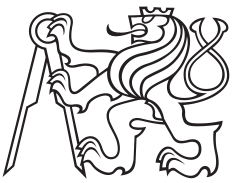
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra elektrických pohonů a trakce

Vektorové řízení asynchronního motoru pomocí DSP

Field Oriented Control of Induction Motor by TI 28335

Bc. Ondřej Lipčák
Studijní program: Elektrotechnika, energetika a management
Studijní obor: Elektrické stroje, přístroje a pohony

květen 2018
Vedoucí práce: Ing. Jan Bauer, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lipčák** Jméno: **Ondřej** Osobní číslo: **420239**

Fakulta/ústav: **Fakulta elektrotechnická**

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

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

Studijní obor: **Elektrické stroje, přístroje a pohony**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vektorové řízení asynchronního motoru pomocí DSP

Název diplomové práce anglicky:

Field Oriented Control of Induction Motor by TI 28335

Pokyny pro vypracování:

1) Popište princip metody vektorového řízení asynchronního motoru
2) Proveďte rešerši modelů asynchronního motoru používaných pro vektorové řízení
3) Prostudujte možnosti propojení DSP TI28335 s laboratorním měničem
4) Realizujte algoritmus vektorového řízení na platformě TI28335
5) Otestujte různé varianty modelu asynchronního motoru s realizovaným algoritmem vektorového řízení

Seznam doporučené literatury:

[1] JAVŮREK, Jiří. Regulace moderních elektrických pohonů. Praha: Grada, 2003.
[2] VAS, Peter. Sensorless vector and direct torque control. Oxford: Oxford University Press, 1998. Monographs in electrical and electronic engineering.
[3] QUANG, Nguyen Phung a Jörg-Andreas DITTRICH. Vector control of three-phase AC machines: system development in the practice. Berlin: Springer, c2008. Power systems.
[4] ORLOWSKA-KOWALSKA, T. and M. DYBKOWSKI. Stator-Current-Based MRAS Estimator for a Wide Range Speed-Sensorless Induction-Motor Drive. IEEE Transactions on Industrial Electronics. 2010, vol. 57, iss. 4, pp. 1296?1308.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Bauer, Ph.D., katedra elektrických pohonů a trakce FEL

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

Datum zadání diplomové práce: **26.04.2018** Termín odevzdání diplomové práce: **25.05.2018**

Platnost zadání diplomové práce: **30.09.2019**

Ing. Jan Bauer, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. 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

Poděkování / Prohlášení

Na tomto místě bych velice rád poděkoval svému vedoucímu, doktoru Bauerovi, neboť bez jeho cenných rad a hlavně trpělivosti při zodpovídání mých dotazů by práce měla o poznání horší kvalitu. Rovněž si cením poskytnutého zázemí pro tvorbu tohoto textu. Další dík patří docentu Kynclovi z katedry elektroenergetiky, a to za poskytnuté rady ohledně numerického řešení rovnic matematického modelu.

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

V Praze dne 1.5.2018

.....

Abstrakt / Abstract

Hlavní náplní této práce je praktická implementace jedné z možných variant vektorového řízení asynchronního motoru do signálového procesoru TMS320F28335 od společnosti Texas Instruments. Zvolená struktura zahrnuje oddělenou regulaci tokotvorné a momentotvorné složky vektoru statorového proudu s přímým zjišťováním transformačního úhlu u Parkovy transformace na základě výpočtu ze složek vektoru rotorového toku. Ty jsou obdrženy z řešení rovnic proudového I-n modelu v souřadnicovém systému spojeném se statorem.

V prvních kapitolách je provedeno odvození matematického modelu asynchronního motoru. Dále je pak probrán princip vektorového řízení včetně navržené regulační struktury. Následují informace o hardwarových a softwarových prostředcích použitých během vývoje, popis konfigurace použitého procesoru a posléze popis programové realizace jednotlivých funkčních bloků. Práci uzavírá rozbor přiložených průběhů zachycujících regulační děje při rozběhu motoru a konečně pak porovnání Eulerovy metody s metodou Runge-Kutta 4. řádu při řešení proudového I-n a napěťového U-I modelu.

Klíčová slova: asynchronní motor, vektorové řízení, digitální signálový procesor, proudový model, napěťový model, Texas Instruments TMS320F28335, Clarkové transformace, Parkova transformace, Eulerova metoda, metoda Runge-Kutta 4. řádu

The main aim of this thesis is a practical implementation of one of possible variants of vector control of induction motor into Texas Instruments TMS320F28335 digital signal processor. The chosen control structure includes separate regulation of torque and flux producing components of stator current vector with direct determination of Parke transform angle. The transformation angle is calculated out of components of rotor flux vector that are obtained by solution of equations of so called I-n current model in stator fixed coordinate system.

In the first chapters, a derivation of induction motor mathematical model is performed. Next, a principle of vector control is explained along with presentation of designed control structure. Next chapters include brief information about software and hardware used during development and a description of configuration of the used processor. After that, sofware realization of respective blocks of the vector control is discussed. One of the final parts of this thesis consist of an analysis of measured motor quantities captured during mechanical transient state. Finally, a comparison of Euler method and Runge-Kutta 4^{th} order method in terms of obtained results from solution of current I-n and voltage U-I models is presented.

Keywords: induction motor, vector control, digital signal processor, current model, voltage model, Texas Instruments TMS320F28335, Clarke transform, Parke transform, Euler method, Runge-Kutta 4^{th} order method

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

V dřívějších dobách dominovaly aplikačním odvětvím s potřebou regulace otáček v širokém rozsahu prakticky výlučně stejnosměrné komutátorové motory. Ty mají tu výhodu, že řízení jejich rychlosti, které se realizuje převážně změnou napětí na kotvě, je jednoduché a přímočaré. K tomuto účelu se zpočátku používaly odporníky nebo rotační meniče (např. Ward-Leonardovo soustrojí), což však představovalo ztrátovou regulaci. Situace se zlepšila s nástupem a rozvojem výkonové elektroniky, kdy bylo nyní možné nahradit rozměrné a nehospodárné odporníky např. tyristorovými usměrňovači nebo stejnosměrnými měniči napětí, čímž bylo možné dosáhnout značných úspor energie.

Problémem a nevýhodou stejnosměrného stroje je však jeho konstrukční složitost a z toho plynoucí vyšší cena, poruchovost a zvýšené nároky na údržbu. Jeho nejslabším článkem je komutátor, který bývá často zdrojem poruch a elektromagnetického rušení vlivem jiskření mezi kartáči a lamelami. Tyto nevýhody by se podařilo odstranit použitím motorů asynchronních, které jsou z konstrukčního hlediska jednoduché a spolehlivé, avšak mají přesně opačnou nevýhodu – tou je mnohem obtížnější řízení otáček, které vyžaduje změnu frekvence napájecího systému.

Počátky zrodu vektorového řízení lze vysledovat už na přelomu sedmdesátých let minulého století. Jeho ideou je za přijmutí některých zjednodušujících předpokladů matematicky popsat asynchronní motor a získané rovnice následně převést pomocí lineárních transformací na regulační strukturu, která je formálně shodná s tou u stejnosměrného cize buzeného stroje. Praktickou realizaci této myšlenky umožnil až komerční nástup mikroprocesorů v osmdesátých letech. Masovému nasazení vektorově regulovaných asynchronních motorů však pořád bránila vysoká cena a složitost celého systému.

V současné době se podařilo tuto bariéru díky rozvoji mikroprocesové techniky a výkonové elektroniky překonat a dnes tak již nic nebrání rozšíření pohonů s asynchronními motory do náročných průmyslových a trakčních aplikací. Zároveň v této oblasti neustále probíhá navazující vývoj a výzkum s cílem dalšího zlepšování.

Tato práce si klade za cíl navrhnout a realizovat pomocí digitálního signálového procesoru algoritmus jedné z možných variant vektorového řízení a dále pak porovnat vliv numerických metod řešení matematického modelu na chování pohonu. První část práce se zabývá zejména matematickým popisem asynchronního stroje a odvozením modelů používaných v rámci vektorového řízení. Dále je pak popsána zvolená metoda vektorového řízení včetně blokového schématu navržené struktury regulace a řízení napájecího měniče. Následující dvě kapitoly popisují prostředky pro vývoj, zejména pak použitý signálový procesor a jeho konfiguraci nutnou pro správné fungování celého řízení. Další části jsou pak věnovány samotné implementaci jednotlivých bloků vektorového řízení do signálového procesoru. Práci uzavírají reálné naměřené průběhy během regulačních dějů a průběhy dokumentující porovnání dvou různých numerických metod při řešení rovnic modelů asynchronního stroje.

V některých kapitolách je pro lepší názornost a pochopení problematiky proveden rozbor vytvořeného kódu včetně přiložení ukázek. Celý projekt doplněný komentáři v anglickém jazyce je pak k dispozici v rámci přílohy.

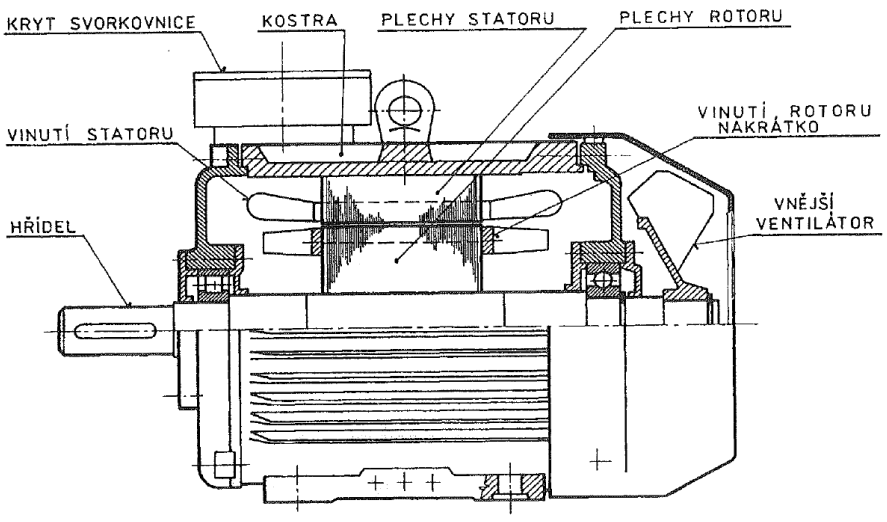
Kapitola 2
Asynchronní motor

Asynchronní motor (někdy také indukční motor z anglického „induction machine“) je nejpoužívanějším motorem v elektrických pohonech. Jeho jednoduchá robustní konstrukce, nízká výrobní cena, malé nároky na údržbu, provozní spolehlivost a proveditelnost v širokém rozsahu výkonů ho předurčují pro nasazení v průmyslových aplikacích i v oblastech běžného života. Jeho hlavní nevýhodu – obtížné řízení rychlosti, se s pomocí dnes dostupných moderních měničů výkonové elektroniky a řídicích počítačů podařilo prakticky eliminovat. Dnes už tedy zdaleka není použití asynchronních motorů vázáno na aplikace s přibližně konstantními otáčkami.[1]

2.1 Konstrukce

Podélný řez asynchronního motoru je znázorněn na obr. 2.1. Stator je složen z navzájem izolovaných plechů pro elektrotechniku. Po vnitřním obvodu je opatřen drážkami pro uložení vinutí, které bývá nejčastěji třífázové. Konce vinutí jsou vyvedeny na svorkovnici, která se nachází na vnější kostře. Svorkovnice je uspořádána tak, aby bylo možné spojit statorové vinutí do trojúhelníku nebo do hvězdy.[1]

Rotor je, stejně jako stator, také složen z plechů pro elektrotechniku. Vinutí může být tvořeno kotvou nakrátko (klecové) nebo kotvou vinutou. Kotvu nakrátko tvoří tyče v drážkách rotoru spojené v čelech kruhy dokrátka. U vinuté kotvy je v rotorových drážkách uloženo třífázové vinutí vyvedené na kroužky, přes které lze pomocí kartáčů připojit do obvodu rotoru spouštěcí odpor.[1]



Obrázek 2.1. Podélný řez asynchronního stroje s kotvou nakrátko [1]

2.2 Princip činnosti

Mějme třífázový asynchronní motor, jehož cívky jsou stejně uspořádané, mají stejný počet závitů a jsou po obvodu stroje natočeny o úhel, který odpovídá elektrickému fázovému posuvu třífázového souměrného harmonického napájecího systému, tj. o 120° neboli $2\pi/3$ radiánů. Po připojení motoru na tento nápájecí systém začnou vinutími protékat harmonické souměrné třífázové proudy. Magnetomotorická napětí vzniklá v důsledku přítomných fázových proudů se sčítají a dohromady tvoří jeden rotující fázor s konstantní amplitudou. Vzniká tak točivé magnetické pole o konstantní amplitudě, jež se ve vzduchové mezeře stroje otáčí rychlostí danou frekvencí napájecího systému a počtem pólů.[1]

Vzniklé točivé magnetické pole indukuje za předpokladu různé vzájemné rychlosti otáčení pole a rotoru do uzavřeného rotorového obvodu napětí, které vyvolá proudy, jež podle Lenzova zákona působí proti příčině svého vzniku. Vzniklý moment tak začne točit rotorem ve smyslu točení pole. V důsledku snížení vzájemné rychlosti točivého pole a rotoru poklesne i velikost indukovaného napětí a vzniklých proudů. V případě nulové relativní rychlosti by se do rotoru indukovalo nulové napětí a stroj by měl nulový moment. Uvažujeme-li nezatížený motor připojený na tvrdou napájecí síť, tak je zřejmé, že motor vždy potřebuje alespoň minimální moment pro krytí mechanických ztrát, proto se i rotor nezatíženého motoru otáčí o něco pomaleji, než je rychlost točivého magnetického pole. Odtud název asynchronní.[1]

2.3 Základní veličiny a jejich značení

V této kapitole je uveden přehled nejzákladnějších pojmů, které jsou dále používány při matematické analýze asynchronního stroje.

Elektrickou úhlovou rychlost hřídele ω získáme z mechanické úhlové rychlosti Ω přepočtem pomocí pólpárů p_p

$$\omega = p_\mathrm{p}\Omega. \tag{2.1}$$

Vztah (2.1) platí pro přepočet mezi elektrickými a mechanickými úhlovými veličnami obecně. Nadále budou mechanické úhlové rychlosti obecně značeny velkým řeckým písmenem omega, elektrické úhlové rychlosti malým řeckým písmenem omega.

Dále definujeme elektrickou úhlovou rychlost točivého magnetického pole ω_s (tzv. synchronní rychlost) při napájení statoru napětím, resp. proudem o kmitočtu f [1]

$$\omega_\mathrm{s} = 2\pi f. \tag{2.2}$$

Dále se zavádí tzv. skluz s, který lze vyjádřit na základě předchozích veličin jako [1]

$$s \ = \frac{\omega_\mathrm{s} - \omega}{\omega_\mathrm{s}}. \tag{2.3}$$

Pro elektrickou skluzovou úhlovou rychlost, tj. rychlost rotoru vůči poli, lze psát [1]

$$\omega_\mathrm{slip} = \omega_\mathrm{s} - \omega = s\ \omega_\mathrm{s}. \tag{2.4}$$

Elektrickou úhlovou rychlost rotoru lze pak vyjádřit také jako [1]

$$\omega = (1 - s)\omega_\mathrm{s}. \tag{2.5}$$

Všechny výše zavedené elektrické úhlové veličiny mají samozřejmě i svůj mechanický ekvivalent.

2.4 Matematický popis asynchronního stroje

2.4.1 Předpoklady pro odvození

Abychom dosáhli rozumné matematické složitosti popisovaného problému, je nutné před odvozením matematického modelu asynchronního stroje přijmout některé zjednodušující předpoklady:[2–3]

- máme k dispozici třífázovou souměrnou napájecí soustavu, kde všechna napětí jsou harmonická,
- tloušťka vzduchové mezery mezi statorem a rotorem je konstantní (zanedbává se drážkování),
- statorová vinutí jsou rozložena po obvodu vzduchové mezery sinusově, vinutí jednotlivých fází jsou vůči sobě natočena o 120°,
- ztráty v železe jsou zanedbány,
- není uvažován vliv sycení magnetického obvodu,
- statorová vinutí jsou souměrná, tj. činné odpory, indukčnosti a vzájemné indukčnosti jednotlivých fází statoru jsou identické

$$\begin{aligned} R_\mathrm{a} &= R_\mathrm{b} = R_\mathrm{c} = R_1 \\ L_\mathrm{a} &= L_\mathrm{b} = L_\mathrm{c} = L_\mathrm{s} \\ L_\mathrm{ab} &= L_\mathrm{ac} = L_\mathrm{bc} = M_\mathrm{s}, \end{aligned}$$

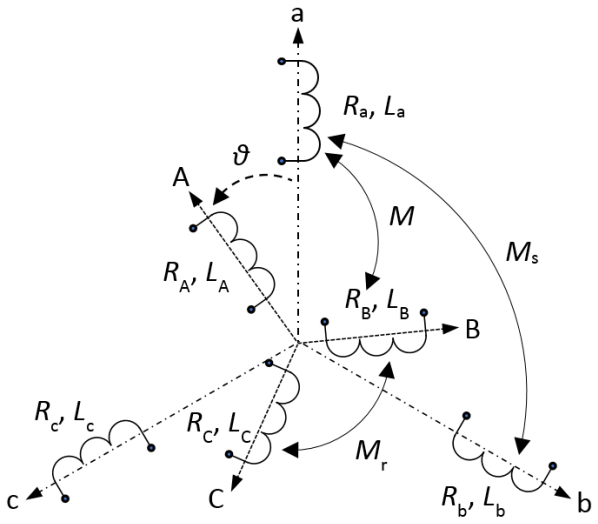
- rotorová vinutí jsou souměrná, tj. činné odpory, indukčnosti a vzájemné indukčnosti jednotlivých fází rotoru jsou identické

$$\begin{aligned} R_\mathrm{A} &= R_\mathrm{B} = R_\mathrm{C} = R_2 \\ L_\mathrm{A} &= L_\mathrm{B} = L_\mathrm{C} = L_\mathrm{r} \\ L_\mathrm{AB} &= L_\mathrm{AC} = L_\mathrm{BC} = M_\mathrm{r}. \end{aligned}$$

Výše uvedené předpoklady nepřímo naznačují, že uvažujeme stroj s třífázovým vinutím na statoru i rotoru. Celá situace je znázorněna na obr. 2.2. Vinutí na statoru a rotoru mají obecně rozdílný počet závitů a krok. Z toho důvodu je výhodné použít spřažené magnetické toky Ψ (součin počtu závitů cívky a celkového magnetického toku cívkou) a přepočítat rotorové veličiny na stator.[2] Výsledné magnetomotorické napětí se nesmí přepočtem změnit.[2] Odvození příslušných přepočítávacích vztahů lze nalézt např. v [1].

