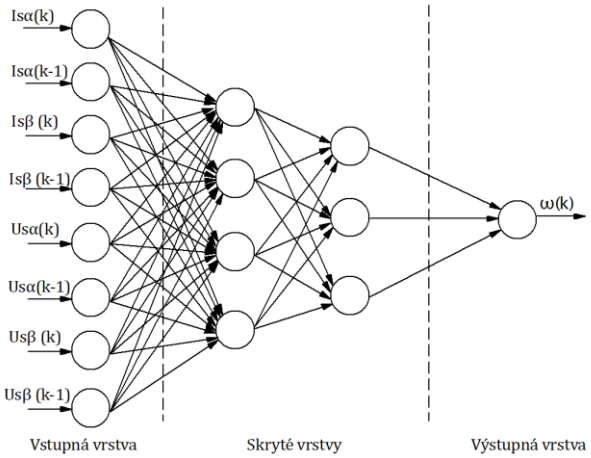
KAPITOLA 6: ODHAD ELEKTRICKEJ UHLOVEJ RÝCHLOSTI ROTORA POMOCOU NEURÓNOVÝCH SIETÍ

Hlavnou časťou tejto diplomovej práce je implementácia neurónových sietí do vektorovej regulácie. Ako už bolo spomínané v predchádzajúcich kapitolách, pohon bude riadený s využitím odhadnutej elektrickej uhlovej rýchlosti motora a magnetického toku rotora. Táto kapitola sa venuje odhadu veličín pomocou NS. V prvej časti budeme túto rýchlosť na nezaťaženom AM odhadovať pomocou doprednej NS, v druhej využijeme rekurentnú NS a v tretej časti opíšeme výsledky simulácie so záťažou.

6.1 Dopredná NS

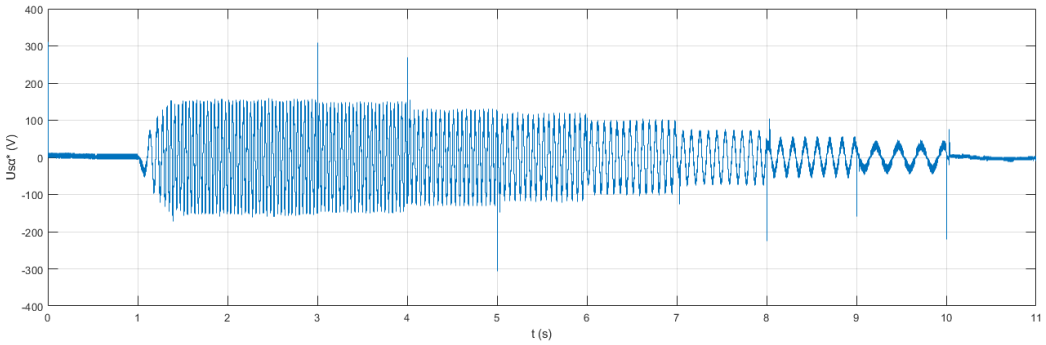
6.1.1 Výber vstupov na učenie

Je veľa rôznych kombinácií meraných parametrov, ktoré sa dajú použiť pri odhadovaní uhlovej rýchlosti, ako napríklad odhad pomocou statorových napätí (*Usd* a *Usq*), rotorového toku a momentu (37), no hoci je tento prípad spoľahlivý a ľahko realizovateľný v simulátore, obsahuje veličiny, ktoré sa ťažko merajú. Ako jedna z najpraktickejších možností sa javí použitie statorových napätí a prúdov (*Usα*, Usβ*, Isα, Isβ*) (38). Statorové napätia budeme v modeli merať na výstupoch z regulátora (v praxi ich merať nemusíme, pretože sú súčasťou regulácie). Statorové prúdy meriame na výstupoch z meniča (v praxi sa merajú buď halovými sondami alebo poklesom napätia na odporoch). Pre správne naučenie NS budeme potrebovať aj hodnotu prúdov a napätí jednu vzorku predtým, takže NS má 8 vstupov. Ako výstup neurónovej siete bude elektrická uhlová rýchlosť rotora (v praxi zmeriame senzorom otáčok a prevedieme). Zjednodušená schéma takejto siete je na nasledujúcom obrázku.

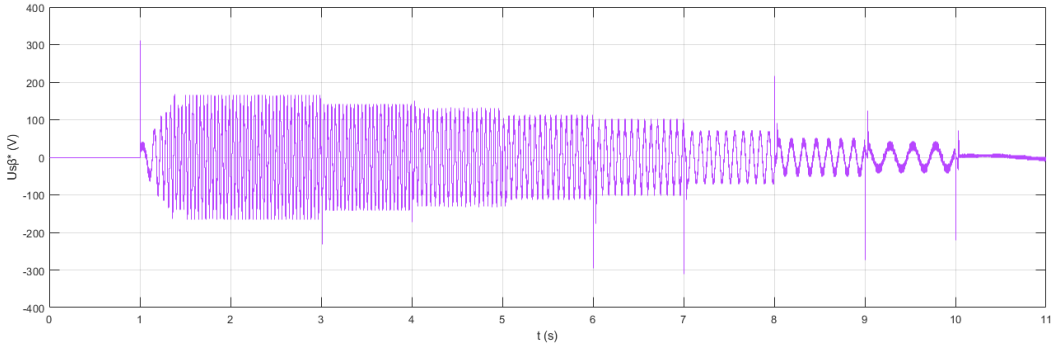


Obrázok 27 Zjednodušená schéma doprednej NS

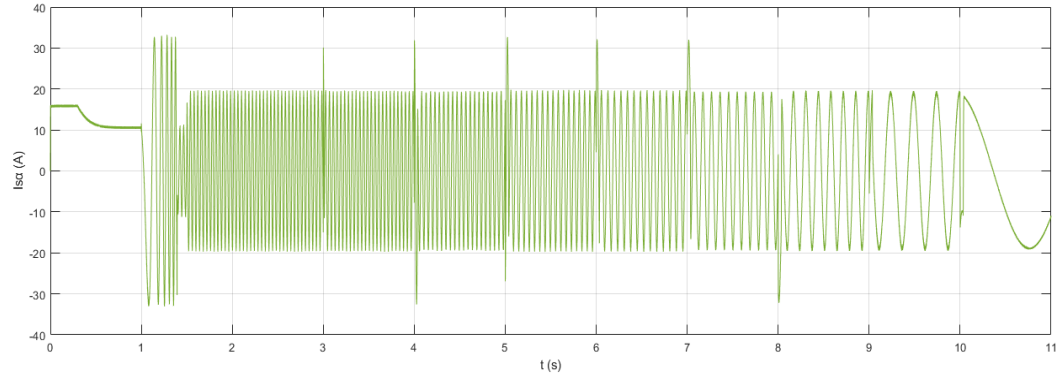
Na zaznamenanie dát sme použili zdroj, ktorý rozbehne motor na nominálnu elektrickú uhlovú rýchlosť a potom reguluje rýchlosť tak, aby sa hodnota ustálila na deviatich úrovniach. Priebeh regulovanej elektrickej uhlovej rýchlosti je na Obrázok 32 Priebeh odhadovanej a skutočnej uhlovej rýchlosti. Vzorkovaciu periódu sme zvolili 5 ms, takže budeme mať 2200 vzoriek z každého parametra (spolu 19800 vzoriek). Všetky tieto parametre budú slúžiť ako vstup na naučenie NS. Zmerané prúdy a napätia sú na nasledujúcich priebehoch. Tieto obrázky majú poskytnúť čitateľovi obraz o tom, z akých dát je NS schopná celkom presne odhadnúť uhlovú rýchlosť.