Všechny výše uvedené indukčnosti jsou uvažovány konstantní. Vzájemné indukčnosti M odpovídajících si fází statoru a rotoru se mění s úhlem natočení ϑ rotoru. Např. vzájemná indukčnost fází „a“ – „A“ je maximální kladná při $\vartheta = 0°$, minimální záporná při $\vartheta = 180°$ a nulová jestliže $\vartheta = 90°$ nebo $\vartheta = 270°$. Tuto závislost lze vyjádřit pomocí matematické funkce cos [4]

$$\begin{aligned} L_\mathrm{aA} &= L_\mathrm{Aa} = L_\mathrm{bB} = L_\mathrm{Bb} = L_\mathrm{cC} = L_\mathrm{Cc} = M\cos\vartheta \\ L_\mathrm{aB} &= L_\mathrm{Ab} = L_\mathrm{bC} = L_\mathrm{Bc} = L_\mathrm{cA} = L_\mathrm{Ca} = M\cos(\vartheta - \frac{2\pi}{3}) \\ L_\mathrm{aC} &= L_\mathrm{Ac} = L_\mathrm{bA} = L_\mathrm{Ba} = L_\mathrm{cB} = L_\mathrm{Cb} = M\cos(\vartheta + \frac{2\pi}{3}). \end{aligned} \tag{2.6}$$



Obrázek 2.2. Třífázový systém vinutí statoru (abc) a rotoru (ABC)

2.4.2 Sestavení základních rovnic

Pro jednotlivá vinutí statoru a rotoru lze psát celkem šest napěťových rovnic [2–3]

$$\begin{aligned} u_\mathrm{a} &= R_1 i_\mathrm{a} + \frac{\mathrm{d}\Psi_\mathrm{a}}{\mathrm{d}t} & u_\mathrm{A} &= R_2 i_\mathrm{A} + \frac{\mathrm{d}\Psi_\mathrm{A}}{\mathrm{d}t} \\ u_\mathrm{b} &= R_1 i_\mathrm{b} + \frac{\mathrm{d}\Psi_\mathrm{b}}{\mathrm{d}t} & u_\mathrm{B} &= R_2 i_\mathrm{B} + \frac{\mathrm{d}\Psi_\mathrm{B}}{\mathrm{d}t} \\ u_\mathrm{c} &= R_1 i_\mathrm{c} + \frac{\mathrm{d}\Psi_\mathrm{c}}{\mathrm{d}t} & u_\mathrm{C} &= R_2 i_\mathrm{C} + \frac{\mathrm{d}\Psi_\mathrm{C}}{\mathrm{d}t}, \end{aligned} \tag{2.7}$$

kde u_o jsou napětí statorových a rotorových fází, i_o jsou proudy statorových a rotorových fází a Ψ_o jsou výsledné spřažené magnetické toky statorových a rotorových fází, index $o = \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{A}, \mathrm{B}, \mathrm{C}$.

Spřažené magnetické toky v soustavě rovnic (2.7) lze rozepsat pomocí vlastních a vzájemných indukčností statoru a rotoru [4]

$$\begin{aligned} \Psi_\mathrm{a} &= i_\mathrm{a}L_\mathrm{s} + i_\mathrm{b}M_\mathrm{s}\cos(-\tfrac{2\pi}{3}) + i_\mathrm{c}M_\mathrm{s}\cos\tfrac{2\pi}{3} + i_\mathrm{A}M\cos\vartheta + i_\mathrm{B}M\cos(\vartheta-\tfrac{2\pi}{3}) + i_\mathrm{C}M\cos(\vartheta+\tfrac{2\pi}{3}) \\ \Psi_\mathrm{b} &= i_\mathrm{a}M_\mathrm{s}\cos\tfrac{2\pi}{3} + i_\mathrm{b}L_\mathrm{s} + i_\mathrm{c}M_\mathrm{s}\cos(-\tfrac{2\pi}{3}) + i_\mathrm{A}M\cos(\vartheta+\tfrac{2\pi}{3}) + i_\mathrm{B}M\cos\vartheta + i_\mathrm{C}M\cos(\vartheta-\tfrac{2\pi}{3}) \\ \Psi_\mathrm{c} &= i_\mathrm{a}M_\mathrm{s}\cos(-\tfrac{2\pi}{3}) + i_\mathrm{b}M_\mathrm{s}\cos\tfrac{2\pi}{3} + i_\mathrm{c}L_\mathrm{s} + i_\mathrm{A}M\cos(\vartheta-\tfrac{2\pi}{3}) + i_\mathrm{B}M\cos(\vartheta+\tfrac{2\pi}{3}) + i_\mathrm{C}M\cos\vartheta, \end{aligned} \tag{2.8}$$

$$\begin{aligned} \Psi_\mathrm{A} &= i_\mathrm{a}M\cos\vartheta + i_\mathrm{b}M\cos(\vartheta+\tfrac{2\pi}{3}) + i_\mathrm{c}M\cos(\vartheta-\tfrac{2\pi}{3}) + i_\mathrm{A}L_\mathrm{r} + i_\mathrm{B}M_\mathrm{r}\cos(-\tfrac{2\pi}{3}) + i_\mathrm{C}M_\mathrm{r}\cos\tfrac{2\pi}{3} \\ \Psi_\mathrm{B} &= i_\mathrm{a}M\cos(\vartheta-\tfrac{2\pi}{3}) + i_\mathrm{b}M\cos\vartheta + i_\mathrm{c}M\cos(\vartheta+\tfrac{2\pi}{3}) + i_\mathrm{A}M_\mathrm{r}\cos\tfrac{2\pi}{3} + i_\mathrm{B}L_\mathrm{r} + i_\mathrm{C}M_\mathrm{r}\cos(-\tfrac{2\pi}{3}) \\ \Psi_\mathrm{C} &= i_\mathrm{a}M\cos(\vartheta+\tfrac{2\pi}{3}) + i_\mathrm{b}M\cos(\vartheta-\tfrac{2\pi}{3}) + i_\mathrm{c}M\cos\vartheta + i_\mathrm{A}M_\mathrm{r}\cos(-\tfrac{2\pi}{3}) + i_\mathrm{B}M_\mathrm{r}\cos\tfrac{2\pi}{3} + i_\mathrm{C}L_\mathrm{r}. \end{aligned} \tag{2.9}$$

Předpokládejme, že statorové ani rotorové vinutí nemají vyvedený uzel, pak platí, že $i_\mathrm{a} + i_\mathrm{b} + i_\mathrm{c} = 0$ a $i_\mathrm{A} + i_\mathrm{B} + i_\mathrm{C} = 0$. Dále využijeme skutečnosti, že $\cos(2\pi/3) = \cos(-2\pi/3) = -1/2$. Pro spřažený magnetický tok fáze „a“ s ostatními fázemi statoru lze pak psát [4]

$$\Psi_\mathrm{a} = i_\mathrm{a}L_\mathrm{s} + i_\mathrm{b}M_\mathrm{s}\cos(-\tfrac{2\pi}{3}) + i_\mathrm{c}M_\mathrm{s}\cos\tfrac{2\pi}{3} =$$

$$= i_\mathrm{a} L_\mathrm{s} + i_\mathrm{b} M_\mathrm{s}(-\frac{1}{2}) + i_\mathrm{c} M_\mathrm{s}(-\frac{1}{2}) =$$
$$= i_\mathrm{a} L_\mathrm{s} + \frac{1}{2} M_\mathrm{s}(-i_\mathrm{b} - i_\mathrm{c}) = i_\mathrm{a} L_\mathrm{s} + \frac{1}{2} M_\mathrm{s} i_\mathrm{a} = i_\mathrm{a} L_1.$$

Obdobně pak pro další fáze. Soustavu (2.8) lze přepsat na

$$\begin{aligned}
\Psi_\mathrm{a} &= i_\mathrm{a} L_1 + i_\mathrm{A} M \cos\vartheta + i_\mathrm{B} M \cos(\vartheta - \frac{2\pi}{3}) + i_\mathrm{C} M \cos(\vartheta + \frac{2\pi}{3}) \\
\Psi_\mathrm{b} &= i_\mathrm{b} L_1 + i_\mathrm{A} M \cos(\vartheta + \frac{2\pi}{3}) + i_\mathrm{B} M \cos\vartheta + i_\mathrm{C} M \cos(\vartheta - \frac{2\pi}{3}) \\
\Psi_\mathrm{c} &= i_\mathrm{c} L_1 + i_\mathrm{A} M \cos(\vartheta - \frac{2\pi}{3}) + i_\mathrm{B} M \cos(\vartheta + \frac{2\pi}{3}) + i_\mathrm{C} M \cos\vartheta,
\end{aligned} \tag{2.10}$$

kde

$$L_1 = (L_\mathrm{s} + \frac{M_\mathrm{s}}{2}) \tag{2.11}$$

označuje výslednou indukčnost jedné fáze statorového vinutí.[3]

Soustavu (2.9) pak na

$$\begin{aligned}
\Psi_\mathrm{A} &= i_\mathrm{a} M \cos\vartheta + i_\mathrm{b} M \cos(\vartheta + \frac{2\pi}{3}) + i_\mathrm{c} M \cos(\vartheta - \frac{2\pi}{3}) + i_\mathrm{A} L_2 \\
\Psi_\mathrm{B} &= i_\mathrm{a} M \cos(\vartheta - \frac{2\pi}{3}) + i_\mathrm{b} M \cos\vartheta + i_\mathrm{c} M \cos(\vartheta + \frac{2\pi}{3}) + i_\mathrm{B} L_2 \\
\Psi_\mathrm{C} &= i_\mathrm{a} M \cos(\vartheta + \frac{2\pi}{3}) + i_\mathrm{b} M \cos(\vartheta - \frac{2\pi}{3}) + i_\mathrm{c} M \cos\vartheta + i_\mathrm{C} L_2,
\end{aligned} \tag{2.12}$$

kde

$$L_2 = (L_\mathrm{r} + \frac{M_\mathrm{r}}{2}) \tag{2.13}$$

označuje výslednou indukčnost jedné fáze rotorového vinutí.[3]

Uvažujme základní pozici rotoru, tj. $\vartheta = 0$. Potom lze celkový spřažený magnetický tok fáze a vyjádřit jako [2–3]

$$\Psi_\mathrm{a} = i_\mathrm{a} L_1 + i_\mathrm{A} M \cos 0 + i_\mathrm{B} M \cos(-\frac{2\pi}{3}) + i_\mathrm{C} M \cos(\frac{2\pi}{3}) =$$
$$= i_\mathrm{a} L_1 + i_\mathrm{A} M + i_\mathrm{B} M(-\frac{1}{2}) + i_\mathrm{C} M(-\frac{1}{2}) = i_\mathrm{a} L_1 + i_\mathrm{A} M + \frac{1}{2} M(-i_\mathrm{B} - i_\mathrm{C}) =$$
$$= i_\mathrm{a} L_1 + \frac{3}{2} i_\mathrm{A} M = L_1 i_\mathrm{a} + L_\mathrm{m} i_\mathrm{A}.$$

Při této úpravě jsme opět využili skutečnosti, že $i_\mathrm{A} + i_\mathrm{B} + i_\mathrm{C} = 0$ a $\cos(2\pi/3) = \cos(-2\pi/3) = -1/2$. Obdobně lze upravit rovnice pro toky ostatních fází. Výsledkem je soustava šesti rovnic pro celkové spřažené magnetické toky [2–3]

$$\begin{aligned}
\Psi_\mathrm{a} &= L_1 i_\mathrm{a} + L_\mathrm{m} i_\mathrm{A} \qquad & \Psi_\mathrm{A} &= L_2 i_\mathrm{A} + L_\mathrm{m} i_\mathrm{a} \\
\Psi_\mathrm{b} &= L_1 i_\mathrm{b} + L_\mathrm{m} i_\mathrm{B} \qquad & \Psi_\mathrm{B} &= L_2 i_\mathrm{B} + L_\mathrm{m} i_\mathrm{b} \\
\Psi_\mathrm{c} &= L_1 i_\mathrm{c} + L_\mathrm{m} i_\mathrm{C} \qquad & \Psi_\mathrm{C} &= L_2 i_\mathrm{C} + L_\mathrm{m} i_\mathrm{c},
\end{aligned} \tag{2.14}$$

kde

$$L_\mathrm{m} = \frac{3}{2} M \tag{2.15}$$

značí vzájemnou indukčnost statoru a rotoru při uvažování vlivu všech fází.[3]

Pokud se rotor otáčí, mění se i úhel ϑ a tím pádem i vzájemné indukčnosti statoru a rotoru. Obecně je tedy $M = M(\vartheta)$.[2]

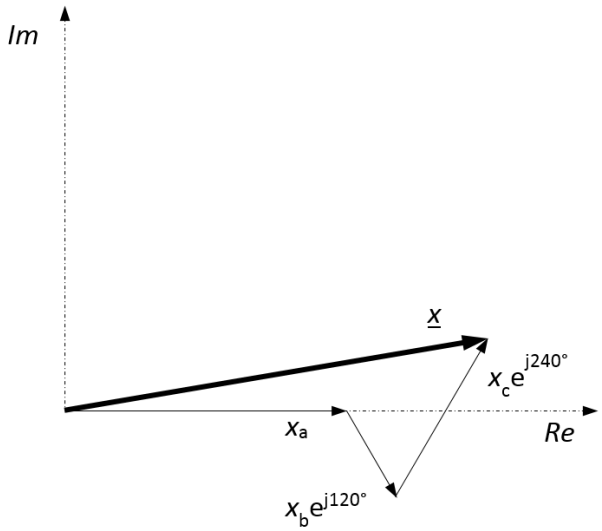
2.4.3 Prostorový vektor

Prostorový vektor je způsob, jakým lze nahradit tři okamžité skalární veličiny jednou veličinou vektorovou, která je reprezentována souřadnicemi v komplexní rovině (obr. 2.3). Pro obecné veličiny x_a, x_b, x_c lze jejich prostorový vektor $\underline{x}$ definovat jako [5–6]

$$\underline{x} = K\ (x_a + x_b\mathbf{a} + x_c\mathbf{a}^2), \tag{2.16}$$

kde

$$\begin{aligned} \mathbf{a} &= \mathrm{e}^{\mathrm{j}\frac{2\pi}{3}} = -\frac{1}{2} + \mathrm{j}\frac{\sqrt{3}}{2} \\ \mathbf{a}^2 &= \mathrm{e}^{-\mathrm{j}\frac{2\pi}{3}} = -\frac{1}{2} - \mathrm{j}\frac{\sqrt{3}}{2}. \end{aligned} \tag{2.17}$$



Obrázek 2.3. Obecný prostorový vektor

Jestliže jsou okamžité třífázové veličiny harmonické o konstantním kmitočtu a tvoří-li symetrický systém, pak má prostorový vektor konstantní amplitudu a rotuje v komplexní rovině konstantní úhlovou rychlostí ω_{SV} [6]

$$\underline{x} = |\underline{x}|\mathrm{e}^{\mathrm{j}\,\omega_{\mathrm{SV}}\,t}, \tag{2.18}$$

kde $|\underline{x}|$ značí velikost (modul) $\underline{x}$.

Rozdělením vektoru $\underline{x}$ do souřadnicových složek ortogonálního systému získáme jeho projekci do reálné osy α a imgainární osy β [6]

$$\underline{x} = \Re\{\underline{x}\} + \mathrm{j}\Im\{\underline{x}\} = x_\alpha + \mathrm{j}x_\beta. \tag{2.19}$$

Pro skalární homopolární složku, pokud je v systému přítomna, platí [5]

$$x_0 = K_0(x_a + x_b + x_c). \tag{2.20}$$

Transformace ze třífázového systému os *abc* do ortogonálního systému $\alpha\beta$ se často nazývá transformace Clarkové. Tuto transformaci lze přehledně vyjádřit pomocí maticového zápisu [6]

$$\begin{pmatrix} x_\alpha \\ x_\beta \\ x_0 \end{pmatrix} = K \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix}. \tag{2.21}$$

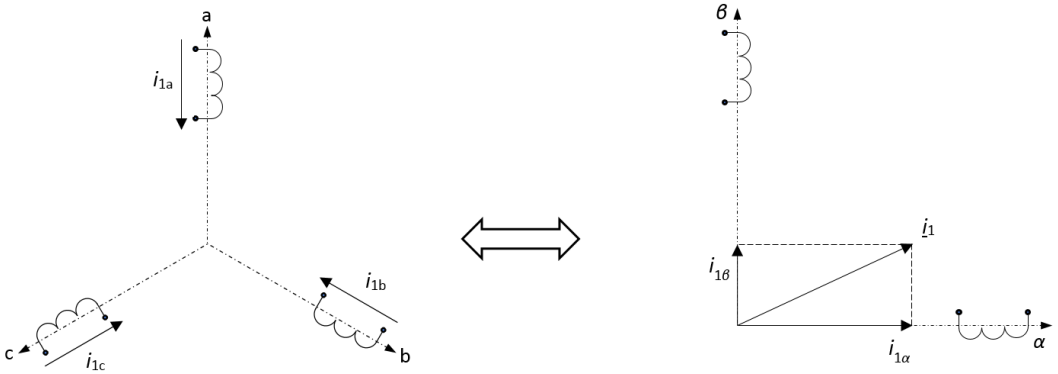
Význam transformačního koeficientu K pro účely transformace veličin v elektrických strojích bude objasněn dále.

Pro inverzní Clarkové transformaci (při $K = \frac{2}{3}$) pak platí [6]

$$\begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{pmatrix} \begin{pmatrix} x_\alpha \\ x_\beta \\ x_0 \end{pmatrix}. \tag{2.22}$$

Vyjádřeme nyní prostorový vektor statorového proudu $\underline{i}_1$ (viz obr. 2.4) ve složkách reálné a imaginární osy ortogonálního systému $\alpha\beta$ (osa fáze „a“ je totožná s osou α) [3, 6]

$$\begin{aligned} i_{1\alpha} &= \Re\{\underline{i}_1\} = K\ (i_{1\mathrm{a}} - \frac{1}{2} i_{1\mathrm{b}} - \frac{1}{2} i_{1\mathrm{c}}) \\ i_{1\beta} &= \Im\{\underline{i}_1\} = \frac{\sqrt{3}}{2} K (i_{1\mathrm{b}} - i_{1\mathrm{c}}). \end{aligned} \tag{2.23}$$



Obrázek 2.4. Transformace statorového proudu ze systému *abc* do ortogonálního systému $\alpha\beta$

Homopolární složka se neuplatní, neboť uvažujeme vinutí s izolovaným uzlem. Transformační koeficient K lze volit. Hodnoty přicházející v úvahu jsou následující: [2–3, 6]

- $K = 1$ – modul transformovného prostorového vektoru bude odpovídat fyzikální skutečnosti (3/2 velikosti amplitudy),
- $K = \sqrt{\frac{2}{3}}$ – bude platit invariantnost výkonů mezi trojfázovým a ortogonálním systémem,
- $K = \frac{2}{3}$ – proud $i_{1\mathrm{a}}$ je roven proudu $i_{1\alpha}$.

Nejčastěji se používá varianta $K = 2/3$ [2–3], která je zvolena i v této práci. Rovnice (2.23) lze pak za předpokladu nevyvedeného uzlu dále upravit na tvar [2, 6]

$$\begin{aligned} i_{1\alpha} &= i_{1\mathrm{a}} \\ i_{1\beta} &= \frac{\sqrt{3}}{3} i_{1\mathrm{a}} + \frac{2\sqrt{3}}{3} i_{1\mathrm{b}} = \frac{1}{\sqrt{3}} i_{1\mathrm{a}} + \frac{2}{\sqrt{3}} i_{1\mathrm{b}}, \end{aligned} \tag{2.24}$$

který je použit v realizační části při softwarové implementace Clarkové transformace.

Clarkové transformaci lze aplikovat i na napětí a spřažené magnetické toky a tím získat vyjádření jejich prostorových vektorů. Soustavu šesti napěťových rovnic (2.7) (tři pro stator a tři pro rotor) tak lze redukovat na dvě vektorové rovnice [2–3]

$$\begin{aligned} \underline{u}_1^1 &= R_1 \underline{i}_1^1 + \frac{\mathrm{d}\underline{\Psi}_1^1}{\mathrm{d}t} \\ \underline{u}_2^2 &= R_2 \underline{i}_2^2 + \frac{\mathrm{d}\underline{\Psi}_2^2}{\mathrm{d}t}. \end{aligned} \tag{2.25}$$

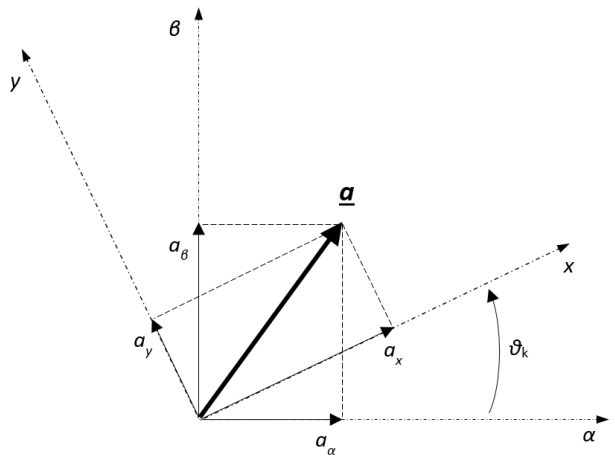
Horní index ^{1} vyjadřuje, že jsou přislušné veličiny svázány se statorem, index ^{2} naopak značí provázanost s rotorem. Uvedené rovnice pro prostorové vektory statorového a rotorového napětí jsou tedy vztaženy k různým souřadnicovým systémům. Pokud se rotor bude otáčet, budou se měnit i vzájemné idnukčnosti mezi statorem a rotorem. Tuto skutečnost lze v tokových vektorových rovnicích respektovat násobením členů se vzájemnou indukčností výrazem $\mathrm{e}^{\mathrm{j}\vartheta}$. Rovnice prostorových vektorů statorového a rotorového spřaženého magnetického toku lze potom psát ve tvaru [2–3]

$$\begin{aligned} \underline{\Psi}_1^1 &= L_1 \underline{i}_1^1 + L_\mathrm{m} \underline{i}_2^2 \mathrm{e}^{\mathrm{j}\vartheta} \\ \underline{\Psi}_2^2 &= L_2 \underline{i}_2^2 + L_\mathrm{m} \underline{i}_1^1 \mathrm{e}^{-\mathrm{j}\vartheta}. \end{aligned} \tag{2.26}$$

V rovnici u rotorového toku je v argumentu exponenciály znaménko minus, neboť z pohledu rotoru se stator otáčí opačným směrem. V následující kapitole bude ukázán způsob, jakým lze rovnice (2.25) a (2.26) transformovat do společného souřadnicového systému a odstranit tak periodickou závislost vzájemné indukčnosti.

2.4.4 Transformace souřadnic

Obecný prostorový vektor $\underline{a}$ lze transformovat ze systému $\alpha\beta$ do obecného systému xy, který vůči předchozímu rotuje úhlovou rychlostí ω_k. Okamžitý úhel ϑ_k mezi těmito systémy lze vyjádřit jako $\vartheta_\mathrm{k} = \omega_\mathrm{k} t + \vartheta_{\mathrm{k}0}$, kde $\vartheta_{\mathrm{k}0}$ je počáteční úhel v čase $t = 0$.[2] Situace je znázorněna na obr. 2.5.



Obrázek 2.5. Transformace ze systému $\alpha\beta$ do obecného rotujícího systému xy

Pro složky a_x, a_y prostorového vektoru $\underline{a}$ v novém souřadnicovém systému platí [2]

$$\begin{aligned} a_x &= a_\alpha \cos\vartheta_\mathrm{k} + a_\beta \sin\vartheta_\mathrm{k} \\ a_y &= -a_\alpha \sin\vartheta_\mathrm{k} + a_\beta \cos\vartheta_\mathrm{k}. \end{aligned} \tag{2.27}$$

Transformaci lze zapsat i vektorově. V komplexní rovině odpovídá natočení vektoru o úhel ϑ_k násobení výrazem $\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}}$ [2–3]

$$\underline{a}_\mathrm{k} = \underline{a}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}}. \tag{2.28}$$

Pro zpětnou transformaci lze psát [2]

$$\begin{aligned} a_\alpha &= a_x \cos\vartheta_\mathrm{k} - a_y \sin\vartheta_\mathrm{k} \\ a_\beta &= a_x \sin\vartheta_\mathrm{k} + a_y \cos\vartheta_\mathrm{k}. \end{aligned} \tag{2.29}$$

A ve vektorovém zápisu pak [2–3]

$$\underline{a} = \underline{a}_\mathrm{k}\mathrm{e}^{-\mathrm{j}\vartheta_\mathrm{k}}. \tag{2.30}$$

Uvedená transformace se někdy nazývá jako Parkova transformace. Výše uvedené vztahy lze přehledně shrnout v maticovém zápisu. Transformace ze systému $\alpha\beta$ do obecného xy [6]

$$\begin{pmatrix} a_x \\ a_y \\ a_0 \end{pmatrix} = \begin{pmatrix} \cos\vartheta_\mathrm{k} & \sin\vartheta_\mathrm{k} & 0 \\ -\sin\vartheta_\mathrm{k} & \cos\vartheta_\mathrm{k} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_\alpha \\ a_\beta \\ a_0 \end{pmatrix}. \tag{2.31}$$

Zpětná transformace [6]

$$\begin{pmatrix} a_\alpha \\ a_\beta \\ a_0 \end{pmatrix} = \begin{pmatrix} \cos\vartheta_\mathrm{k} & -\sin\vartheta_\mathrm{k} & 0 \\ \sin\vartheta_\mathrm{k} & \cos\vartheta_\mathrm{k} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_x \\ a_y \\ a_0 \end{pmatrix}. \tag{2.32}$$

Uvedené transformační vztahy je možno použít pro transformaci napěťových a tokových vektorových rovnic (2.25) a (2.26) do společného souřadnicového systému rotujícího rychlostí ω_k vůči statoru a $\omega_\mathrm{k} - \omega$ vůči rotoru. Tomu odpovídá násobení statorových rovnic členem $\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}}$ a rotorových členem $\mathrm{e}^{\mathrm{j}(\vartheta_\mathrm{k}-\vartheta)}$ [2–3]

$$\begin{aligned} \underline{u}_1^\mathrm{k}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}} &= R_1\underline{i}_1^\mathrm{k}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}} + \frac{\mathrm{d}(\underline{\Psi}_1^\mathrm{k}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}})}{\mathrm{d}t} \\ \underline{u}_2^\mathrm{k}\mathrm{e}^{\mathrm{j}(\vartheta_\mathrm{k}-\vartheta)} &= R_2\underline{i}_2^\mathrm{k}\mathrm{e}^{\mathrm{j}(\vartheta_\mathrm{k}-\vartheta)} + \frac{\mathrm{d}(\underline{\Psi}_2^\mathrm{k}\mathrm{e}^{\mathrm{j}(\vartheta_\mathrm{k}-\vartheta)})}{\mathrm{d}t}. \end{aligned} \tag{2.33}$$

Člen s derivací u první rovnice lze rozepsat pomocí pravidla o derivaci složené funkce

$$\frac{\mathrm{d}(\underline{\Psi}_1^\mathrm{k}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}})}{\mathrm{d}t} = \frac{\mathrm{d}\underline{\Psi}_1^\mathrm{k}}{\mathrm{d}t}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}} + \frac{\mathrm{d}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}}}{\mathrm{d}t}\underline{\Psi}_1^\mathrm{k} = \frac{\mathrm{d}\underline{\Psi}_1^\mathrm{k}}{\mathrm{d}t}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}} + \mathrm{j}\omega_\mathrm{k}\underline{\Psi}_1^\mathrm{k}\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}}. \tag{2.34}$$

Obdobnou úpravu lze provést i pro druhou z rovnic. Po zpětném dosazení a vydělení první rovnice výrazem $\mathrm{e}^{\mathrm{j}\vartheta_\mathrm{k}}$ a druhé $\mathrm{e}^{\mathrm{j}(\vartheta_\mathrm{k}-\vartheta)}$ dostáváme výslednou soustavu vektorových napěťových rovnic pro stator ve společném souřadnicovém systému [2–3]

$$\begin{aligned} \underline{u}_1^\mathrm{k} &= R_1\underline{i}_1^k + \frac{\mathrm{d}\underline{\Psi}_1^\mathrm{k}}{\mathrm{d}t} + \mathrm{j}\omega_\mathrm{k}\underline{\Psi}_1^\mathrm{k} \\ \underline{u}_2^\mathrm{k} &= R_2\underline{i}_2^\mathrm{k} + \frac{\mathrm{d}\underline{\Psi}_2^\mathrm{k}}{\mathrm{d}t} + \mathrm{j}(\omega_\mathrm{k} - \omega)\underline{\Psi}_2^\mathrm{k}. \end{aligned} \tag{2.35}$$

Jestliže jsou statorové a rotorové ortogonální systémy souřadnic transformované do společného systému, pak se jejich relativní poloha s otáčením rotoru nemění, nemění se ani vzájemné indukčnosti M statoru a rotoru ($\vartheta = 0$) a rovnice pro toky (2.26) lze upravit do tvaru [2–3]

$$\begin{aligned} \underline{\Psi}_1^\mathrm{k} &= L_1\underline{i}_1^\mathrm{k} + L_\mathrm{m}\underline{i}_2^\mathrm{k} \\ \underline{\Psi}_2^\mathrm{k} &= L_2\underline{i}_2^\mathrm{k} + L_\mathrm{m}\underline{i}_1^\mathrm{k}. \end{aligned} \tag{2.36}$$

V rovnicích (2.35) nazýváme napětí indukovaná časovou změnou magnetického toku jako transformační napětí a napětí indukovaná vlivem otáčení vinutí v magnetickém poli jako rotační napětí.[2]

2.4.5 Rovnice asynchronního stroje v různých souřadnicových soustavách

Rovnice matematického modelu (2.35) a (2.36) lze rozepsat do složek v obecném souřadnicovém systému xy [2]

$$\begin{aligned}
u_{1x} &= R_1 i_{1x} + \frac{\mathrm{d}\Psi_{1x}}{\mathrm{d}t} - \omega_\mathrm{k}\Psi_{1y} \\
u_{1y} &= R_1 i_{1y} + \frac{\mathrm{d}\Psi_{1y}}{\mathrm{d}t} + \omega_\mathrm{k}\Psi_{1x} \\
u_{2x} &= R_2 i_{2x} + \frac{\mathrm{d}\Psi_{2x}}{\mathrm{d}t} - (\omega_\mathrm{k} - \omega)\Psi_{2y} \\
u_{2y} &= R_2 i_{2y} + \frac{\mathrm{d}\Psi_{2y}}{\mathrm{d}t} + (\omega_\mathrm{k} - \omega)\Psi_{2x} \\
\Psi_{1x} &= L_1 i_{1x} + L_\mathrm{m} i_{2x} \\
\Psi_{1y} &= L_1 i_{1y} + L_\mathrm{m} i_{2y} \\
\Psi_{2x} &= L_2 i_{2x} + L_\mathrm{m} i_{1x} \\
\Psi_{2y} &= L_2 i_{2y} + L_\mathrm{m} i_{1y}.
\end{aligned} \tag{2.37}$$

Pro speciální volbu ω_k pak dostáváme modifikaci soustavy (2.37) v různých souřadnicových systémech. V elektrických pohonech mají význam zejména následující dvě souřadnicové soustavy:[2–3]

- **Souřadnicový systém svázaný se statorem**. Značení $\alpha\beta$. Platí, že $\omega_\mathrm{k} = 0$:

$$\begin{aligned}
u_{1\alpha} &= R_1 i_{1\alpha} + \frac{\mathrm{d}\Psi_{1\alpha}}{\mathrm{d}t} \\
u_{1\beta} &= R_1 i_{1\beta} + \frac{\mathrm{d}\Psi_{1\beta}}{\mathrm{d}t} \\
u_{2\alpha} &= R_2 i_{2\alpha} + \frac{\mathrm{d}\Psi_{2\alpha}}{\mathrm{d}t} + \omega\Psi_{2\beta} \\
u_{2\beta} &= R_2 i_{2\beta} + \frac{\mathrm{d}\Psi_{2\beta}}{\mathrm{d}t} - \omega\Psi_{2\alpha} \\
\Psi_{1\alpha} &= L_1 i_{1\alpha} + L_\mathrm{m} i_{2\alpha} \\
\Psi_{1\beta} &= L_1 i_{1\beta} + L_\mathrm{m} i_{2\beta} \\
\Psi_{2\alpha} &= L_2 i_{2\alpha} + L_\mathrm{m} i_{1\alpha} \\
\Psi_{2\beta} &= L_2 i_{2\beta} + L_\mathrm{m} i_{1\beta}.
\end{aligned} \tag{2.38}$$