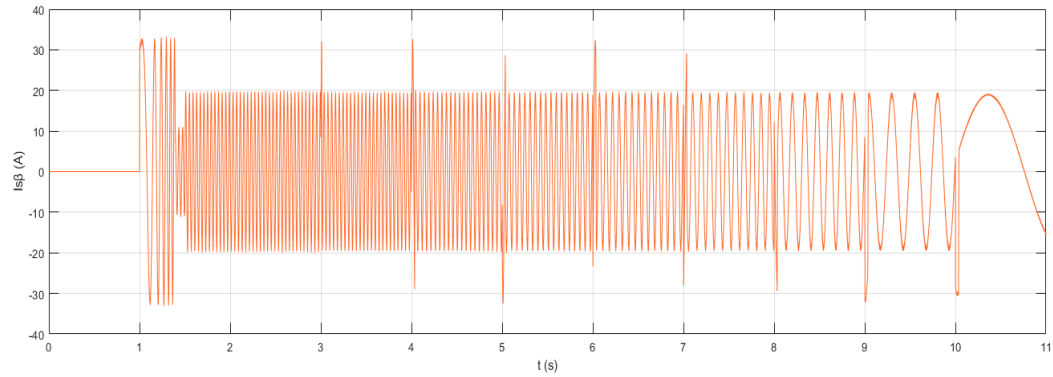
Obrázok 28 Priebeh $U_{sα}^{*}$



Obrázok 29 Priebeh $U_{sβ}^{*}$



Obrázok 30 Priebeh $I_{sα}$



Obrázok 31 Priebeh $I_{sβ}$

6.1.2 Generovanie neurónovej siete

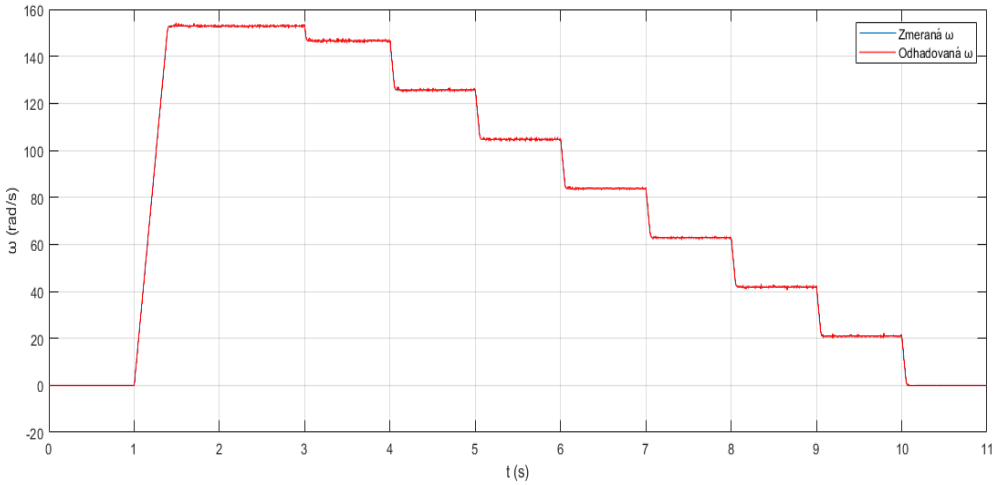
Použitá dopredná NS má 8 vstupných neurónov, 18 neurónov v prvej skrytej vrstve, 25 neurónov v druhej skrytej vrstve, 45 v tretej, 36 v štvrtej, 28 v piatej, 11 v šiestej a 1 výstupný neurón (pre takúto NS sa používa značenie 8-18-25-45-36-28-11-1). Zvyšné parametre sme nechali podľa základného nastavenia NN Toolboxu. Takže aktivačné funkcie pre vstupnú a skryté vrstvy sú hyperbolický tangens (tanh) a výstupná vrstva používa lineárnu aktivačnú funkciu. Ako základná tréningová funkcia sa používa Levenberg-Marquardtov algoritmus.

V ďalšej časti sme vyberali parametre na trénovanie. Maximálny počet epoch sme zvolili 3000 (trénovanie sa zastaví ešte pred dosiahnutím tohto limitu na 861 epochách). Parameter "výpočtový cieľ" sme nastavil na 0,05. A maximálny gradient sme zvolili o dva rady väčší ako výpočtový cieľ (0,0005). Nakoniec sa spustí učenie cez príkaz *train*. Kód na vygenerovanie takejto siete je nasledujúci:

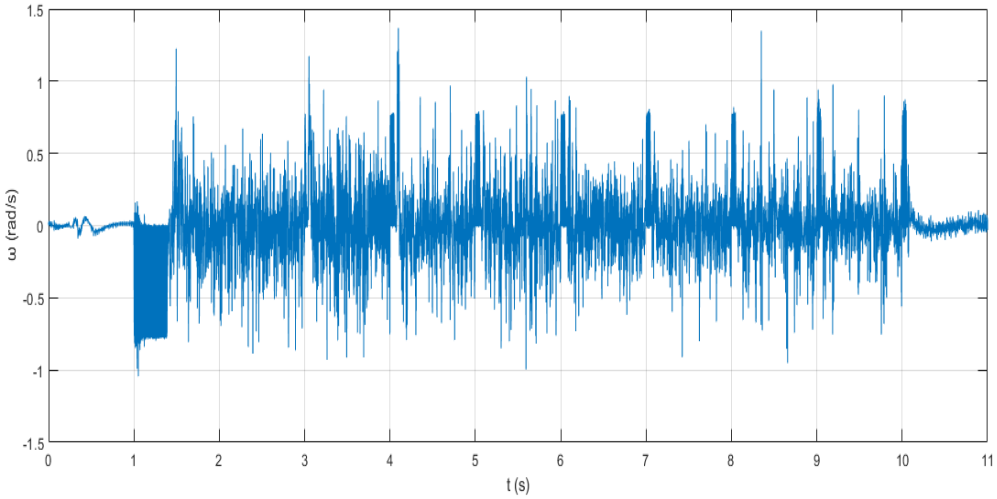
```
%načítanie vstupov a zmena na sekvenčné

I=[out.is_alfa()';out.is_alfa_p()';out.is_beta()';out.is_beta_p()';out.u_al
fa()';out.u_alfa_p()';out.u_beta()';out.u_beta_p()'];
T=out.omega()';
I=con2seq(I);
T=con2seq(T);

%definovanie neurónovej siete
net=feedforwardnet([18,25,45,36,28,11]);
%parametre a spustenie učenia
net.trainParam.epochs=3000;
net.trainParam.goal=5e-2;
net.trainParam.min_grad=5e-4;
net=train(net,I,T);
```

6.1.3 Testovanie neurónovej siete

Po importovaní NS do Simulinku sme ako prvé otestovali, ako sa správa, ak vstupy sú dáta, ktoré boli použité na učenie. Podľa predpokladov je priebeh skoro zhodný s reálnym priebehom uhlovej rýchlosti a ako môžeme vidieť v grafe rozdielu medzi skutočnou a odhadovanou uhlovou rýchlosťou.