- **Souřadnicový systém rotující synchronní rychlostí**. Značení dq. Platí, že $\omega_\mathrm{k} = \omega_\mathrm{s}$:

$$\begin{aligned}
u_{1d} &= R_1 i_{1d} + \frac{\mathrm{d}\Psi_{1d}}{\mathrm{d}t} - \omega_\mathrm{s}\Psi_{1q} \\
u_{1q} &= R_1 i_{1q} + \frac{\mathrm{d}\Psi_{1q}}{\mathrm{d}t} + \omega_\mathrm{s}\Psi_{1d} \\
u_{2d} &= R_2 i_{2d} + \frac{\mathrm{d}\Psi_{2d}}{\mathrm{d}t} - (\omega_\mathrm{s} - \omega)\Psi_{2q} \\
u_{2q} &= R_2 i_{2q} + \frac{\mathrm{d}\Psi_{2\mathrm{q}}}{\mathrm{d}t} + (\omega_\mathrm{s} - \omega)\Psi_{2d} \\
\Psi_{1d} &= L_1 i_{1d} + L_\mathrm{m} i_{2d} \\
\Psi_{1q} &= L_1 i_{1q} + L_\mathrm{m} i_{2q} \\
\Psi_{2d} &= L_2 i_{2d} + L_\mathrm{m} i_{1d} \\
\Psi_{2q} &= L_2 i_{2q} + L_\mathrm{m} i_{1q}.
\end{aligned} \tag{2.39}$$

2.5 Výkon a moment

Elektrický výkon přivedený na svorky statoru asynchronního motoru s kotvou nakrátko se dělí na ztráty v železe, výkon zmařený na odporu statorového vinutí (Jouleovy ztráty) a na výkon přenášený vzduchovou mezerou P_δ, jehož s-tá část odpovídá ztrátám v rotorovém obvodu a $(1-s)$-tá část pak elektromechanickému výkonu, který se rovná součtu výkonu na hřídeli a mechanických ztrát.[1]

Činný výkon odebíraný asynchronním motorem lze vyjádřit jako reálnou část součinu statorového vektoru napětí a komplexně sdruženého statorového vektoru proudu [2–3]

$$P = A\ \Re\{\underline{u}_1 \underline{i}_1^*\}. \tag{2.40}$$

Po dosazení za $\underline{u}_1$ z (2.25) a využití skutečnosti, že $\underline{i}_1 = i_{1\alpha} + \mathrm{j}i_{1\beta}$, $\underline{i}_1^* = i_{1\alpha} - \mathrm{j}i_{1\beta}$ a $\underline{\Psi}_1 = \Psi_{1\alpha} + \mathrm{j}\Psi_{1\beta}$ lze vztah (2.40) dále rozepsat [2]

$$\begin{aligned} P = A\ \Re\{R_1(i_{1\alpha} + \mathrm{j}i_{1\beta})(i_{1\alpha} - \mathrm{j}i_{1\beta}) + \mathrm{j}\omega(\Psi_{1\alpha} + \mathrm{j}\Psi_{1\beta})(i_{1\alpha} - \mathrm{j}i_{1\beta})\} = \\ = A\ [R_1(i_{1\alpha}^2 + i_{1\beta}^2) + \omega(\Psi_{1\alpha} i_{1\beta} - \Psi_{1\beta} i_{1\alpha})]. \end{aligned}$$

Operátor časové derivace byl nahrazen násobením výrazem $\mathrm{j}\omega$. Výkon odebíraný ze zdroje lze tedy vyjádřit jako [2–3]

$$P = A[R_1(i_{1\alpha}^2 + i_{1\beta}^2) + \omega(\Psi_{1\alpha} i_{1\beta} - \Psi_{1\beta} i_{1\alpha})]. \tag{2.41}$$

První člen v závorce v rovnici (2.41) odpovídá ztrátám ve vinutí a druhý výkonu ve vzduchové mezeře stroje P_δ. [2–3]

Jelikož při Clarkové transformaci bylo zvoleno $K = \frac{2}{3}$, tak platí, že [2–3]

$$i_{1\alpha}^2 + i_{1\beta}^2 = I_{1\mathrm{m}}^2 = (\sqrt{2} I_1)^2, \tag{2.42}$$

kde I_{1m} značí amplitudu statorového fázového proudu a I_1 pak jeho efektivní hodnotu.

Konstantu A lze pak získat porovnáním (2.42) se členem vyjadřujícím Jouleovy ztráty ve vinutí v rovnici (2.41) [2–3]

$$3R_1 I_1^2 = A\ R_1(\sqrt{2} I_1)^2 \Rightarrow A = \frac{3}{2}.$$

S využitím výše uvedeného lze pak psát výsledný vztah pro vnitřní elektromechanický moment m_i [2–3]

$$m_\mathrm{i} = \frac{P_\delta(1-s)}{\Omega} = \frac{3}{2} p_\mathrm{p}(\Psi_{1\alpha} i_{1\beta} - \Psi_{1\beta} i_{1\alpha}), \tag{2.43}$$

kde bylo využito skutečnosti, že [2–3]

$$\Omega = \frac{\omega}{p_\mathrm{p}}(1-s). \tag{2.44}$$

Dá se odvodit, že moment asynchronního motoru lze vyjádřit pomocí různých kombinací prostorových vektorů $\underline{V}, \underline{W}$ statorových a rotorových veličin [3, 6]

$$m_\mathrm{i} = k_1 k_2\, p_\mathrm{p} |\underline{V} \times \underline{W}|, \tag{2.45}$$

kde $\times$ značí vektorový součin a k_1 je konstanta závislá na volbě koeficientu K u Clarkové transformace [5–6]

$$k_1 = \frac{2}{3K^2}. \tag{2.46}$$

Koeficient k_2 je multiplikační koeficient, jehož hodnota závisí na kombinaci zvolených vektorů dle tab. 2.1.

Podrobný matematický rozbor výkonu a momentu asynchronního stroje lze nalézt např. v [5].

varianta	1	2	3	4	5	6	7	8
$\underline{V}$	$\underline{i}_1$	$\underline{i}_1$	$\underline{i}_1$	$\underline{i}_1$	$\underline{i}_2$	$\underline{i}_2$	$\underline{i}_2$	$\underline{i}_2$
$\underline{W}$	$\underline{i}_2$	$\underline{\Psi}_1$	$\underline{\Psi}_\mu$	$\underline{\Psi}_2$	$\underline{\Psi}_1$	$\underline{\Psi}_\mu$	$\underline{\Psi}_2$	$\underline{\Psi}_2$
k_2	L_m	–	–	L_m/L_2	L_m/L_1	-	-	$L_\mathrm{m}/\sigma L_1 L_2$

Tabulka 2.1. Koeficienty pro výpočet momentu

2.6 Náhradní schéma asynchronního motoru pro harmonické napájení v ustáleném stavu

Známé náhradní schéma asynchronního motoru ve tvaru T-článku pro ustálené stavy lze odvodit ze soustavy vektorových rovnic v systému $\alpha\beta$ svázaného se statorem, které lze pro ustálené harmonické průběhy upravit do tvaru (není explicitně indexem zdůrazňován vztažný souřadnicový systém) [2–3]

$$\begin{aligned} \underline{U}_1 &= R_1\underline{I}_1 + \mathrm{j}X_1\underline{I}_1 + \mathrm{j}X_{1\mathrm{m}}\underline{I}_2 \\ \underline{U}_2 &= R_2\underline{I}_2 + \mathrm{j}X_2\underline{I}_2 + \mathrm{j}s\ X_{2\mathrm{m}}\underline{I}_1, \end{aligned} \tag{2.47}$$

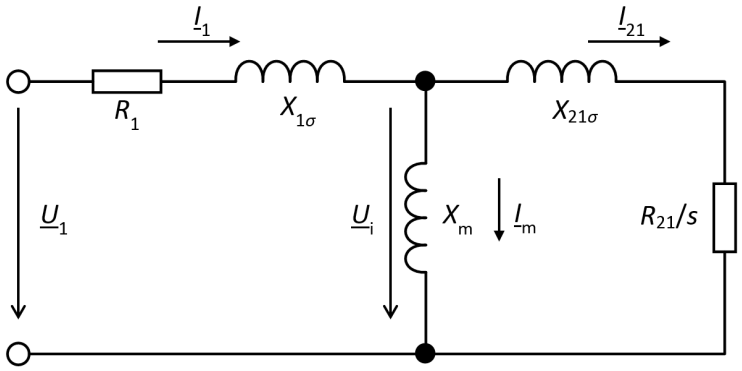
kde X s příslušným indexem značí reaktance (např. $X_1 = \omega_\mathrm{s}L_1$) a $s = (\omega_\mathrm{s} - \omega)/\omega_\mathrm{s}$ je již dříve zavedený skluz. Pro zdůraznění, že uvažujeme harmonický ustálený stav, budou vektory napětí a proudů značeny velkým písmenem.

Po zavedení rozptylových indukčností, resp. reaktancí (značeno dolním indexem σ, např. $X_{1\sigma} = X_1 - X_{1m}$), přepočítání rotorových veličin na stator (značení druhým dolním indexem $_1$ u rotorových veličin, pravidla pro přepočet lze nalézt např. v [1]) a uvažování motoru s kotvou nakrátko (tj. $\underline{U}_2 = 0$) dostáváme [2–3]

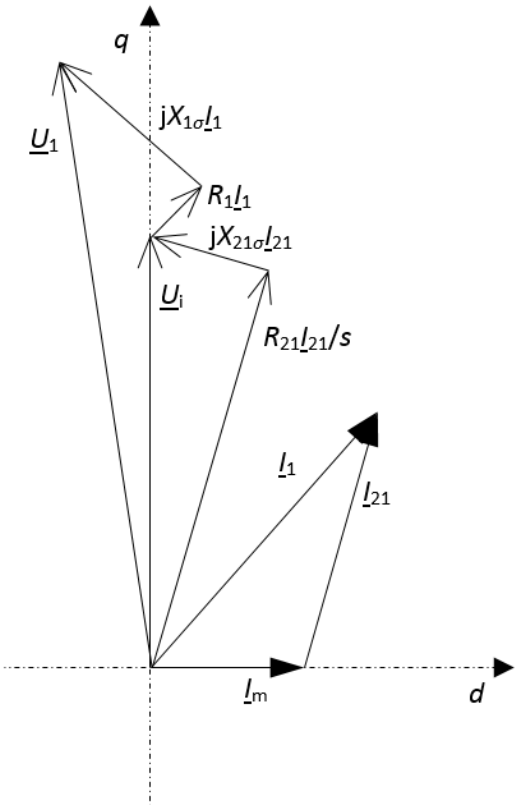
$$\begin{aligned} \underline{U}_1 &= R_1\underline{I}_1 + \mathrm{j}X_{1\sigma}\underline{I}_1 + \mathrm{j}X_\mathrm{m}(\underline{I}_1 - \underline{I}_{21}) \\ \underline{U}_{21} &= R_{21}\underline{I}_{21} + \mathrm{j}s\ X_{21\sigma}\underline{I}_{21} - \mathrm{j}s\ X_\mathrm{m}(\underline{I}_1 - \underline{I}_{21}). \end{aligned} \tag{2.48}$$

Druhou rovnici vydělíme skluzem a dále zavedením indukovaného napětí $\underline{U}_\mathrm{i} = X_\mathrm{m}(\underline{I}_1 - \underline{I}_{21})$ a označením $\underline{I}_\mathrm{m} = \underline{I}_1 - \underline{I}_{21}$ za tzv. magnetizační proud, získáváme výsledné rovnice popisující náhradní schéma ve tvaru T-článku, které je pro motor s kotvou nakrátko (tj. $\underline{U}_{21} = 0$) znázorněno na obr. 2.6. Fázorový diagram k tomuto schématu je pak zakreslen na obr. 2.7.[2–3]

$$\begin{aligned} \underline{U}_1 &= (R_1 + \mathrm{j}X_{1\sigma})\underline{I}_1 + \underline{U}_\mathrm{i} \\ \underline{U}_\mathrm{i} &= (\frac{R_{21}}{s} + \mathrm{j}X_{21\sigma})\underline{I}_{21} + \frac{\underline{U}_{21}}{s}. \end{aligned} \tag{2.49}$$



Obrázek 2.6. Náhradní schéma asynchronního motoru ve tvaru T-článku pro ustálené harmonické napájení



Obrázek 2.7. Fázorový diagram k náhradnímu schématu ve tvaru T-článku

Kapitola 3
Vektorové řízení pohonů s asynchronními motory

Tato část práce pojednává o principu vektorově orientované regulace pohonů s asynchronními motory. Jsou zde také uvedeny nejvyužívanější typy modelů asynchronních motorů při vektorovém řízení. Na závěr je pak představeno navržené regulační schéma, které je realizováno v praktické části a dále pak ještě řízení použitého napájecího měniče.

3.1 Matematické modely asynchronního motoru používané při vektorovém řízení

Mezi nejpoužívanější modely ve vektorové regulaci patří proudový I-n model a napěťový U-I model.[3] Tyto modely lze odvodit ze soustav rovnic (2.35) a (2.36), které lze za předpokladu uvažování asynchronního motoru s kotvou nakrátko (tj. $\underline{u}_2 = 0$) přepsat do tvaru

$$\begin{aligned} \underline{u}_1 &= R_1\underline{i}_1 + \frac{\mathrm{d}\underline{\Psi}_1}{\mathrm{d}t} + \mathrm{j}\omega_\mathrm{k}\underline{\Psi}_1 \\ 0 &= R_2\underline{i}_2 + \frac{\mathrm{d}\underline{\Psi}_2}{\mathrm{d}t} + \mathrm{j}(\omega_\mathrm{k} - \omega)\underline{\Psi}_2 \\ \underline{\Psi}_1 &= L_1\underline{i}_1 + L_\mathrm{m}\underline{i}_2 \\ \underline{\Psi}_2 &= L_2\underline{i}_2 + L_\mathrm{m}\underline{i}_1. \end{aligned} \tag{3.1}$$

3.1.1 Proudový I-n model

Vstupem do I-n modelu je prostorový vektor statorového proudu, resp. jeho reálná a imaginární část a elektrická úhlová rychlost rotoru. Výstupem je pak prostorový vektor rotorového spřaženého magnetického toku, resp. jeho reálná a imaginární část.

Z poslední rovnice soustavy (3.1) vyjádříme (neměřitelný) rotorový proud

$$\underline{i}_2 = \frac{\underline{\Psi}_2 - L_\mathrm{m}\underline{i}_1}{L_2}, \tag{3.2}$$

který dosadíme do rovnice pro rotorové napětí

$$0 = R_2\frac{\underline{\Psi}_2 - L_\mathrm{m}\underline{i}_1}{L_2} + \frac{\mathrm{d}\underline{\Psi}_2}{\mathrm{d}t} + \mathrm{j}(\omega_\mathrm{k} - \omega)\underline{\Psi}_2. \tag{3.3}$$

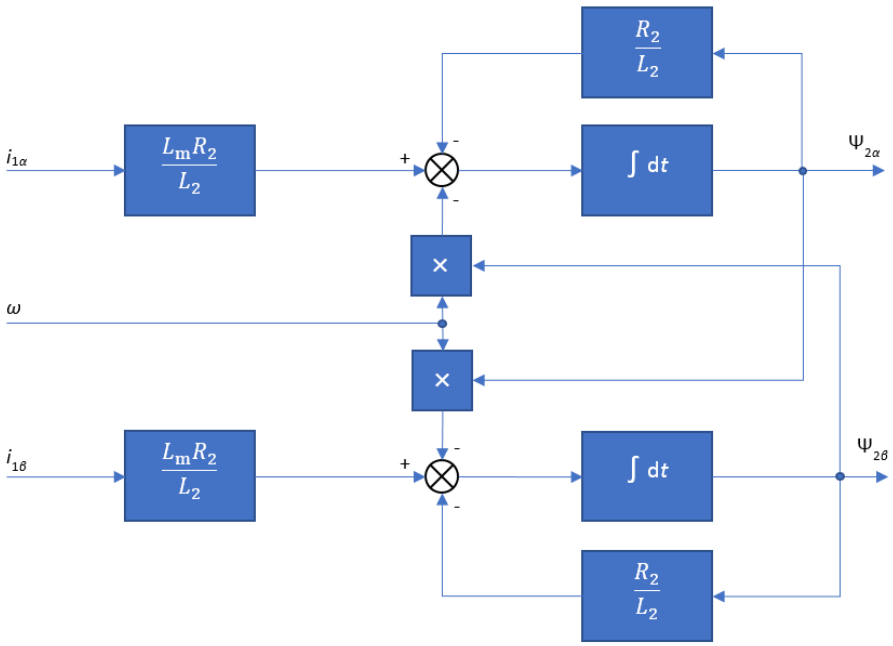
Po osamostatnění členu s derivací získáváme proudový I-n model ve vektorovém tvaru v obecném souřadnicovém systému [3, 5]

$$\frac{\mathrm{d}\underline{\Psi}_2}{\mathrm{d}t} = \frac{L_m R_2}{L_2}\underline{i}_1 - [\frac{R_2}{L_2} + \mathrm{j}(\omega_\mathrm{k} - \omega)]\underline{\Psi}_2. \tag{3.4}$$

Vyjádřeme nyní vektorový I-n model ve složkách souřadnicového systému $\alpha\beta$ svázaného se statorem, kde platí, že $\omega_\mathrm{k} = 0$. Separací rovnice (3.4) na reálnou a imaginární část dostáváme [3, 5]

$$\begin{aligned} \frac{\mathrm{d}\Psi_{2\alpha}}{\mathrm{d}t} &= \frac{L_\mathrm{m}R_2}{L_2} i_{1\alpha} - \frac{R_2}{L_2}\Psi_{2\alpha} - \omega\Psi_{2\beta} \\ \frac{\mathrm{d}\Psi_{2\beta}}{\mathrm{d}t} &= \frac{L_\mathrm{m}R_2}{L_2} i_{1\beta} - \frac{R_2}{L_2}\Psi_{2\beta} + \omega\Psi_{2\alpha}. \end{aligned} \tag{3.5}$$

Na základě soustavy rovnic (3.5) pak můžeme kreslit blokové schéma proudového modelu ve složkách souřadnicového systému $\alpha\beta$ (viz obr. 3.1).