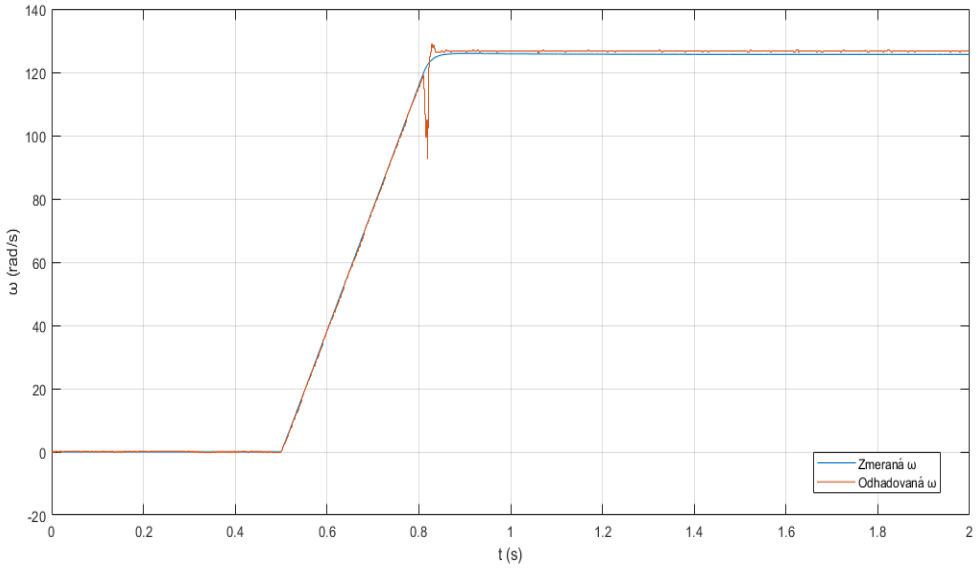


Obrázok 32 Priebeh odhadovanej a skutočnej uhlovej rýchlosti

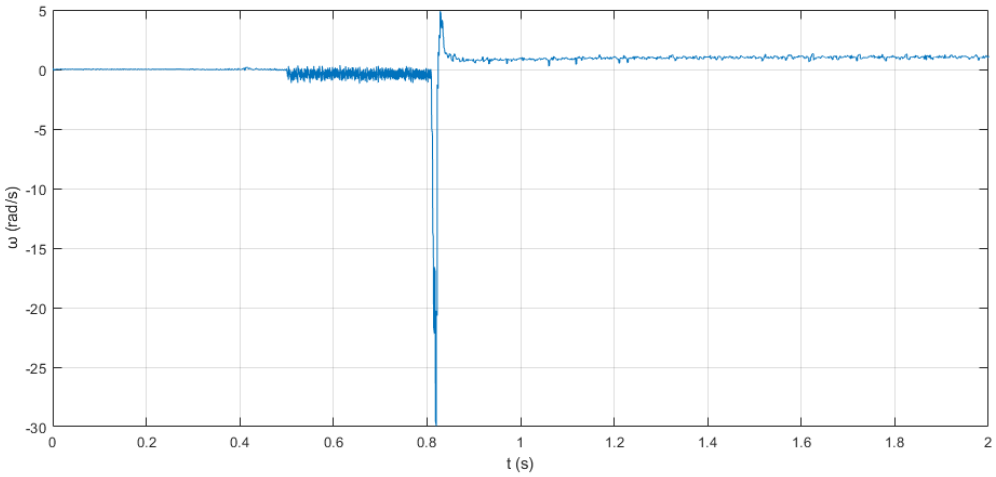


Obrázok 33 Rozdiel medzi odhadovanou a skutočnou rýchlosťou

Aby sme overili správne naučenie NS, testovali sme ju na rozbehy motora na rôzne uhlové rýchlosti, na ktoré nebola trénovaná. Na nasledujúcich grafoch je vidieť príklad rozbehu motora na nenaučenú uhlovú rýchlosť. Veľkosť tejto rýchlosti nie je podstatná, pretože v tejto práci skúmame podobnosť rýchlostí a nie ich presnú hodnotu. Viac informácií o splnení úlohy podáva graf rozdielu a v ňom je chyba oproti prvému priebehu väčšia, no po ustálení prechodného deja môžeme povedať, že odhadnutá rýchlosť je podobná skutočnej. Táto chyba by sa dala ešte zmenšiť optimalizáciou tréningového procesu alebo rozšírením neurónovej siete.



Obrázok 34 Priebeh odhadovanej a skutočnej rýchlosti pri nenaučenom rozbehu

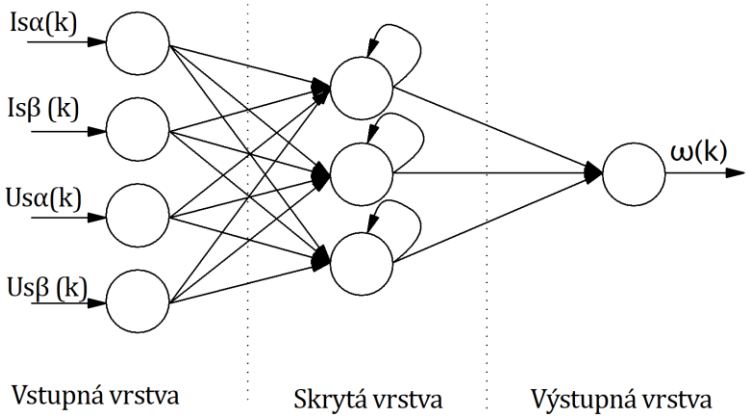


Obrázok 35 Rozdiel medzi odhadovanou a skutočnou rýchlosťou pri nenaučenom rozbehu

6.2 Rekurentná neurónová sieť

6.2.1 Výber vstupov na učenie

Ďalší typ siete opísaný v Kapitola 2: je rekurentná NS. Aj v tomto prípade budeme na trénovanie používať merané statorové prúdy a požadované statorové napätia v *αβ* súradniciach ako vstupy. Očakávaný výstup bude elektrická uhlová rýchlosť. Výhodou tejto siete je, že nebudeme potrebovať hodnoty týchto parametrov z predchádzajúcej vzorky. NS má spätnú väzbu (v našom prípade je spätná väzba elektrická uhlová rýchlosť), a tak veľmi dobre odhaduje spojité časové deje. Zjednodušená schéma je na nasledujúcom obrázku. Priebehy vstupných premenných sú zhodné s predchádzajúcou časťou a nebudem ich tu uvádzať.



Obrázok 36 Zjednodušená schéma rekurentnej NS

6.2.2 Generovanie neurónovej siete

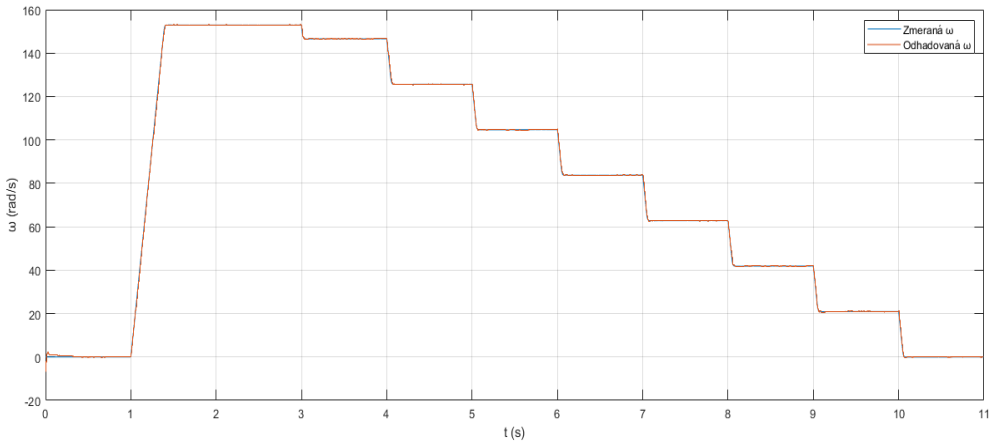
Vďaka spätnej väzbe nemusí mať rekurentná NS toľko vrstiev ako dopredná NS. No aj pri menšom počte neurónov trvá trénovanie takejto siete rádovo dlhšie ako doprednej. Je to dané jej zložitejšou štruktúrou. Ako príklad môžeme uviesť sieť z predchádzajúcej časti, kde jej naučenie trvalo približne pol hodiny, zatiaľ čo trénovanie rekurentnej NS s jednou skrytou vrstvou použitej v tomto príklade trvalo asi 3 hodiny.

Navrhnutá sieť ma len jeden skrytú vrstvu, a tá má 28 neurónov. Ostatné parametre učenia sú zhodné s doprednou NS. Potrebný kód na zostrojenie tejto siete je pod odsekom. V ňom vidíme, že v tomto prípade sme načítali len štyri vstupné premenné a jednu výstupnú, ktoré sme potom preformátovali na sekvenčné typy. Vygenerovali sme rekurentnú NS a pripravili si dáta na učenie cez príkaz *preparets*. Nakoniec sme spustili tréning. Ten sa zastavil po 308 epochách.

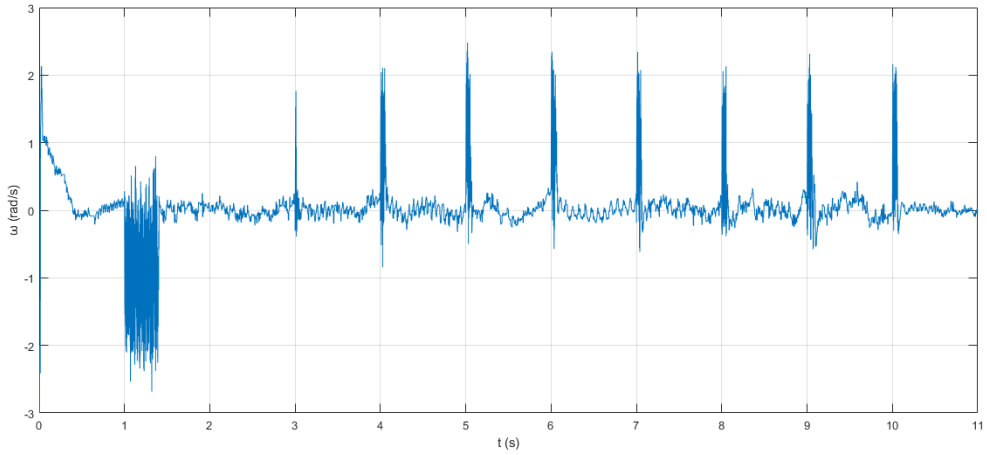
```
%načítanie vstupov a zmena na sekvenčné
I=[out.is_alfa()';out.is_beta()';out.u_alfa()';out.u_beta()'];
T=out.omega()';
I=con2seq(I);
T=con2seq(T);

%definovanie neurónovej siete
net = layrecnet(1:2,[28]);
[Xs,Xi,Ai,Ts] = preparets(net,I,T);
%parametre a spustenie učenia
net.trainParam.epochs=2500;
net.trainParam.goal=5e-2;
net.trainParam.min_grad=5e-4;
net=train(net,Xs,Ts,Xi,Ai);
```

6.2.3 Testovanie neurónovej siete

Po naučení sme vygenerovanú sieť znova importovali do Simulinku a ako v predchádzajúcej úlohe sme otestovali jej výstupy na vstupné dáta, ktoré boli použité na učenie. Aj tu sme potvrdili, že táto rekurentná sieť je naučená správne a ako môžeme vidieť na nasledujúcich obrázkoch, kopíruje priebeh skutočnej hodnoty. Z grafu rozdielu odhadovanej a skutočnej uhlovej rýchlosti tiež vidíme, že chyba je minimálna pri ustálenej uhlovej rýchlosti a zväčší sa pri zmene.

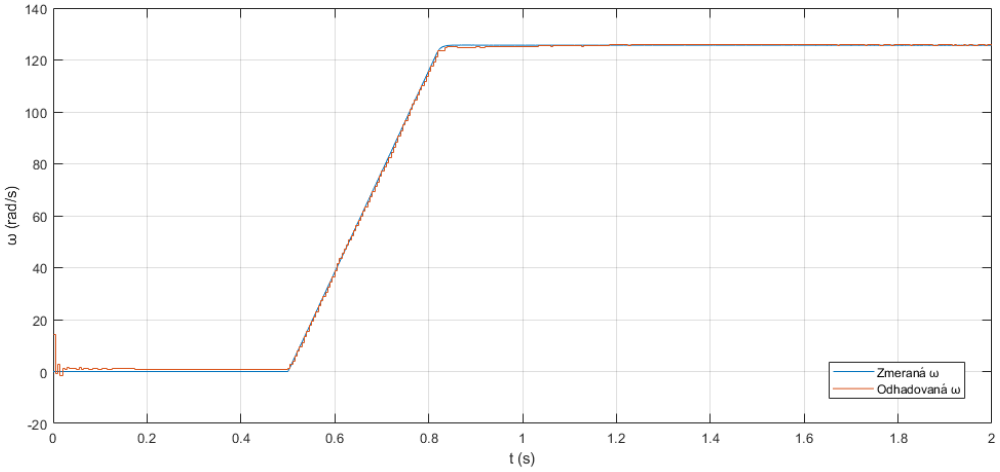


Obrázok 37 Priebeh odhadovanej a skutočnej uhlovej rýchlosti

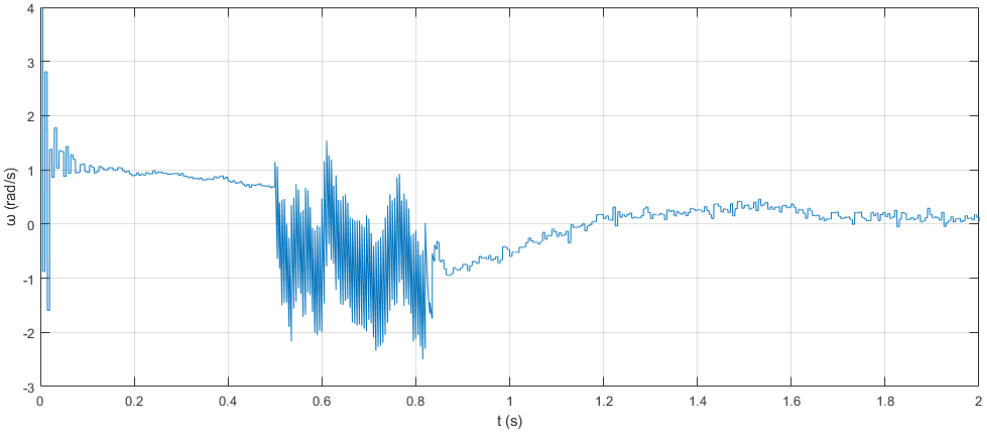


Obrázok 38 Rozdiel medzi odhadovanou a skutočnou rýchlosťou

Na otestovanie funkčnosti siete sme použili rovnaký zdroj ako v predchádzajúcej podkapitole, aby sme mohli porovnať výsledné priebehy, a hlavne priebehy rozdielov, z ktorých učíme vhodnejšiu sieť na túto aplikáciu. Z obrázkov vidíme, že odozva NS na nenaučenú uhlovú rýchlosť dáva minimálnu chybu.

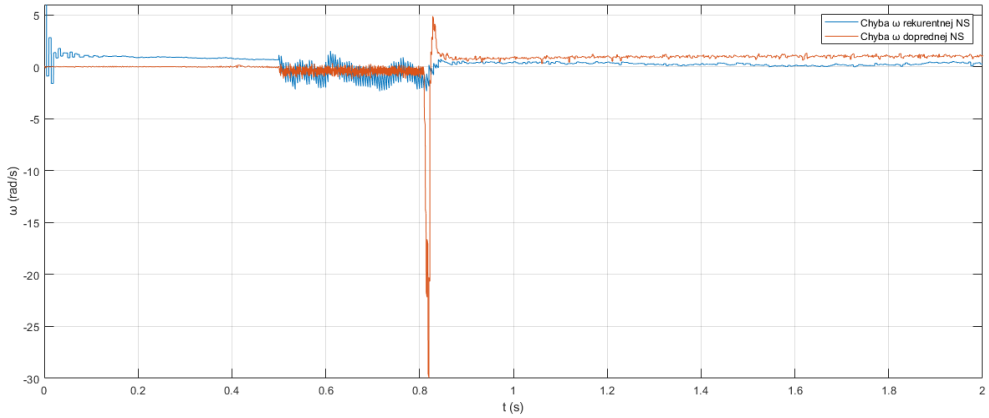


Obrázok 39 Priebeh odhadovanej a skutočnej rýchlosti pri nenaučenom rozbehu



Obrázok 40 Rozdiel medzi odhadovanou a skutočnou rýchlosťou pri nenaučenom rozbehu