Obrázek 3.1. Blokové schéma I-n modelu asynchronního motoru v souřadnicích $\alpha\beta$

Pro I-n model ve složkách souřadnicového systému dq svázaného s rotorovým tokem, tj. $\omega_\mathrm{k} = \omega_\mathrm{s}$ lze za předpokladu položení osy d do směru prostorového vektoru rotorového spřaženého magnetického toku $\underline{\Psi}_2$ obdobným způsobem odvodit [3]

$$\begin{aligned} \frac{\mathrm{d}\Psi_{2d}}{\mathrm{d}t} &= \frac{L_\mathrm{m}R_2}{L_2} i_{1d} - \frac{R_2}{L_2}\Psi_{2d} \\ \Psi_{2d}\omega_\mathrm{slip} &= \frac{L_\mathrm{m}R_2}{L_2} i_{1q}. \end{aligned} \tag{3.6}$$

I-n model vykazuje dobrou stabilitu v oblasti nízkých otáček. Jeho nevýhodou je potřeba instalace otáčkového čidla, což zvyšuje výslednou cenu pohonu.[3, 6]

3.1.2 Napěťový U-I model

Vstupními veličinami do U-I modelu jsou vektory statorového napětí a statorového proudu, resp. jejich reálné a imaginární části. Výstupem je pak vektor rotorového toku, resp. jeho reálná a imaginární část.

Upravíme rovnice (3.1) pro rotorové toky vyloučením vektoru rotorového proudu $\underline{i}_2$ [3, 5]

$$\underline{\Psi}_1 = L_1\underline{i}_1 + L_\mathrm{m}\frac{\underline{\Psi}_2 - L_\mathrm{m}\underline{i}_1}{L_2} = \sigma L_1\underline{i}_1 + \frac{L_\mathrm{m}}{L_2}\underline{\Psi}_2, \tag{3.7}$$

kde

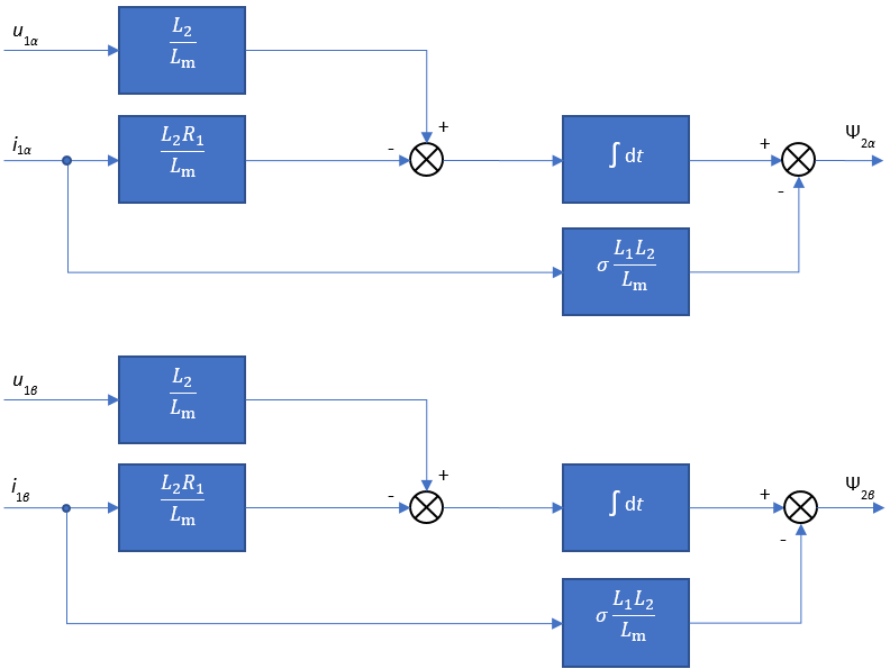
$$\sigma = 1 - \frac{L_{\mathrm{m}}^2}{L_1 L_2} \tag{3.8}$$

je tzv. rozptyl.

Nejčastěji se U-I model používá v souřadnicích $\alpha\beta$.[3] Získaný vztah dosadíme do rovnice (3.1) pro vektor statorového napětí $\underline{u}_1$. Po několika úpravách a s uvažováním $\omega_{\mathrm{k}} = 0$ dostáváme rovnice U-I modelu ve složkách $\alpha\beta$ souřadnic svázaných se statorem [3, 5]

$$\begin{aligned} \frac{\mathrm{d}\Psi_{2\alpha}}{\mathrm{d}t} &= \frac{L_2}{L_{\mathrm{m}}}(u_{1\alpha} - R_1 i_{1\alpha} - \sigma L_1 \frac{\mathrm{d}i_{1\alpha}}{\mathrm{d}t}) \\ \frac{\mathrm{d}\Psi_{2\beta}}{\mathrm{d}t} &= \frac{L_2}{L_{\mathrm{m}}}(u_{1\beta} - R_1 i_{1\beta} - \sigma L_1 \frac{\mathrm{d}i_{1\beta}}{\mathrm{d}t}). \end{aligned} \tag{3.9}$$

Blokové schéma U-I modelu v souřadnicích $\alpha\beta$ je na obr. 3.2. Model U-I dává dobré výsledky zejména při vyšších otáčivých rychlostech a je jednou z možností pro bezsenzorové zjišťování otáček.[3]



Obrázek 3.2. Blokové schéma U-I modelu asynchronního motoru v souřadnicích $\alpha\beta$

Literatura [5] udává napěťový modelu poněkud v odlišném tvaru, který vychází z vektorové rovnice pro statorové napětí

$$\begin{aligned} \frac{\mathrm{d}\Psi_{1\alpha}}{\mathrm{d}t} &= u_{1\alpha} - R_1 i_{1\alpha} - \gamma \Psi_{1\alpha} \\ \frac{\mathrm{d}\Psi_{1\beta}}{\mathrm{d}t} &= u_{1\beta} - R_1 i_{1\beta} - \gamma \Psi_{1\beta} \\ \Psi_{2\alpha} &= \frac{\Psi_{1\alpha} - i_{1\alpha}(L_{1\sigma} + L_{2\sigma}\frac{L_{\mathrm{m}}}{L_2})}{1 - \frac{L_{2\sigma}}{L_2}} \\ \Psi_{2\beta} &= \frac{\Psi_{1\beta} - i_{1\beta}(L_{1\sigma} + L_{2\sigma}\frac{L_{\mathrm{m}}}{L_2})}{1 - \frac{L_{2\sigma}}{L_2}}. \end{aligned} \tag{3.10}$$

Napěťový model vyhodnocuje složky toku správně pouze při zcela přesně zadaných počátečních podmínkách, což je v praxi samozřejmě nerealizovatelné. Proto je v rovnicích (3.10) zaveden koeficient γ, kterým je tlumena nežádoucí stejnosměrná složka (zaniká s časovou konstantou $1/\gamma$). Toto opatření ale vnáší do výpočtu toku určitou chybu, která se projeví ne zcela přesným vyhodnocením polohy vektoru $\underline{\Psi}_1$.[5]

3.2 Vektorové řízení orientované na rotorový tok

Princip vektorového řízení spočívá v rozkladu statorového proudu na tokotvornou a momentotvornou složku, které jsou řízeny odděleně. Struktura a způsob řízení asynchronního motoru jsou potom formálně shodné s těmi u cize buzeného stejnosměrného motoru se všemi výhodami z toho plynoucími. Toho bude docíleno za podmínek, které zároveň vektorové řízení charakterizují:[7]

- řízení magnetického toku a momentu musí být odděleno,
- při konstantním magnetickém toku by měl být moment přímo úměrný momentotvorné složce proudu,
- v ustáleném stavu by regulované veličiny měly být stejnosměrné, resp. kvazistejnosměrné,
- moment by měl být úměrný skluzu.

Požadavek stejnosměrných, resp. kvazistejnosměrných veličin lze zajistit použitím ortogonální vztažné soustavy rotující synchronní elektrickou úhlovou rychlostí ω_s. Lineární závislost mezi momentotvornou složkou proudu a momentem stroje je splněna položením osy d vztažného souřadnicového systému do směru vektoru toku. Zbývající požadavky lze splnit orientováním vektorového řízení na rotorový magnetický tok ($|\underline{\Psi}_2| = \Psi_{2d}$, $\Psi_{2q} = 0$), které se ukazuje jako nejvýhodnější.[5, 7] V ustáleném stavu je pak možno považovat proud i_{1d} za magnetizační a navíc platí, že vektor rotorového proudu je kolmý na vektor rotorového toku, z čehož plyne, že $i_{2d} = 0$. Z toho pak dále [5]

$$\Psi_2 = L_\mathrm{m} i_{1d}. \tag{3.11}$$

Vyjádřeme vztah pro moment asynchronního stroje v souřadnicích dq svázaných s prostorovým vektorem rotorového spřaženého magnetického toku $\underline{\Psi}_2$ pomocí složek vektorů statorového proudu i_{1d}, i_{1q} a rotorového toku Ψ_{2d}, Ψ_{2q} (viz. kap. 2.5)

$$m_\mathrm{i} = \frac{3}{2} p_\mathrm{p} \frac{L_\mathrm{m}}{L_2} (\Psi_{2d} i_{1q} - \Psi_{2q} i_{1d}). \tag{3.12}$$

Jelikož platí $\Psi_{2q} = 0$, lze vztah (3.12) dále zjednodušit na

$$m_\mathrm{i} = \frac{3}{2} p_\mathrm{p} \frac{L_\mathrm{m}}{L_2} \Psi_2 i_{1q}. \tag{3.13}$$

Vyjádření pro Ψ_2 lze získat z první rovnice soustavy (3.6)

$$\frac{\mathrm{d}\Psi_2}{\mathrm{d}t} = \frac{L_\mathrm{m} R_2}{L_2} i_{1d} - \frac{R_2}{L_2} \Psi_2, \tag{3.14}$$

který transformujeme pomocí Laplaceovy (3.14) transformace (p značí Laplaceův operátor)

$$\Psi_2(p) = \frac{L_\mathrm{m}}{1 + p\tau_\mathrm{r}} i_{1d}(p), \tag{3.15}$$

kde $\tau_\mathrm{r} = L_2/R_2$ je rotorová časová konstanta.

Z rovnic (3.13) a (3.15) vyplývá, že

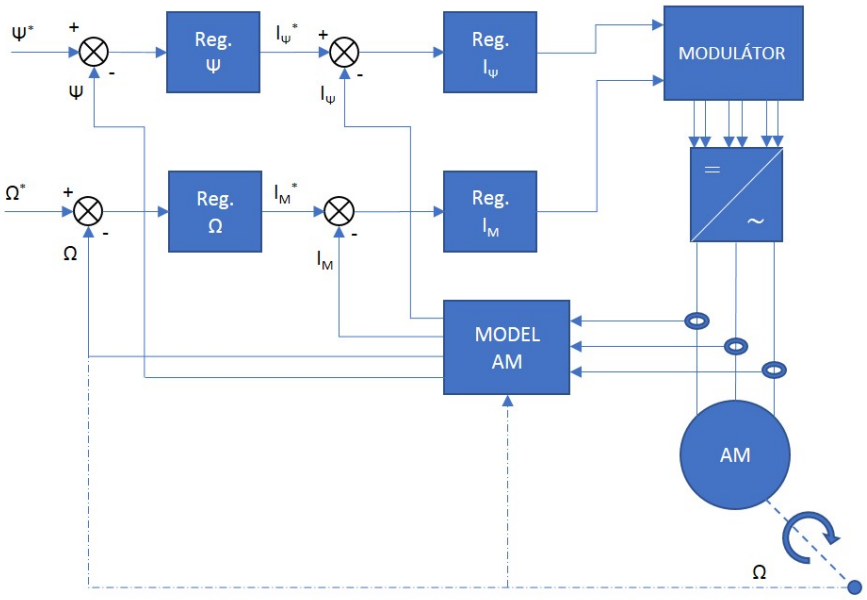
$$m_\mathrm{i} \approx \Psi_2 i_{1q}, \tag{3.16}$$

$$\Psi_2 \approx L_\mathrm{m} i_{1d}. \tag{3.17}$$

Složka proudu i_{1q} je úměrná momentu stroje a nazývá se **momentotvorná složka** statorového proudu, složka i_{1d} je úměrná toku stroje a nazývá se **tokotvorná složka** statorového proudu.

3.3 Principiální blokové schéma vektorové regulace

Na obr. 3.3 je znázorněno principiální schéma vektorové regulace.



Obrázek 3.3. Principiální schéma vektorové regulace

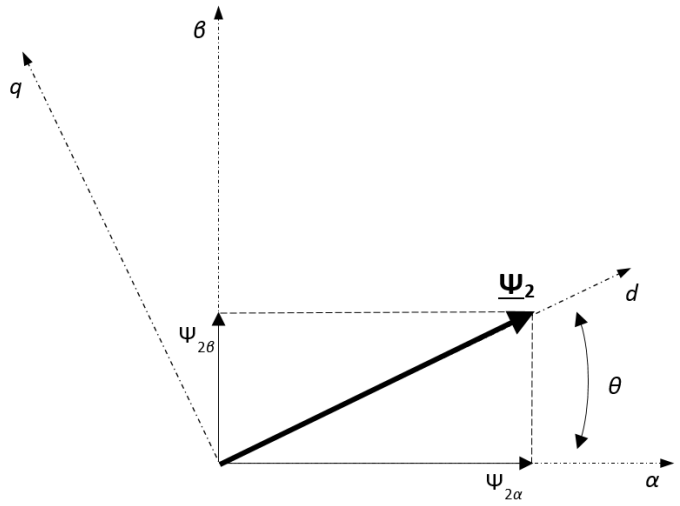
Výstupem regulátoru toku je požadovaná hodnota tokotvorné složky proudu, výstupem regulátoru otáček pak požadovaná hodnota momentotvorné složky proudu. Skutečné hodnoty toku, momentotvorného proudu a tokotvorného proudu jsou vypočítávány v bloku MODEL AM na základě rovnic matematického modelu asynchronního motoru, do kterého vstupují měřitelné veličiny motoru relevantní ke konkrétnímu typu modelu. V závislosti na tom, zda je zjišťování otáček senzorové nebo bezsenzorové, pak jako skutečná hodnota rychlosti do regulátoru otáček vstupují buď změřené, nebo vypočtené otáčky, což je ve schématu zachyceno čerchovanou čarou vedoucí z čidla otáček. Blok MODULÁTOR pak generuje na základě výstupní požadované hodnoty vektoru statorového napětí nebo proudu signály pro střídač, který spínáním přivede tento vektor na svorky motoru.