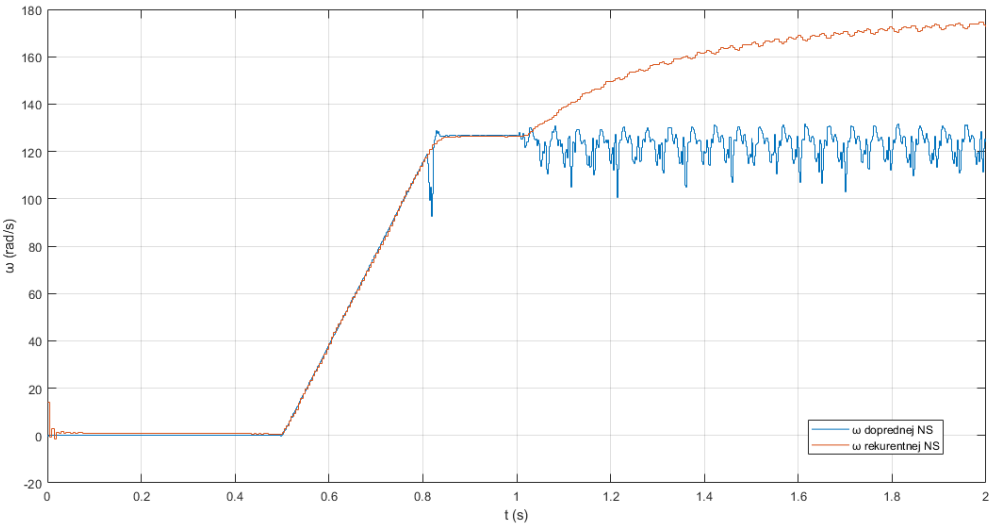
Na poslednom obrázku tejto podkapitoly porovnávame chyby odhadu rekurentnej a doprednej NS. V priebehoch vidíme, že pri nulovej uhlovej rýchlosti má lepší odhad dopredná NS. Zlý odhad rekurentnej je spôsobený prekmitom na začiatku merania. Dôvod, prečo vzniká tento prekmit sa nám nepodarilo zistiť. Po prechodnom deji sa ale odhad rektrentnej NS zlepší a pri ustálení uhlovej rýchlosti ju odhaduje lepšie ako dopredná. Na tomto obrázku ešte vidíme veľkú chybu odhadu doprednej NS pri zmene stavu z rozbehu na ustálenie na určitej uhlovej rýchlosti. Toto môžeme považovať výstup NS pri neznámom stave. Tento problém sa ešte viac prejaví pri používaní dopredných NS v regulácii, kde umiestnenie tejto NS do spätnej väzby pohonu spôsobí, že v prípade, keď nastane takáto náhla zmena odhadu (v skutočnosti sa takáto zmena rýchlosti na motore nemôže udiať) zlyhá celá regulácia.



Obrázok 41 Porovnanie chýb rekurentnej a doprednej NS

6.3 Odozva neurónových sietí na pridanie záťaže

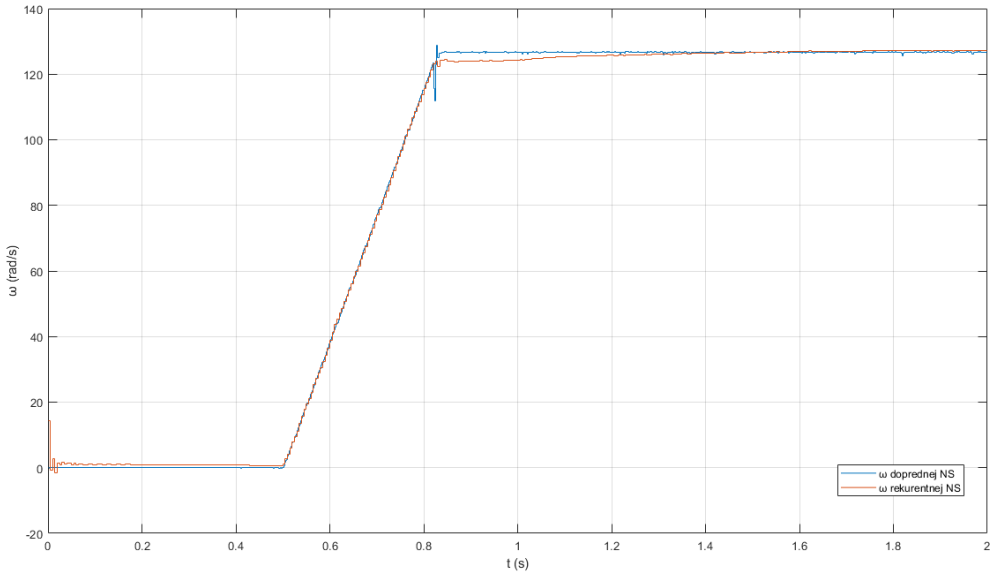
V tejto časti sme skúmali vplyv pripojenia záťaže na odhad elektrickej uhlovej rýchlosti rotora pomocou NS. Na prvé meranie sme použili otestované NS z predchádzajúcich podkapitol a z priebehov môžeme usúdiť, že v prípade nenaučenia NS na pripojenie záťaže (nastane zmena vstupných parametrov do NS) je odhad chybný. No na obrázku môžeme sledovať prístup oboch NS k vyriešeniu tohto problému.



Obrázok 42 Porovnanie odozvy nenaučenej rekurentnej a doprednej NS na záťaž

Keďže ma rekurentná NS spätnú väzbu, tak zlý odhad rýchlosti slúži ako vstup do siete, a ten spôsobí ďalší zlý odhad. Chyba sa tak zväčšuje, na rozdiel od doprednej NS. Do tej vstupuje len aktuálna a predchádzajúca vzorka vstupných premenných, a to spôsobí, že odhad je zlý, ale nezhoršuje sa s časom.

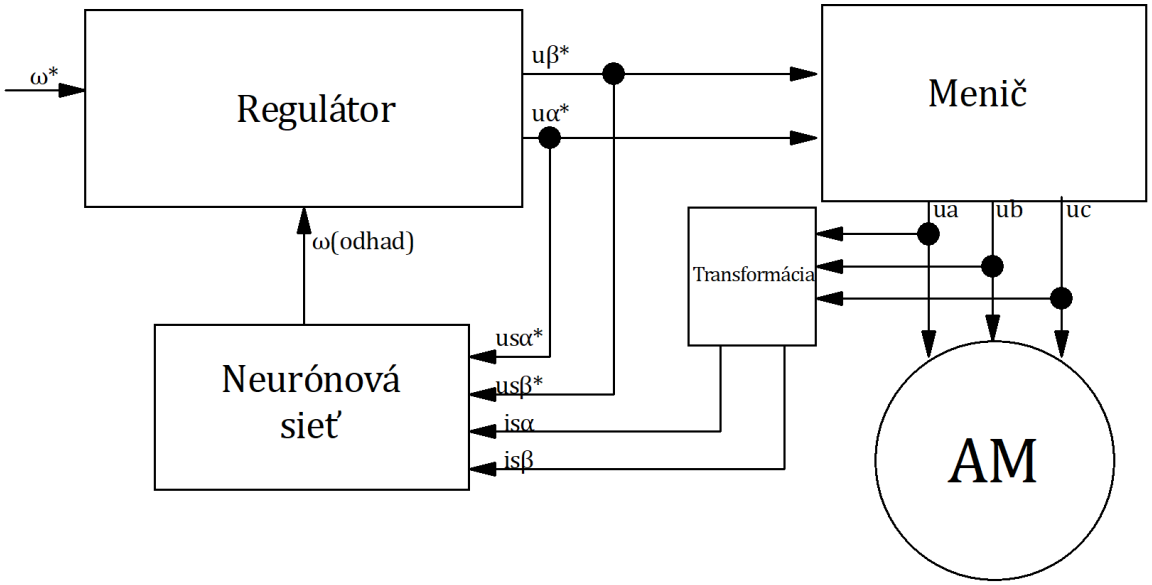
Na ďalšom obrázku je priebeh uhlových rýchlosti z už naučených NS na danú záťaž (tá sa pripojí v prvej sekunde). Odhad uhlovej rýchlosti je správny.



Obrázok 43 Porovnanie odozvy naučenej rekurentnej a doprednej NS na záťaž

KAPITOLA 7: BEZSENZOROVÁ REGULÁCIA ASYNCHRÓNNEHO MOTORA S VYUŽITÍM NEURÓNOVÝCH SIETÍ

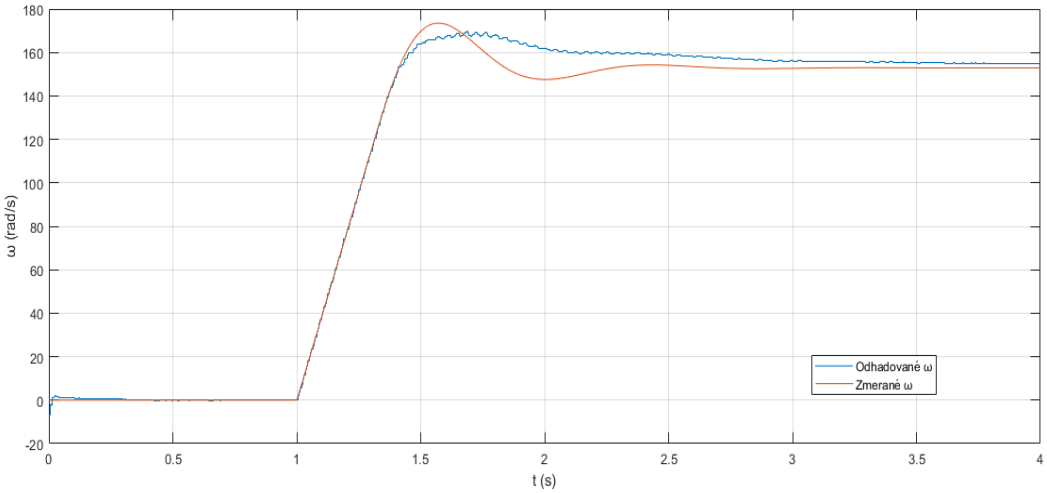
V tejto kapitole sa pokúsime riadiť asynchrónny motor pomocou vektorovej regulácie s využitím odhadnutej uhlovej rýchlosti z predchádzajúcej kapitoly. Na to sme použili rekurentnú NS (dôvod nepoužitia doprednej NS je tiež načrtnutý v predchádzajúcej kapitole). Schéma tejto regulácie je na nasledujúcom obrázku.



Obrázok 44 Zjednodušená schéma regulácie s NS

Na reguláciu sme museli vytvoriť novú NS. Je to kvôli tomu, že vzorkovacia perióda predchádzajúcich sietí je 5 ms, tá je príliš malá a novú periódu som zvolil 1ms. Tá reaguje dosť rýchlo na zmeny z regulácie. Regulátor je spínaný s periódou 0,1 ms. Tu v Simulinku vznikol problém pri spojení signálov s rôznymi frekvenciami, ktorý sme vyriešili použitím bloku "Rate Transition" na prevody frekvencií signálov.

Z priebehu vidíme, že NS dokáže rozbehnúť motor na požadovanú uhlovú rýchlosť. No nemôžeme z neho usudzovať, že každá vygenerovaná NS je vhodná na bezsenzorovú reguláciu asynchrónneho motora. Tento priebeh vznikol po dlhom optimalizovaní všetkých regulátorov a hľadaní správnych parametrov, či už v regulácii alebo pri učení NS.



Obrázok 45 Priebeh odhadovanej a skutočnej uhlovej rýchlosti pri bezsenzorovej regulácii

ZÁVER

7.1 Zhodnotenie výstupov práce

V tejto diplomovej práci som sa venoval využitiu umelých neurónových sietí vo vektorovej regulácii asynchrónneho motora. Keďže som sa umelej inteligencii a neurónovým sieťam nevenoval počas štúdia, všetky informácie o nich som tak musel načerpať počas práce na zadaní. Neviem teda s určitosťou povedať, ako dobre som pochopil princíp optimalizácie neurónových sietí a či sú všetky konštatované závery správne. Svoje tvrdenia som sa snažil potvrdiť použitím overených zdrojov, no keďže sa jedná o málo preskúmaný smer, bolo to v mnohých prípadoch náročné.

Jedným z prínosov tejto práce by ale mohol byť návod na implementáciu neurónových sietí v oblasti odhadu premenných. Hľadanie informácií o tom, ako pracovať s neurónovými sieťami v programe Matlab, bolo najťažšou časťou tejto práce. A tak text obsahuje súhrn poznatkov, ktoré som počas práce nadobudol a môže urýchliť proces implementácie iných neurónových sietí.

V prvej kapitole sme, čo najstručnejšie, opísali asynchrónny motor a vektorovú reguláciu, pretože na zmeranie vstupných premenných do siete a následnú kontrolu sme potrebovali jej model. Ten sme zostrojili v Simulinku a opísali v kapitole 4. Na pochopenie názvoslovia a fungovania neurónových sietí bolo potrebné v kapitole 2 vysvetliť ich základy. Poznatky z prvých dvoch kapitol sú potom spojené v kapitole 3, kde opisujeme využitie neurónových sietí v pohonoch. V kapitole 5 je potom vysvetlené generovanie a optimalizácia učenia dopredných a rekurentných sietí. Informácie v týchto kapitolách by mali poskytnúť všetky potrebné informácie na replikáciu tejto úlohy.

Hlavným výstupom z tejto práce je model simulácie elektrického pohonu a neurónových sietí v Simulinku. Je navrhnutý na jednoduchú a rýchlu prácu so sieťami a vhodný na použitie v ďalších úlohách.

V nasledujúcej časti poskytneme zhrnutie nameraných výsledkov a zhodnotenie použitých neurónových sietí. Výsledky nie sú presné a nemajú slúžiť na potvrdenie, či vyvrátenie hypotéz, keďže sa ťažko kvantifikujú. Z týchto priebehov len zhodnotíme, či sú priebehy podobné, lebo presnosť výsledku závisí od konfigurácie siete, správneho nastavenia parametrov, výpočtového výkonu a času učenia.

7.2 Zmerané výsledky

V praktickej časti diplomovej práce som ako prvé implementoval odhad elektrickej uhlovej rýchlosti najprv pomocou doprednej neurónovej siete a potom pomocou rekurentnej. Z merania a porovnania výsledkov som došiel k záveru, že oba typy neurónových sietí odhadujú správne uhlovú rýchlosť. Z mnohých testov (nie sú načrtnuté v práci) sme ale zistili, že dopredné NS odhadujú presnejšie uhlovú rýchlosť, ktorá bola vstupom v učení, no rekurentné NS dokážu odhadnúť lepšie aj uhlové rýchlosti (a priebehy),ktoré neboli vstupom.

Nakoniec sme použili rekurentnú NS v bezsenzorovej regulácii asynchrónneho motora. Z výsledkov meraní a simulácie môžeme povedať, že implementácia neurónových sietí do regulácie elektrických pohonov je náročná a vyžaduje presné naladenie systému.

7.3 Ďalšie vylepšenia

Ako už bolo spomínané v Kapitola 3:, NS majú veľký potenciál na uplatnenie v elektrických pohonoch a existuje veľa možností na ich ďalšie použitie. Jeden smer, ktorým by sa mohla posúvať táto úloha, je implementácia ďalších typov neurónových sietí a ich porovnanie. Ďalší smerovaním,

hlavne z pohľadu simulácie elektrických pohonov, je odhad iných parametrov asynchrónneho motora. Ako už bolo spomínané v úvode, odpor rotora je jeden z parametrov, ktorý nie je počas fungovania konštantný a je závislý na teplote. Presný odhad rotorového odporu s využitím umelej inteligencie by tak mohol priniesť zvýšenie účinnosti týchto pohonov.

Ďalší okruh aplikácie neurónových sietí je diagnostika motorov a určovanie ich porúch . Toto využitie má podľa nášho názoru veľký potenciál do budúcnosti. Implementácia takejto úlohy ale vyžaduje obšírne znalosti nielen z poľa elektrických pohonov a ich regulácie, ale aj z odboru neurónových sietí a umelej inteligencie.

Keďže boli všetky hypotézy v tejto práci overované v Simulinku na matematickom modeli, zanedbali sme niektoré parametre, ktoré sa prejavujú v reálnych pohonoch. Ďalším smerom teda môže byť overenie výsledkov na reálnom asynchrónnom motore. Môžu sa zmerať reálne vstupné premenné a podľa nich riadiť pohon. Jedným z moderných spôsobov implementácie by mohlo byť použitie programovateľného hradlového poľa (FPGA) na implementáciu neurónovej siete. Tiež je možné smerovanie s použitím moderných technológií, kde na jednom čipe je integrovaný mikroprocesor (použitie na riadenie pohonu) a FPGA (implementovaná neurónová sieť).

LITERATÚRA

[1] **Voženílek, Petr, Novotný, Vladimír a Mindl, Pavel.** *Elektromechanické meniče.* 2. vydání. Praha : Česká technika - nakladatelství ČVUT, 2015.

[2] **Electrical Technology.** https://www.electricaltechnology.org/. [Online] [Dátum: 30. júl 2021.] https://www.electricaltechnology.org/2020/05/three-phase-induction-motor.html.