3.4 Získávání polohy toku

Konkrétních způsobů vektorové regulace je celá řada a lze je dělit dle mnoha kritérií. Jestliže vektorové řízení orientujeme na rotorový tok $\underline{\Psi}_2$, tak je nutné znát okamžitou polohu vektoru tohoto toku. Dle způsobu získávání této polohy lze metody v zásadě dělit na Direct Field Oriented Control (DFOC), tj. na přímé vektorové řízení, a Indirect Field Oriented Control (IFOC), tj. na nepřímé vektorové řízení. U přímých metod je rotorový tok získáván výpočtem na základě ostatních měřitelných veličin motoru (napětí, proudy, otáčky) a u nepřímých pak výpočtem na základě rychlosti otáčení rotoru a žádaných veličin.[6]

3.4.1 Výpočet transformačního úhlu

Abychom mohli odděleně řídit složky proudu i_{1d} a i_{1q}, je nutné znát jejich skutečné hodnoty. Ty získáme Clarkové transformací (kap. 2.4.3) fázových statorových proudů ze systému abc do systému $\alpha\beta$ a následnou Parkovou transformací (kap. 2.4.4) ze systému $\alpha\beta$ do dq. Pro Parkovu transformaci je ale nutné znát okamžitý úhel θ mezi souřadnicovým systémem $\alpha\beta$ a dq (obr. 3.4).[2–3, 5–6]



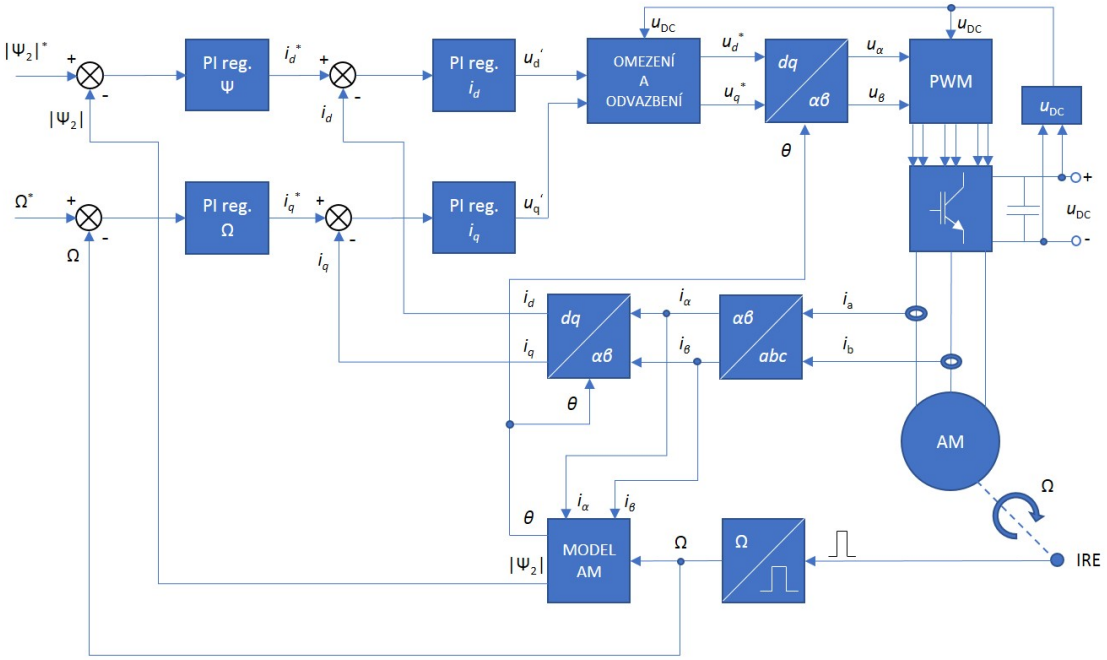
Obrázek 3.4. Transformační úhel

U nepřímých metod vektorového řízení je transformační úhel získáván integrací součtu elektrické úhlové rychlosti rotoru ω a dle druhé rovnice soustavy (3.6) ze žádaných hodnot vypočtené elektrické skluzové rychlosti ω_{slip} [2, 7]

$$\theta = \int (\omega + \omega_{\text{slip}}) \mathrm{d}t. \tag{3.18}$$

U přímých metod vektorového řízení je úhel počítán ze složek vektoru rotorového toku $\Psi_{2\alpha}, \Psi_{2\beta}$ v souřadnicích $\alpha\beta$, které lze získat řešením některého z matematických modelů stroje (viz kap. 3.1). Úhel θ je potom možné vypočítat (viz obr. 3.4) jako [2, 7]

$$\theta = \arctan \frac{\Psi_{2\beta}}{\Psi_{2\alpha}}. \tag{3.19}$$



Obrázek 3.5. Zjednodušené blokové schéma otáčkové regulační smyčky

3.5 Navržené schéma vektorové regulace

Na obr. 3.5 je znázorněno zjednodušené schéma vektorového řízení realizovaného v praktické části. Podrobnému popisu celé navržené struktury řízení včetně programové implementace jednotlivých bloků je věnována kap. 7.

Na základě dvou změřených fázových proudů (statorové vinutí motoru spojeno do hvězdy) je provedena Clarkové transformace dle rovnic (2.24). V dalším bloku je provedena Parkova transformace, pro kterou je nutné znát transformační úhel. Ten je vypočítáván ze vztahu (3.19) v bloku MODEL AM. Složky $\Psi_{2\alpha}, \Psi_{2\beta}$ jsou získávány řešením rovnic (3.5) I-n modelu asynchronního stroje. Výstupem z bloku MODEL AM je rovněž vypočtená hodnota rotorového magnetického toku. Otáčivá rychlost je vypočítávána na základě výstupních pulzů kvadraturního enkodéru.

Skutečné hodnoty toku, resp. otáček motoru pak vstupují do tokového, resp. otáčkového regulátoru. Výstupem regulátoru tokotvorné složky proudu je žádaná složka statorového napětí v ose d, výstupem regulátoru momentotvorné složky proudu je žádaná složka statorového napětí v ose q. Jako skutečné hodnoty vstupují do regulátorů tokotvorného a momentotvorného proudu výstupní transformované proudy z bloku Parkovy transformace.

Složky žádaného vektoru statorového napětí v systému dq jsou odvazbeny (o problematice odvazbení pojednává kap. 3.6), omezeny podle dostupného napětí ve stejnosměrném meziobvodu třífázového napěťového střídače, transformovány pomocí inverzní Parkovy transformace do systému $\alpha\beta$ a následně přivedeny do modulátoru. Modulátor pak generuje řídicí signály pro střídač.

Poznámka: Řídicí algoritmus implementovaný v realizační části umožňuje výběr z více regulačních smyček. Ty se ale liší pouze řazením regulátorů v momentotvorné regulační větvi, případně různým nastavením omezení veličin, se kterými regulátory pracují.

3.6 Odvazbení

Vyjádřeme z posledních dvou rovnic soustavy (2.39) proudy i_{2d} a i_{2q}

$$i_{2d} = \frac{\Psi_{2d} - L_\mathrm{m} i_{1d}}{L_2}, \tag{3.20}$$

$$i_{2q} = \frac{\Psi_{2q} - L_\mathrm{m} i_{1q}}{L_2}. \tag{3.21}$$

Tyto vztahy dosadíme do rovnic (2.39) pro složky statorového toku Ψ_{1d} a Ψ_{1q}. Po úpravě dostáváme

$$\Psi_{1d} = L_1 i_{1d} + \frac{L_\mathrm{m}}{L_2}\Psi_{2d} - \frac{L_\mathrm{m}^2}{L_2} i_{1d}, \tag{3.22}$$

$$\Psi_{1q} = L_1 i_{1q} + \frac{L_\mathrm{m}}{L_2}\Psi_{2q} - \frac{L_\mathrm{m}^2}{L_2} i_{1q}. \tag{3.23}$$

Získaná vyjádření pro složky toku následně dosadíme do rovnic (2.39) pro složky statorového napětí u_{1d} a u_{1q}. Dosazením do rovnice pro u_{1d} za vzniklý člen $\mathrm{d}\Psi_{2d}/\mathrm{d}t$ z (3.6) a provedení úpravy s respektováním, že $\Psi_{2q} = 0$, získáváme

$$u_{1d} = \underbrace{(R_1 + \frac{L_\mathrm{m}^2 R_2}{L_2^2}) i_{1d} + (L_1 - \frac{L_\mathrm{m}^2}{L_2})\frac{\mathrm{d}i_{1d}}{\mathrm{d}t}}_{\text{lineární část}} \underbrace{- \frac{L_\mathrm{m} R_2}{L_2^2}\Psi_{2d} - \omega_\mathrm{s}(L_1 - \frac{L_\mathrm{m}^2}{L_2}) i_{1q}}_{\text{vazba}}, \tag{3.24}$$

$$u_{1q} = \underbrace{R_1 i_{1q} + (L_1 - \frac{L_\mathrm{m}^2}{L_2})\frac{\mathrm{d}i_{1q}}{\mathrm{d}t}}_{\text{lineární část}} \underbrace{+ \omega_\mathrm{s}(L_1 - \frac{L_\mathrm{m}^2}{L_2}) i_{1d} + \omega_\mathrm{s}\frac{L_\mathrm{m}}{L_2}\Psi_{2d}}_{\text{vazba}}. \tag{3.25}$$

Ze vztahů (3.24) a (3.25) plyne několik důležitých závěrů. Prvním je důkaz v předchozí kapitole vysloveného předpokladu, že výstupem regulátoru tokotvorné složky proudu je žádaná složka statorového napětí v ose d a výstupem regulátoru momentotvorného proudu je pak žádaná složka statorového napětí v ose q. V rovnicích (3.24) a (3.25) jsou členy vyjadřující tuto úměru značeny jako „lineární část“.

Druhým je, že mimo lineární členy se zde také vyskytují členy znázorněné jako „vazba“, které jsou důsledkem provázanosti s ostatními motorovými veličinami. Napětí u_{1d} je dále funkcí momentotvorné složky proudu i_{1q}, synchronní elektrické úhlové rychlosti ω_s a rotorového toku Ψ_{2d}. Napětí u_{1q} je pak funkcí tokotvorné složky proudu i_{1d}, synchronní elektrické úhlové rychlosti ω_s a rotorového toku Ψ_{2d}. Přítomnost rotorového toku v rovnicích reprezentuje indukované napětí.

Třetí plyne pro použité proudové regulátory. Abychom dosáhli co nejkvalitnějších regulačních pochodů i s použitím jednoduchých PI regulátorů, je vhodné tuto vazbu kompenzovat přičtením vazebních členů k výstupu regulátorů před blokem omezení:

$$u_{d\,\mathrm{decouple}} = -\frac{L_\mathrm{m} R_2}{L_2^2}\Psi_{2d} - \omega_\mathrm{s}(L_1 - \frac{L_\mathrm{m}^2}{L_2}) i_{1q}, \tag{3.26}$$

$$u_{q\,\mathrm{decouple}} = \omega_\mathrm{s}(L_1 - \frac{L_\mathrm{m}^2}{L_2}) i_{1d} + \omega_\mathrm{s}\frac{L_\mathrm{m}}{L_2}\Psi_{2d}. \tag{3.27}$$

Synchronní elektrická úhlová rychlost je rovna součtu elektrické úhlové rychlosti hřídele a elektrické skluzové rychlosti

$$\omega_\mathrm{s} = \omega + \omega_\mathrm{slip} = \omega + \frac{L_\mathrm{m} R_2}{L_2}\frac{i_{1q}}{\Psi_{2d}}, \tag{3.28}$$

kde za ω_{slip} bylo dosazeno z (3.6). Při akceptování snížené přesnosti lze počítat pouze s rychlostí hřídele, která se od synchronní v ustáleném stavu liší jen velmi málo (o hodnotu skluzu). V mechanických přechodných dějích se tento rozdíl pak prohlubuje.

K podobnému tvaru rovnic pro odvazbení dospěli i autoři literatury [8] a [9]. Další pohled na problematiku včetně alternativních vztahů lze nalézt např. v [5] nebo [10].

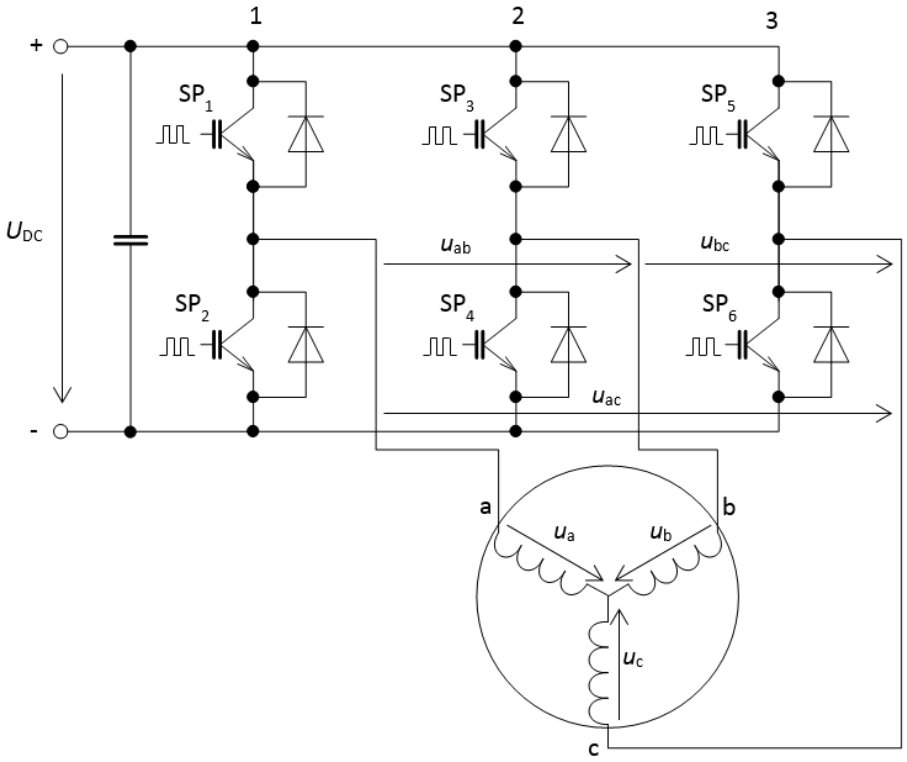
3.7 Napájecí měnič a jeho řízení

3.7.1 Střídač

Asynchronní motor, na němž probíhá vektorového řízení, je napájen přes napěťový dvouúrovňový střídač. Toto uspořádání je schématicky znázorněno na obr. 3.6.

Střídač se skládá ze tří větví. V každé větvi jsou zapojeny dva spínače (ve schématu znázorněné jako IGBT tranzistory) opatřené zpětnými diodami. Celkem je tedy použito šest spínačů SP_1 až SP_6. Každý z nich je pak řízen pulzy generovanými uvnitř řídicího mikrokontroléru. Ve stejnosměrném meziobvodu je pak zapojen jeden nebo více kondenzátorů do série, na kterých lze naměřit napětí U_{DC}. Mezi dva tranzistory v každé větvi je pak připojena jedna fáze třífázové zátěže, kterou v našem případě tvoří asynchronní motor.

Při generování řídicích signálů pro budiče tranzistorů je nutné zajistit, aby nikdy nebyly sepnuty dva tranzistory v jedné větvi. Takovéto spojení by znamenalo zkrat napájecího zdroje, což by bez včasného zapůsobení ochran vedlo ke zničení tranzistorů v měniči. Proto se mezi řídicí signály pro horní a spodní tranzistor vkládá ochranná doba, tzv. dead-time, který musí být zvolen tak, aby se sepnutý tranzistor stačil dostat do stavu vysoké impedance. Až poté může být vydán povel pro sepnutí druhého tranzistoru ve stejné větvi.[7]



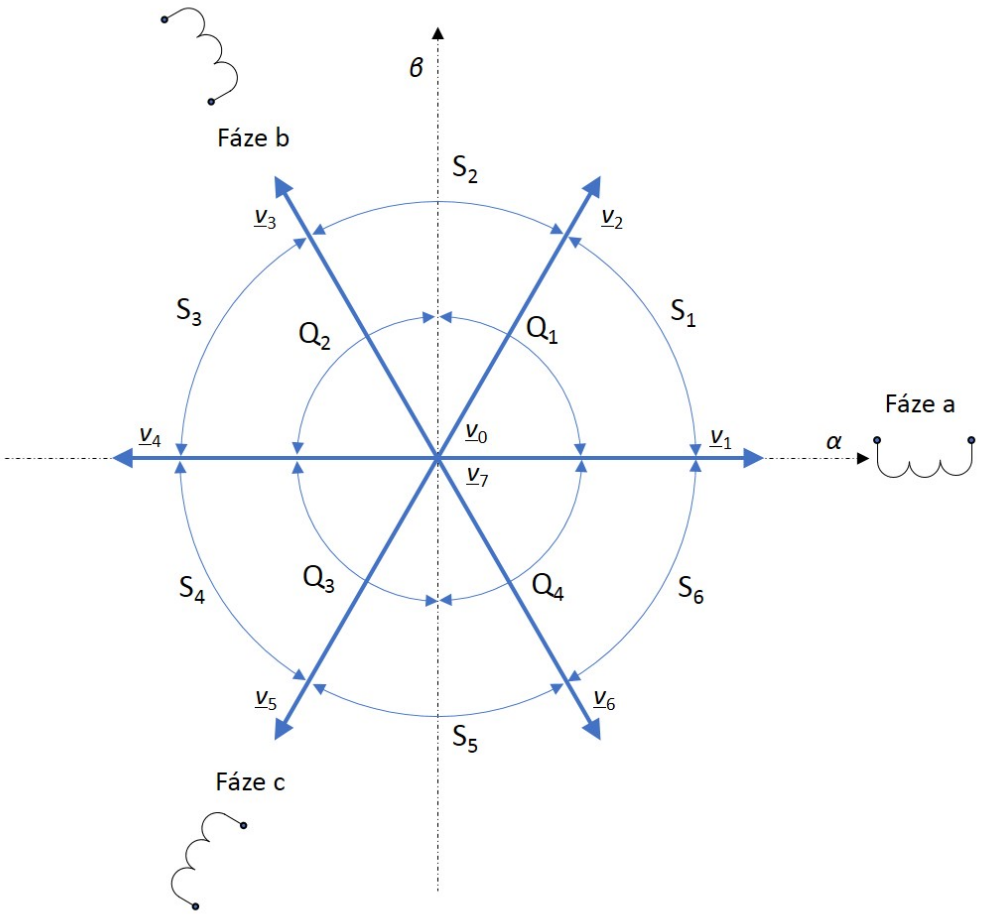
Obrázek 3.6. Asynchronní motor napájený třífázovým napěťovým střídačem

U dvouúrovňového střídače existuje osm možných přípůstných kombinací sepnutí jednotlivých spínačů. Každá z těchto kombinací pak odpovídá jednomu prostorovému vektoru napětí v souřadnicovém systému $\alpha\beta$ svázaného se statorem. Poloha těchto vektorů ve vztahu k jednotlvým statorovým vinutím je znázorněna na obr. 3.7. Reálná a imaginární osa rozděluje rovinu $\alpha\beta$ na čtyři kvadranty Q_1 až Q_4, jednotlivé napěťové vektory pak na šest sektorů S_1 až S_6. Vektory $\underline{v}_0$, resp. $\underline{v}_7$ jsou nulové vektory, které vznikají sepnutím všech dolních, resp. horních tranzistorů. V těchto případech je pak na všech fázích motoru nulové napětí.[7, 11]