[3] **Kobrle, Pavel a Pavelka, Jiří.** *ELEKTRICKÉ POHONY A JEJICH ŘÍZENÍ.* 3. vydání. Praha : Česká technika - nakladatelství ČVUT, 2016.

[4] **Lipčák, Ondřej.** *Vektorové řízení asynchronního motoru pomocí DSP.* Praha : České vysoké učení technické v Praze, 2018. diplomová práce.

[5] **Kobrle, Pavel.** *Odvození matematického modelu asynchronního motoru a vektorově orientované metody regulace.* Podkladový materiál ke cvičením z předmětu „Elektrické pohony a trakce". s.l. : České vysoké učení technické v Praze, Fakulta elektrotechnická, 2017.

[6] **Quang, Nguyen Phung a Dittrich, Jorg-Andreas.** *Vector Control of Three-Phase AC Machines.* Berlin : Springer-Verlag, 2008. Zv. I. ISBN: 978-3-540-79028-0.

[7] **Wikipedia.** Artificial neural network. *wikipedia.org.* [Online] [Dátum: 31. júl 2021.] https://en.wikipedia.org/wiki/Artificial_neural_network.

[8] **Nazir, Nouman.** *Introduction to Artificial Neural Networks & Hidden Layer.* Gujrat : University of Gujrat, 2015.

[9] **Wilson, Bill.** The Machine Learning Dictionary. *www.cse.unsw.edu.au.* [Online] 2021. August 9. [Dátum: 9. August 2021.] https://web.archive.org/web/20180826151959/http://www.cse.unsw.edu.au/~billw/mldict.html#activnfn.

[10] **Livingstone, David J.** *Artificial Neural Networks: Methods and Applications.* Totowa : Humana Press. LLC, 2008. ISBN: 978-1-58829-718-1.

[11] **Wikipedia.** Reinforcement learning. *wikipedia.org.* [Online] [Dátum: 9. August 2021.] https://en.wikipedia.org/wiki/Reinforcement_learning.

[12] **Brownlee, Jason.** How to Choose an Activation Function for Deep Learning. *machinelearningmastery.com.* [Online] 22. január 2021. [Dátum: 1. august 2021.] https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/.

[13] **Feng, Junxi, a iní.** *Reconstruction of porous media from extremely limited information using conditional generative adversarial networks.* Chengdu, China : College of Electronics and Information Engineering, Sichuan University, 2019.

[14] **Bolton, Kris;.** A QUICK INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS (PART 2). [Online] 5. Jún 2018. [Dátum: 9. August 2021.] https://krisbolton.com/a-quick-introduction-to-artificial-neural-networks-part-2.

[15] **Demuth, Howard a Beale, Mark.** *Neural Network Toolbox.* 4.0.4. s.l. : MathWorks, Inc., 2004.

[16] **Koivo, Heikki N.** *NEURAL NETWORKS: Basics using MATLAB Neural Network Toolbox.* 2008.

[17] **Coulom, M. Rémi.** *Reinforcement Learning Using Neural Networks, with Applications to Motor Control.* Grenoble : INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, 2002.

[18] **Guyon, Isabelle.** *A Scaling Law for the Validation-Set Training-Set Size Ratio.* s.l. : AT&T Bell Laboratories, 1997.

[19] **Vojt, Ján.** *Deep neural networks and their implementation.* Prague : Charles University in Prague, Faculty of Mathematics and Physics, 2016.

[20] **Rashid, Muhammad H.** *Power Electronics Handbook.* Oxford : Elsevier, Inc., 2018. ISBN: 978-0-12-811407-0.

[21] **Haykin, Simon.** *Neural Networks and Learning Machines.* Tretia. New Jersey : Pearson Education, Inc, 2009. ISBN-13: 978-0-13-147139-9.

[22] **Great Learning Team.** Types of Neural Networks and Definition of Neural Network. *Great Learning.* [Online] 29. Apríl 2020. [Dátum: 9. August 2021.] https://www.mygreatlearning.com/blog/types-of-neural-networks/.

[23] **Mijwel, Maad M., Esen, Adam a Shamil, Aysar.** *Overview of Neural Networks.* Baghdad : Computer Engineering Techniques Department, Baghdad College of Economics Sciences University, Iraq, 2019.

[24] **Trzynadlowski, Andrzej M.** *Control of Induction Motors.* San Diego : Academic Press, 2001. ISBN:0-12-701510-8.

[25] **Bose, Bimal K.** *Modern Power Electronics and AC Drives.* Upper Saddle River : Prentice-Hall, Inc., 2002. ISBN: 0-13-016743-6.

[26] **Vas, P., Stronach, A. F. a Neuroth, M.** *APPLICATION OF CONVENTIONAL AND AI-BASED TECHNIQUES IN SENSORLESS HIGH-PERFORMANCE TORQUE-CONTROLLED INDUCTION MOTOR DRIVES.* Aberdden : Department of Engineering, University of Aberdeen, 1998.

[27] **Finch, J. W. a Giaouris, D.** *Controlled AC Electrical Drives.* s.l. : IEEE Transactions on Industrial Electronics, 2008.

[28] **Orlowska-Kowalska, Teresa a Dybkowski, Mateusz.** Stator-Current-Based MRAS Estimator for a Wide Range Speed-Sensorless Induction-Motor Drive. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.* VOL. 57, 2010, NO. 4.

[29] **Gunabalan, R., Subbiah, V. a Ramy Reddy, B.** Sensorless Control of Induction Motor with Extended Kalman Filter on TMS320F2812 Processor. *International Journal of Recent Trends in Engineering.* Vol 2, 2009, No. 5.

[30] **Chan, Tze-Fun a Shi, Keli.** *Applied Intelligent Control of Induction Motor Drives.* I. Singapore : John Wiley & Sons (Asia) Pte Ltd, 2011. ISBN: 978-0-470-82556-3.

[31] **Ozer, A. Bedri a Akin, Erhan.** *ARTIFICIAL NEURAL NETWORK BASED SPEED ESTIMATOR FOR VECTOR CONTROLLED INDUCTION MOTOR .* Elazig : Firat University, Department of Computer Engineering.

[32] **Bauer, Ján a Lipčák, Ondřej.** *3. Vektorová regulace.* České vysoké učení technické v Praze, Fakulta elektrotechnická : Podkladový material k prednáškam z předmětu "Elektrické pohony a trakce", 2020.

[33] **The MathWorks, Inc.** feedforwardnet. *www.mathworks.com.* [Online] The MathWorks, Inc. [Dátum: 12. August 2021.] https://www.mathworks.com/help/deeplearning/ref/feedforwardnet.html?searchHighlight=feedforwardnet&s_tid=srchtitle#bu_do1g.

[34] —. train. *www.mathworks.com.* [Online] The MathWorks, Inc. [Dátum: 12. August 2021.] https://www.mathworks.com/help/deeplearning/ref/network.train.html.

[35] —. layrecnet. *www.mathworks.com.* [Online] The MathWorks, Inc. [Dátum: 12. August 2021.] https://www.mathworks.com/help/deeplearning/ref/layrecnet.html.

[36] —. preparets. *www.mathworks.com.* [Online] The MathWorks, Inc. [Dátum: 12. August 2021.] https://www.mathworks.com/help/deeplearning/ref/preparets.html.

[37] **Hasan, Mahmudul.** *Artificial Neural Network Based Speed Estimator for Sensorless Field Oriented Control of Three Phase Induction Motor.* Rajshahi, Bangladesh : Rajshahi University of Engineering & Technology, 2019.

[38] **Vas, Peter.** *Sensorless Vector and Direct Torque Control .* Oxford, New York, Tokyo : OXFORD UNIVERSITY PRESS , 1998.

[39] **The MathWorks, Inc.** con2seq. *www.mathworks.com.* [Online] The MathWorks, Inc. [Dátum: 12. August 2021.] https://www.mathworks.com/help/deeplearning/ref/con2seq.html.

[40] —. trainlm. *www.mathworks.com.* [Online] The MathWorks, Inc. [Dátum: 12. August 2021.] https://www.mathworks.com/help/deeplearning/ref/trainlm.html.

PRÍLOHA A: ZOZNAM SYMBOLOV A SKRATIEK

A.1 Zoznam symbolov

$cos\varphi$ (-)	účinník
e (-)	chyba neurónu
E (-)	stredná kvadratická chyba vrstvy NS
f (s^{-1})	frekvencia napájacieho napätia, resp. prúdu
i, I (A)	prúd
J ($J \cdot m^{-2}$)	moment zotrvačnosti
L_1 (H)	celková statorová indukčnosť
L_2 (H)	celková rotorová indukčnosť
$L_{1\sigma}$ (H)	statorová rozptylová indukčnosť
$L_{2\sigma}$ (H)	rotorová rozptylová indukčnosť
L_m (H)	magnetizačná indukčnosť
M (Nm)	moment
M_Z (Nm)	záťažný moment
n (min^{-1})	otáčavá rýchlosť rotora
p_p (-)	počet polpárov
P (W)	výkon
R_1 (Ω)	statorový elektrický odpor
R_2 (Ω)	rotorový elektrický odpor
t (s)	čas
u, U (A)	napätie
U_{DC} (V)	napätie v jednosmernom medziobvode striedača
θ (rad)	okamžitý uhol medzi súradnicovými systémami $\alpha\beta$ a dq
ϑ_k (rad)	okamžitý uhol medzi stojacim a obecným rotujúcim súradnicovým systémom
Ψ (Wb)	spriahnutý magnetický tok
Ψ_1 (Wb)	statorový spriahnutý magnetický tok
Ψ_2 (Wb)	rotorový spriahnutý magnetický tok
ω (s^{-1})	elektrická uhlová rýchlosť rotoru
Ω (s^{-1})	mechanická uhlová rýchlosť rotoru

abc	trojfázový systém statora
a, b, c	označenie jednotlivých fáz statora
$\alpha\beta$	ortogonálny súradnicový systém zviazaný so statorom
dq	ortogonálny súradnicový systém rotujúci synchrónnou rýchlosťou
xy	obecný súradnicový systém
e	Eulerovo číslo e=2,718...

A.1.1 Zoznam skratiek

AF	aktivačná funkcia
AM	asynchrónny motor
DC	Direct Current (jednosmerný prúd)
DSP	Digital Signal Processor (digitálny signálový procesor)
DTC	Direct Torque Control (priame riadenie momentu)
EKF	Extended Kalman Filter (rozšírený Kalmanov filter)
EMF	electromotive force (elektromotorická sila)
FPGA	Field Programmable Gate Array (programovatelné hradlové pole)
LSTM	Long Short-term Memory (typ NS)
MRAS	Model Reference Adaptive System (adaptívny pozorovateľ s referenčným modelom)
NS	neurónová sieť
PI	regulátor s proporčnou a integračnou zložkou
PWM	Pulse-Width Modulation (pulzne šírková modulácia)
UI	umelá inteligencia

PRÍLOHA B: POPIS PRÍKAZOV Z NN TOOLBOX

B.1 con2seq

Príkaz mení súbežné vektory na sekvenčné. Deep Learning Toolbox™ vníma súbežné vektory ako matica a sekvenčné vektory ako pole buniek (druhý index je časový krok). Štruktúra príkazu je *S = con2seq(b)*, kde (39):

- *S* matica s veľkosťou RxTs ,
- *b* pole s veľkosťou 1xTs alebo vektor Rx1.

B.2 feedforwardnet

Funkcia vracia doprednú NS. Štruktúra príkazu je *net = feedforwardnet(hiddenSizes,trainFcn)* kde:

- *hiddenSizes* veľkosť skrytých vrstiev v sieti definovaná ako riadkový vektor. Dĺžka vektora určuje počet skrytých vrstiev. Základná hodnota je 10.
- *trainFcn* výber trénovacej funkcie. Základný algoritmus je *trainlm.* Ďalšie príklady možných algoritmov sú na (33).

B.3 layrecnet

Príkaz vracia rekurentnú NS. Syntax je *net = layrecnet(layerDelays,hiddenSizes,trainFcn)* kde:

- *layerDelays* riadkový vektor, ktorý definuje odklad (delay). Môže mať hodnotu rastúcich pozitívnych čísel alebo nulu. Základná hodnota je [1:2].
- *hiddenSizes* veľkosť skrytých vrstiev, ako v prípade *feedforwardnet.*
- *trainFcn* výber trénovacej funkcie. Základný algoritmus takisto je *trainlm* (35).

B.4 trainParam

Tento príkaz nie je samostatná funkcia. Pomocou neho sa len pristupuje k argumentom na nastavovanie učenia NS. Výber týchto argumentov je podmienený výberom trénovacieho algoritmu. Pre náš prípad (výber algoritmu *trainlm*) má príkaz štruktúru *net.trainParam.fcn = val*, kde *net* je objekt neurónovej siete a *val* je veľkosť nastavovaného argumentu. Za *fcn* potom môžeme dosadiť (uvádzam príklady najpoužívanejších príkazov) (40):

- *epochs* maximálny počet epoch na učenie (základná hodnota je 1000),
- *goal* cieľ učenia (základná hodnota je 0),
- *max_fail* maximálny počet zlyhaní v overovaní (základná hodnota je 6),
- *min_grad* minimálny gradient (základná hodnota je 1e-7),
- *time* maximálny čas na učenie v sekundách (základná hodnota je nekonečno).

B.5 preparets

Funkcia preformátuje dáta do vhodného tvaru na trénovanie. Keďže sa tu definujú počiatočné odklady a váhy chýb, je vhodná pre rekurentné. Definujeme ju pomocou príkazu *[Xs,Xi,Ai,Ts,EWs] = preparets(net,Xnf,Tnf,Tf,EW)*, kde vstupné argumenty sú (36):

- *net* neurónová sieť,
- *Xnf* vstupy bez spätnej väzby,
- *Tnf* cieľové hodnoty bez spätnej väzby,
- *Tf* cieľové hodnoty so spätnou väzbou,
- *EW* váhy chýb (základná hodnota je {1}).

Príkaz potom vracia (36):

- *Xs* posunuté vstupy,
- *Xi* počiatočné stavy odkladu vstupu,
- *Ai* počiatočné stavy odkladu vrstiev,
- *Ts* posunuté cieľové hodnoty,
- *EWs* posunuté váhy chýb.

B.6 train

Funkcia slúži na trénovanie plytkej NS. Má štruktúru *trainedNet = train(net,X,T,Xi,Ai,EW)*, kde *trainedNet* je konečná vytrénovaná sieť a parametre sú (34):

- *net* vygenerovaný objekt siete, napríklad príkazom *feedforwardnet*.
- *X* vstupy do siete, definované ako matica alebo pole buniek.
- *T* cieľové hodnoty, definované ako matica alebo pole buniek.
- *Xi* počiatočné podmienky oneskorenia vstupu. Vygenerované cez *preparets*.
- *Ai* počiatočné podmienky oneskorenia vrstiev. Vygenerované cez *preparets*.
- *EW* váhy chýb. Vygenerované cez *preparets*.

PRÍLOHA C: PARAMETRE ASYNCHRÓNNEHO MOTORA

C.1 Štítkové parametre AM

V úlohe bol použitý trojfázový asynchrónny motor s rotorovým vinutím. Statorové vinutie je spojené do hviezdy. Štítkové parametre stroja sú:

P_n	12 kW
I_n	22 A
U_n	380 V
f_n	50 Hz
$\cos\varphi_n$	0.8
n_n	1460 min^{-1}
p_p	2

Tabuľka 1 Štítkové parametre AM

C.2 Zmerané parametre AM

Parametre stroja, ktoré sa používajú v matematickom modeli, boli zmerané pomocou klasických skúšok nakrátko a naprázdno. Moment zotrvačnosti bol zmeraný dobehovou metódou. Tieto parametere sú v nasledujúcej tabuľke:

R_1	370 mΩ
R_2	225 mΩ
$L_{1\sigma}$	2.27 mH
$L_{2\sigma}$	2.27 mH
L_m	82.5 mH
L_1	84.77 mH
L_2	84.77 mH
J	0.4 $kg \cdot m^{-2}$

Tabuľka 2 Zmerané parametre AM