V tab. 3.1 je uvedeno, pomocí jakých kombinací sepnutí střídačových tranzistorů získáme základní vektory napětí. U jednotlivých fází a, b, c odpovídá „1“ sepnutí horního spínače větve, ke které je fáze připojena a „0“ pak sepnutí dolního spínače.[7, 11]

	$\underline{v}_0$	$\underline{v}_1$	$\underline{v}_2$	$\underline{v}_3$	$\underline{v}_4$	$\underline{v}_5$	$\underline{v}_6$	$\underline{v}_7$
a	0	1	1	0	0	0	1	1
b	0	0	1	1	1	0	0	1
c	0	0	0	0	1	1	1	1

Tabulka 3.1. Základní vektory napětí a jim odpovídající sepnutí spínačů



Obrázek 3.7. Základní vektory napětí střídače (Q...kvadrant, S...sektor)

3.7.2 Modulace prostorového vektoru

Nyní se blíže podíváme, jakým způsobem lze ze základních vektorů napětí uvedených na obr. 3.7 získat obecný napěťový vektor. Předpokládejme, že požadovaný vektor napětí $\underline{u}_1^*$ se nachází v sektoru S_1. Situace je znázorněna na obr. 3.8. Napětí $\underline{u}_1^*$ lze získat vektorovým součtem napětí $\underline{u}_r$ a $\underline{u}_l$, které leží ve směru základních vektorů $\underline{v}_1$ a $\underline{v}_2$. Modul všech základních prostorových vektorů napětí má velikost [7]

$$|\underline{u}_{1\max}| = |\underline{v}_1| = |\underline{v}_2| = |\underline{v}_3| = |\underline{v}_4| = |\underline{v}_5| = |\underline{v}_6| = \frac{2}{3}U_{\mathrm{DC}}. \tag{3.29}$$

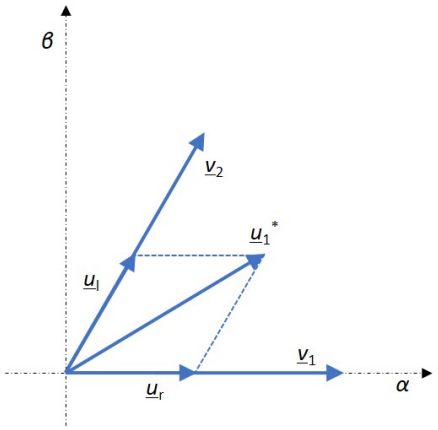
Označmě T_p dobu mezi okamžiky, kdy je do modulátoru vložen požadavek na nový napěťový vektor. Vektory $\underline{u}_\mathrm{l}$ a $\underline{u}_\mathrm{r}$ získáme různým poměrem dob sepnutí základních vektorů $\underline{v}_1$ a $\underline{v}_2$ během spínací periody T_p. Označme T_l dobu sepnutí $\underline{u}_\mathrm{l}$ a T_r dobu sepnutí $\underline{u}_\mathrm{r}$. Pak platí [7]

$$\begin{aligned} T_\mathrm{l} &= T_\mathrm{p}\frac{|\underline{u}_l|}{|\underline{u}_{1\max}|} \\ T_\mathrm{r} &= T_\mathrm{p}\frac{|\underline{u}_r|}{|\underline{u}_{1\max}|}. \end{aligned} \tag{3.30}$$

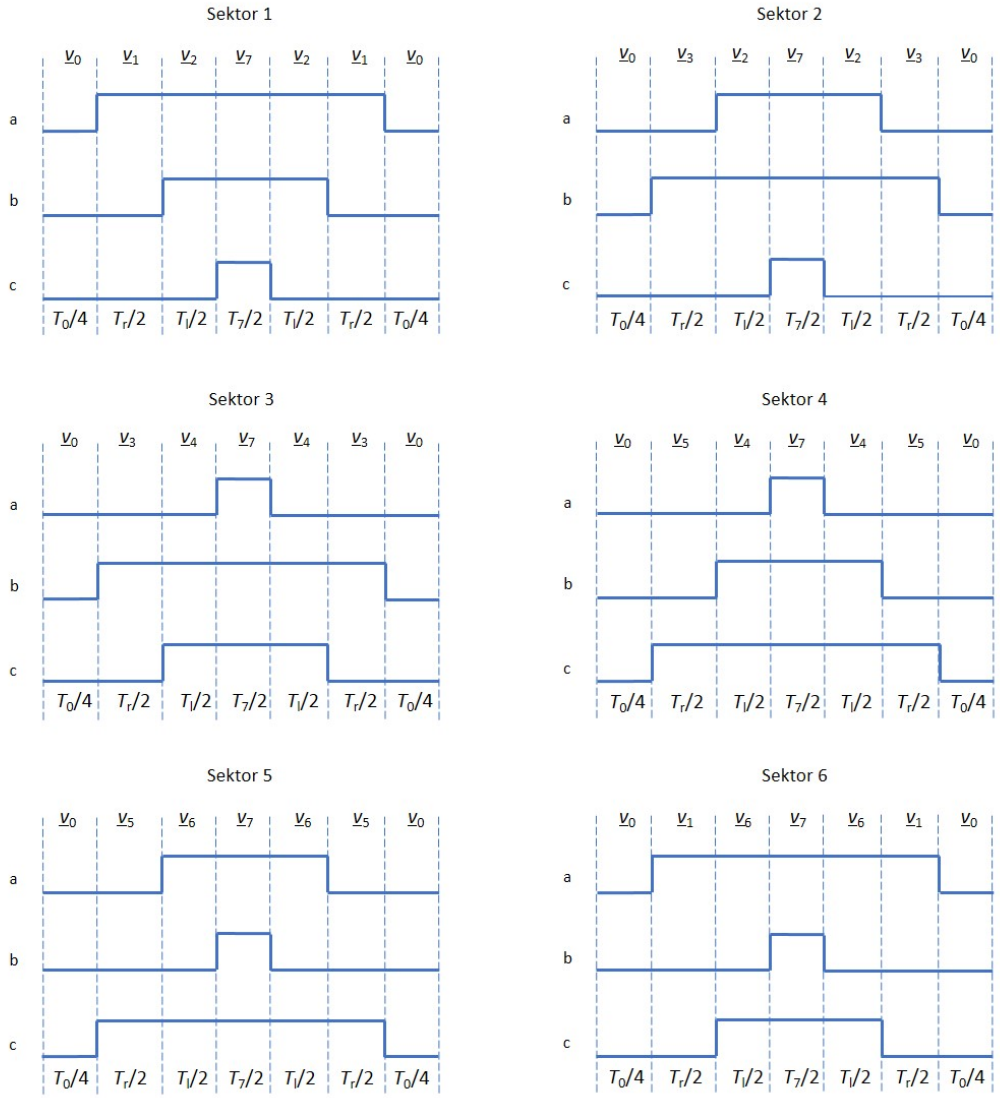
Zbytek času $T_\mathrm{p} - (T_\mathrm{l} + T_\mathrm{r})$ pak vyplňuje jeden z nulových vektorů $\underline{v}_0$ nebo $\underline{v}_7$. Požadovaný vektor napětí $\underline{u}_1^*$ lze tedy získat jako [7]

$$\underline{u}_1^* = \frac{T_\mathrm{r}}{T_\mathrm{p}}\underline{v}_1 + \frac{T_\mathrm{l}}{T_\mathrm{p}}\underline{v}_2 + \frac{T_\mathrm{p} - (T_\mathrm{l} + T_\mathrm{r})}{T_\mathrm{p}}\underline{v}_0 \,(\text{nebo}\,\underline{v}_7). \tag{3.31}$$

Obdobně bychom postupovali pro další sektory. Vezmeme-li v úvahu hledisko minimálních spínacích ztrát a také to, že pro generování PWM uvnitř mikropočítače se používají převážně obousměrné čítače (up/down counters), vychází jako nejvýhodnější spínací sekvence znázorněná pro všechny sektory na obr. 3.9. Tranzistory v jedné větvi jsou pak spínány komplementárně a řídicí signály pro páry v jednotlivých větvích jsou osově symetrické kolem vrcholové (maximální) hodnoty obousměrného čítače.[7, 11]



Obrázek 3.8. Složení požadovaného vektoru napětí z vektorů základních pro sektor 1



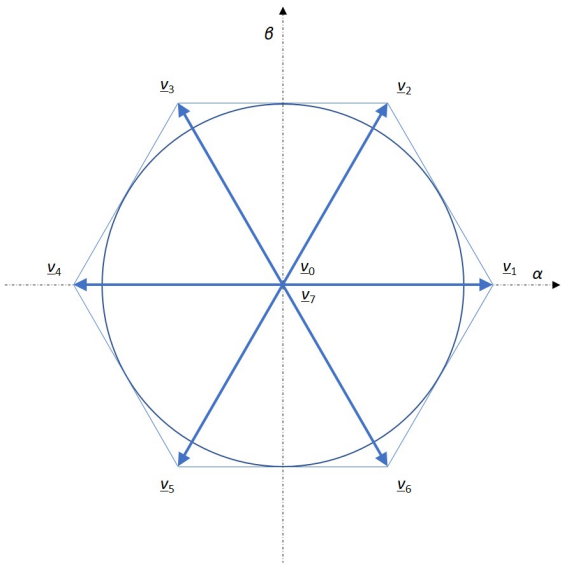
Obrázek 3.9. Sekvence spínacích pulzů v jednotlivých sektorech

3.7.3 Omezení při generaci obecného prostorového vektoru

Dá se ukázat [7], že teoreticky maximální možná generovatelná množina prostorových vektorů leží uvnitř pravidelného šestiúhelníku, který je tvořen spojnicemi koncových bodů základních napěťových vektorů (obr. 3.10). To plyne z toho, že vektorový součet $\underline{u}_l$ a $\underline{u}_r$ neodpovídá skalárnímu součtu T_l a T_r. V praxi se ale většinou kvůli omezení vyšších harmonických používá oblast tvořená vepsanou kružnicí, jejíž poloměr je roven [7, 11]

$$|\underline{u}_{1\max}| = \frac{1}{\sqrt{3}} U_\mathrm{DC}. \tag{3.32}$$

Další omezení plyne z dynamických vlastností použitých polovodičových spínačů. Pokud bude doba trvání nulových vektorů T_0 a T_7 příliš krátká, tak se může stát, že některý ze spínačů nedokáže včas sepnout, resp. vypnout, neboť řídicí puls pro jeho budič bude kratší, než je doba potřebná pro jeho přechod ze stavu rozepnuto do stavu sepnuto a naopak. Tato situace nastává v blízkosti hran již zmíněného šestiúhelníku



Obrázek 3.10. Omezení pro realizaci obecného prostorového vektoru napětí

na obr. 3.10. K určité odchylce výstupního vektoru oproti žádanému dochází i vlivem vloženého dead-timu.

3.7.4 Výpočet spínacích časů

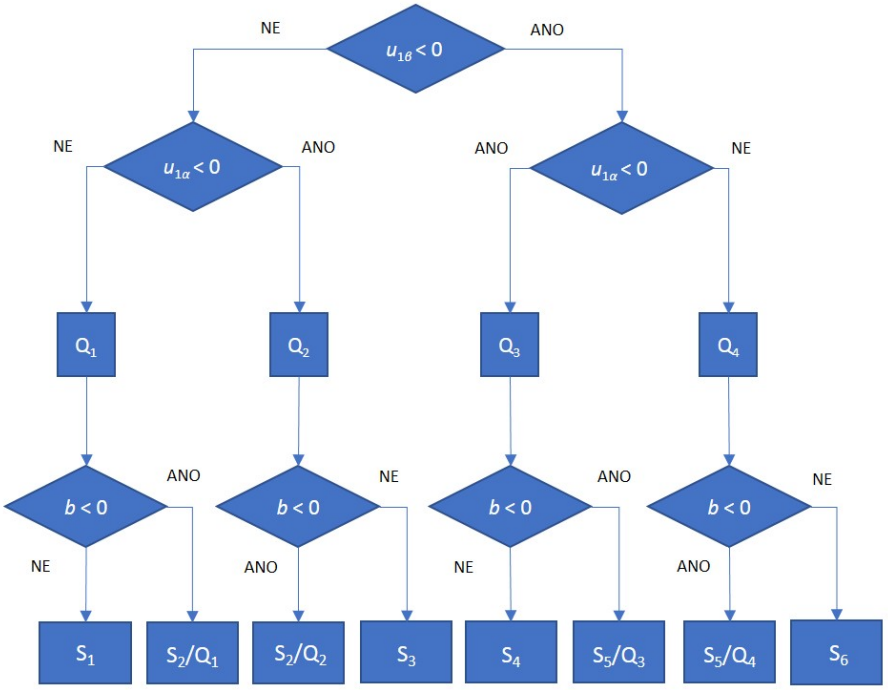
Z rovnic (3.30) plyne, že pro výpočet doby trvání příslušných základních vektorů je nutné znát velikosti $|\underline{u}_\mathrm{l}|$ a $|\underline{u}_\mathrm{r}|$. Jedna z možných strategií, která je založena na znalosti složek požadovaného napěťového vektoru v souřadnicovém systému $\alpha\beta$, který je výstupem z proudových regulátorů, vychází z tabulky 3.2.[7]

		$\lvert\underline{u}_\mathrm{r}\rvert$	$\lvert\underline{u}_\mathrm{l}\rvert$
S_1	Q_1	$\lvert u_{1\alpha}\rvert - \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$\frac{2}{\sqrt{3}}\lvert u_{1\beta}\rvert$
S_2	Q_1	$\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$-\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$
	Q_2	$-\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$
S_3	Q_2	$\frac{2}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$\lvert u_{1\alpha}\rvert - \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$
S_4	Q_3	$\lvert u_{1\alpha}\rvert - \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$\frac{2}{\sqrt{3}}\lvert u_{1\beta}\rvert$
S_5	Q_3	$\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$-\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$
	Q_4	$-\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$\lvert u_{1\alpha}\rvert + \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$
S_6	Q_4	$\frac{2}{\sqrt{3}}\lvert u_{1\beta}\rvert$	$\lvert u_{1\alpha}\rvert - \frac{1}{\sqrt{3}}\lvert u_{1\beta}\rvert$

Tabulka 3.2. Výpočet velikosti hraničních vektorů napětí na základě znalosti $u_{1\alpha}$ a $u_{1\beta}$

Při bližším pohledu zjistíme, že se v tabulce vyskytují pouze tři členy:[7]

$$\begin{aligned} a &= |u_{1\alpha}| + \frac{1}{\sqrt{3}}|u_{1\beta}| \\ b &= |u_{1\alpha}| - \frac{1}{\sqrt{3}}|u_{1\beta}| \\ c &= \frac{2}{\sqrt{3}}|u_{1\beta}|. \end{aligned} \tag{3.33}$$



Obrázek 3.11. Postup stanovení sektoru a kvadrantu pro požadovaný vektor napětí

Zbývá už jen určit v jakém sektoru, resp. kvadrantu, se požadovaný vektor nachází. Na základě znamének $u_{1\alpha}$ a $u_{1\beta}$ lze určit kvadrant a ze znaménka členu b (rovnice (3.33)) pak sektor. Celý postup je znázorněn vývojovým diagramem na obr. 3.11.[7]

3.7.5 Omezení žádaného napětí vzhledem k minimální šířce pulzu

V kap. 3.7.3 již bylo zmíněno, že v oblasti kolem hran šestiúhelníku tvořeného spojnicemi koncových bodů základních napěťových vektorů nastává situace, kdy je doba trvání nulového napěťového vektoru příliš krátká. To v důsledku způsobí, že tranzistory nemají čas plně sepnout, resp. vypnout. Na obr. 3.12 je tato oblast vyznačena světle modrou barvou.

Jedno z možných řešení je provést oříznutí vypočtených komparačních hodnot vzhledem k minimální, resp. maximální hodnotě. Toto řešení je sice bezpečné a nezávislé na stejnosměrném napětí v meziobvodu, na druhou stranu v důsledku vede k tomu, že žádané napětí vstupující do modulátoru neodpovídá skutečnosti.

Další řešení je naznačeno na obr. 3.12. V principu se jedná o redukci množiny požadovatelných napětí z původní vepsané kružnice o poloměru $|\underline{u}_{1\max}| = U_{\mathrm{DC}}/\sqrt{3}$ na kružnici vepsanou vnitřnímu šestiúhelníku zmenšeného o barevně vyznačenou zakázanou oblast, jejíž poloměr je roven velikosti vektoru $\underline{\tilde{u}}_{1\max}$. Odvozením zde bude ukázáno, že tento poloměr se dá vyjádřit v závislosti na zvolené minimální délce sepnutí nulového vektoru $T_{0\min}$ jako násobek modulu vektoru $|\underline{u}_{1\max}|$. Bez újmy na obecnosti bude provedeno odvození pro první sektor s hraničními vektory $\underline{v}_1$ a $\underline{v}_2$. Vektor $\underline{u}_{1\max}$ získáme z vektorů $\underline{v}_1$ a $\underline{v}_2$ tak, že každý z nich bude sepnut právě polovinu spínací periody T_{p}. Označme $T_1 = \frac{1}{2}T_{\mathrm{p}}$, lze tedy psát

$$\underline{u}_{1\max} = \frac{T_1}{T_{\mathrm{p}}}\underline{v}_1 + \frac{T_1}{T_{\mathrm{p}}}\underline{v}_2 = \frac{T_1}{T_{\mathrm{p}}}(\underline{v}_1 + \underline{v}_2). \tag{3.34}$$

Vektor $\tilde{\underline{u}}_{1\max}$ lze vyjádřit jako λ-násobek vektoru $\underline{u}_{1\max}$. Pro jeho získání je nutné právě po dobu $T_{0\min}$ sepnout i vektoru nulový. Hraniční základní vektory budou nyní každý sepnut po dobu T_2:

$$\lambda \underline{u}_{1\max} = \frac{T_2}{T_\mathrm{p}} \underline{v}_1 + \frac{T_2}{T_\mathrm{p}} \underline{v}_2 + T_{0\min} \underline{v}_0 = \frac{T_2}{T_\mathrm{p}} (\underline{v}_1 + \underline{v}_2) + T_{0\min} \underline{v}_0. \tag{3.35}$$

Dobu sepnutí T_2 lze vyjádřit pomocí T_1 (součet všech dob sepnutí musí dát dohromady T_p) jako

$$T_2 + T_2 + T_{0\min} = T_\mathrm{p} = 2T_1 \;\Rightarrow\; T_2 = T_1 - \frac{T_{0\min}}{2}. \tag{3.36}$$

Nyní dosadíme (3.36) do (3.35) a následně odečteme od (3.34). Po úpravě máme

$$\underline{u}_{1\max}(1-\lambda) = \frac{T_{0\min}}{T_\mathrm{p}} [\frac{1}{2}(\underline{v}_1 + \underline{v}_2) - \underline{v}_0]. \tag{3.37}$$

Vztah (3.37) lze s využitím toho, že

$$\frac{1}{2}(\underline{v}_1 + \underline{v}_2) - \underline{v}_0 = \underline{u}_{1\max} - \underline{v}_0 = \underline{u}_{1\max}$$

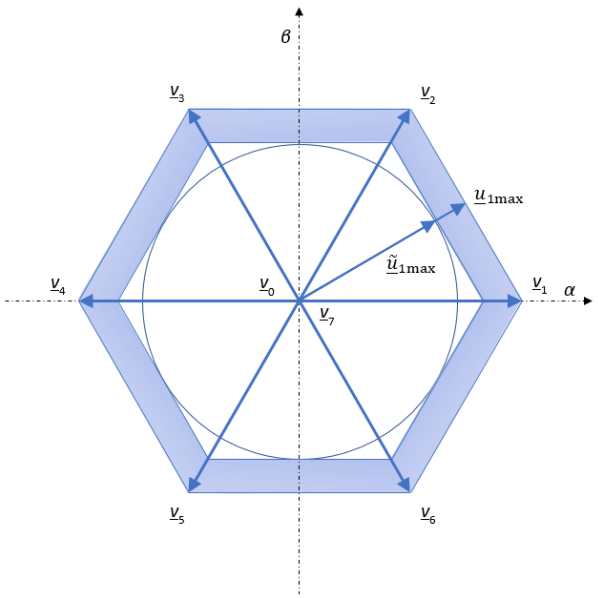
upravit na

$$\underline{u}_{1\max}(1-\lambda) = \frac{T_{0\min}}{T_\mathrm{p}} \underline{u}_{1\max}. \tag{3.38}$$

Aby byla rovnost (3.38) splněna, tak je zřejmé, že musí platit

$$\lambda = 1 - \frac{T_{0\min}}{T_\mathrm{p}}. \tag{3.39}$$

Abychom v závislosti na zvolené minimání délce trvání nulových vektorů omezili množinu žádaných napětí na oblast vyznačenou kružnicí na obr. 3.12, je nutné původní maximální napětí $|\underline{u}_{1\max}| = U_\mathrm{DC}/\sqrt{3}$ snížit právě λ-krát.



Obrázek 3.12. Omezení žádaného vektoru napětí vzhledem k minimální šířce pulzu

Kapitola 4
S/W a H/W prostředky použité při vývoji

V této kapitole jsou uvedeny informace o hardwaru a softwaru použitém během vývoje programu pro vektorové řízení. Při popisu funkčních bloků použitého mikrokontroléru je kladen důraz zejména na ty části, které jsou relevantní pro tuto práci.

4.1 eZdsp F28335

Celý vývoj programu pro vektorové řízení probíhal na desce eZdsp F28335 od společnosti Spectrum Digital. Tato deska, jejíž srdcem je signálový procesor TMS320F28335 od firmy Texas Instruments, obsahuje konektory a periferní obvody zajišťující jednak komunikaci mezi deskou a počítačem (nahrání kódu, debugging) a jednak umožňující ovládání akčních členů řízeného systému na základě nahraného programu. Rozhraní pro emulátory poskytuje onboard JTAG konektor. Pro usnadnění vývoje kódu a zkrácení času nutného pro debugging je k dispozici C2000 Code Composer Studio driver.[12]

Klíčový hardware:[12]

- 30 MHz vstupní hodiny,
- onboard RS-232 konektor s line driverem,
- množství expanzních konektorů (analog, vstup/výstup),
- vestavěný USB JTAG Controller,
- napájecí napětí 5 V,
- onboard IEEE 1149.1 JTAG emulation konektor.

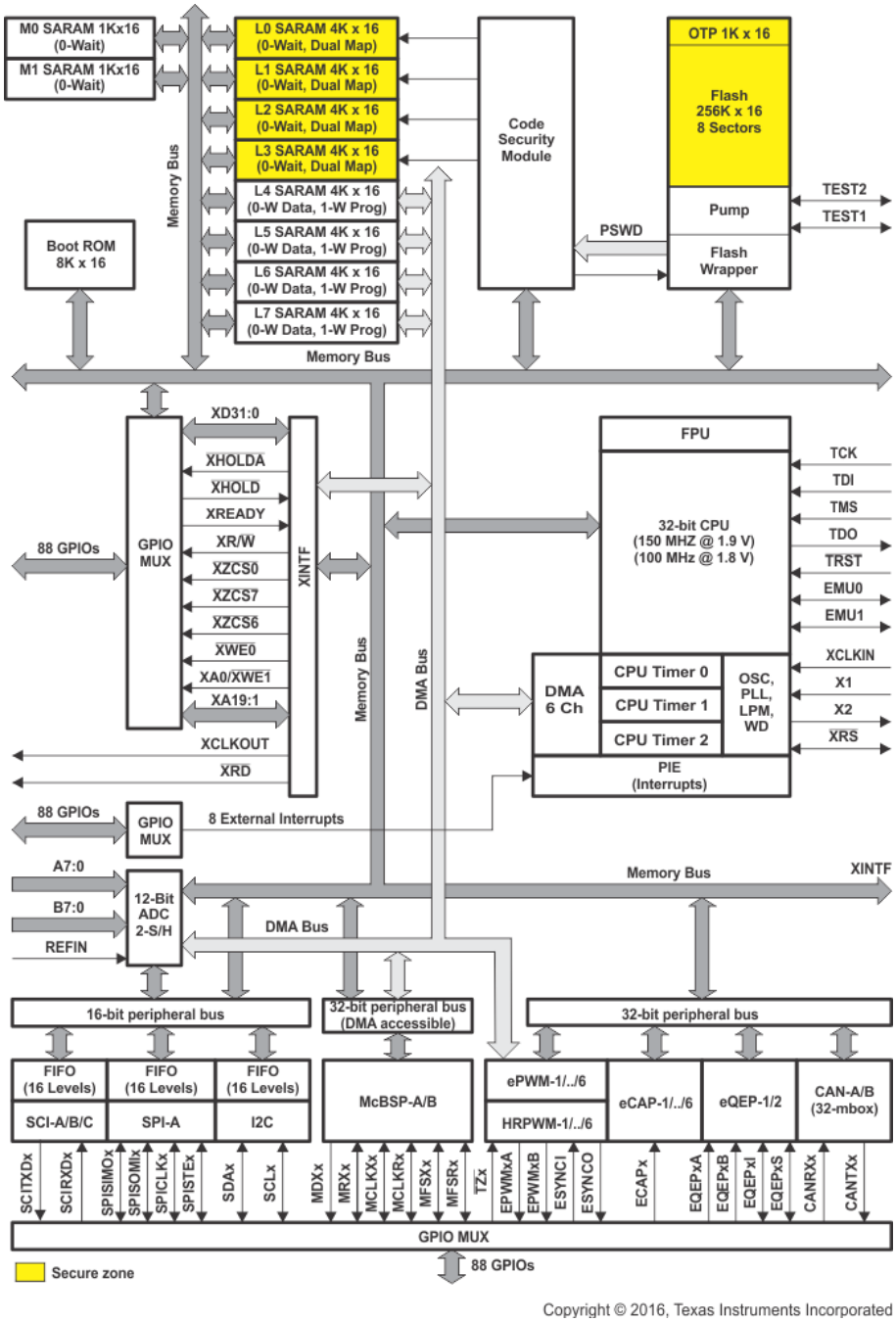
UPOZORNĚNÍ: Vstupy a výstupy TMS320F28335 pracují na napěťové úrovni 3, 3 V. Připojením vyššího napětí na vstup či výstup desky může dojít k nevratnému poškození procesoru!

4.2 TMS320F28335

4.2.1 Základní přehled

TMS320F28335 (dále jen „ F28335“) je digitální signálový procesor (DSP) z řady Delfino, která je podskupinou rodiny procesorů C2000. Ta v sobě zahrnuje 32bitové mikrokontroléry navržené pro pokročilé řízení zpětnovazebních systémů. Blokové schéma F28335 je na obr. 4.1.[13]

Jádro F28335 tvoří 32bitová centrální procesorová jednotka (CPU) spolu se 32bitovou single-precision jednotkou pro výpočty v plovoucí řádové čárce (FPU). Architektura procesoru je Harvardská, což znamená, že sběrnice a tudíž i paměť pro program a data jsou odděleny. Maximální možná taktovací frekvence procesoru, která se odvijí od použitého oscilátoru a konfigurace integrovaného fázového závěsu (PLL) činí 150 MHz. F28335 zvládá efektvině zpracovat i kódy napsané v C/C++, což umožňuje uživateli komfortnější vývoj programu ve vyšším programovacím jazyce. Výpočetní rychlost



Obrázek 4.1. Funkční bloky signálového procesoru TMS320F28335 [13]

dále zvyšuje 8úrovňový pipelining, který dovoluje procesoru provádět 8 instrukcí naráz během jednoho taktu a rychlý systém přerušení s automatickým ukládáním kritických registrů (kontextu). F28335 má implementován i tzv. přímý přístup do pamětí (DMA), který zrychluje přenos dat mezi operační pamětí a periferiemi, resp. vstupně/výstupními zařízeními.[13]

Fyzická paměť F28335 se skládá z 34K × 16 single-acces-random-acces (SARAM), 256K × 16 flash, 8K × 16 boot read-only memory (ROM) a 1K × 16 one-time programable (OTP) memory. Boot ROM je předprogramovaná od výrobce bootovacím softwarem. Různými kombinacemi logických úrovni na pinech GPIO87/XA15, GPIO86/XA14, GPIO85/XA13 a GPIO86/XA12 lze volit bootovací mód po zapnutí

napájení. V boot ROM se rovněž nachází standardní matematické tabulky pro výpočty některých funkcí v matematických algoritmech.[13]

F28335 obsahuje množství sériových komunikačních periferií. Jmenovitě eCAN, McBSP, SCI, SPI a I2C. Mimo komunikační periferie disponuje mikrokontrolér rovněž řídicími periferiemi jako je ePWM (enhanced PWM), eCAP (enhanced capture), eQEP (enhanced quadrature encoder pulse) a ADC (analog to digital converter – analogově číslicový převodník). Dále uvnitř F28335 můžeme nalézt ještě např. tři 32bitové CPU-Timery, watchdog timer, oscilátor a PLL (phase-locked loop – obvod fázového závěsu). Po obvodu čipu je 88 individuálně programovatelných a multiplexovatelných GPIO (general purpose input/output – vstupů/výstupů pro všeobecné použití). Důležitým blokem je Peripheral Interrupt Expansion (PIE) Block, který multiplexuje veškeré zdroje přerušení do menších skupin. Jako poslední ze stručného výčtu a popisu vlastností F28335 lze uvést možnost chránit program proti reverznímu inženýrství 128bitovým bezpečnostním kódem.[13]

4.2.2 General-Purpose Input/Output (GPIO)

Jednotlivé piny u F28335 jsou typicky sdílené. Přiřazení konkrétní funkce konkrétnímu pinu řídí GPIO multiplexovací registry (MUX). Každý GPIO pin je očíslován (GPIO0–GPIO87) a může být nakonfigurován jako čistý digitální vstup/výstup (směr se řídí zapsanými bity v registrech GPxDIR) nebo připojen k jednomu až ze tří vstupních či výstupních periferních signálů (přes GPxMUXn registry). GPIO jsou logicky členěny na tři vstupně/výstupní 32bitové porty – port A (GPIO0–GPIO31), port B (GPIO32–GPIO63) a port C (GPIO64–GPIO87). F28335 umožňuje kvalifikací vstupů (registry GPxQSELn, GPACTRL a GPBCTRL) filtraci nežádoucího šumu.[14]

4.2.3 Systém přerušení

Jak již bylo zmíněno v základním přehledu o F28335, o multiplexování velké skupiny všech zdrojů přerušení do skupiny menší se stará blok Peripheral Interrupt Expansion (PIE). Periferie využívají 58 z celkových 96 možných přerušení. Všech 96 přerušení je sekupeno do bloků po osmi. Každá skupina přerušení je dále vedena do jedné z dvanácti cest na úroveň CPU. Všech 96 zdrojů přerušení je reprezentován svým vektorem, který je uložen v příslušné části operační (RAM) paměti. Doba od žádosti do spuštění obsluhy přerušení, při níž CPU automaticky vyzvedne příslušnou adresu z tabulky vektorů přerušení a uloží kontext, trvá 9 hodinových cyklů. Každý zdroj přerušení může být na úrovni PIE zablokován. Prioritu přerušení lze řešit jak hardwarově, tak softwarově. F28335 na CPU úrovni podporuje jedno nemaskovatelné (NMI) a 16 maskovatelných zdrojů přerušení s prioritou (INT1–INT14, RTOSINT a DLOGINT). Tabulka vektorů přerušení, ve které se nachází adresy příslušných obsluh přerušení (ISR), by měla být inicializována v uživatelském kódu.[14]

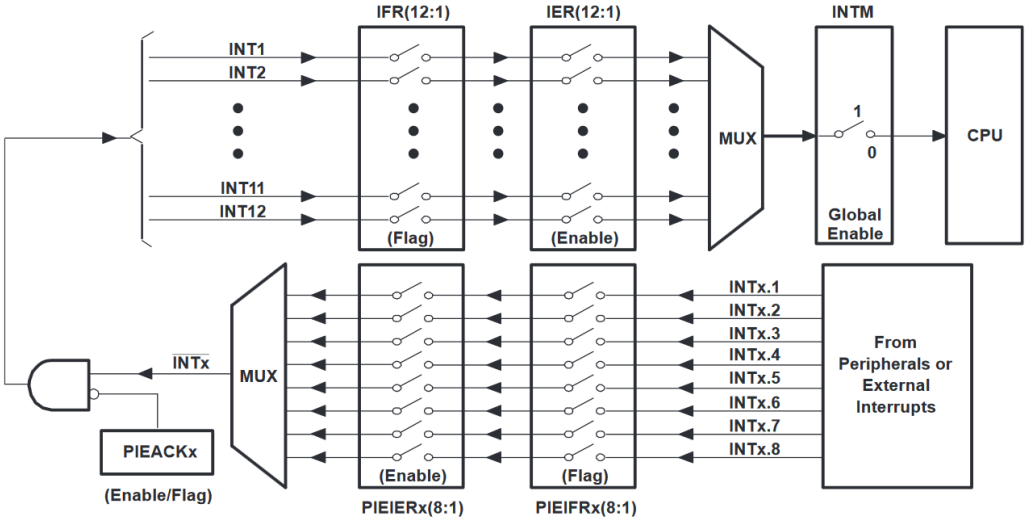
Obr. 4.2 znázorňuje celý proces od žádosti přerušení po jeho zpracování. Nemultiplexované zdroje jsou přivedeny k CPU přímo.

Celý proces obsluhy přerušení lze v souladu s obr. 4.2 rozdělit do jednotlivých úrovní:[14]

- **Periferní úroveň**
  Při aktivní žádosti o přerušení je nastaven příslušný příznak (IF). Pokud je zdroj tohoto přerušení povolen (IE), periferie vygeneruje požadavek o přerušení. Pokud tento zdroj přerušení není na periferní úrovni povolen, tak zůstává příznak aktivní do té doby, dokud není softwarově smazán. Jestliže je přerušení povoleno během

nastaveného příznaku, je rovněž vygenerována žádost o obsluhu. Přiznaky přerušení v řídicích registrech periferií musí být nulovány manuálně.

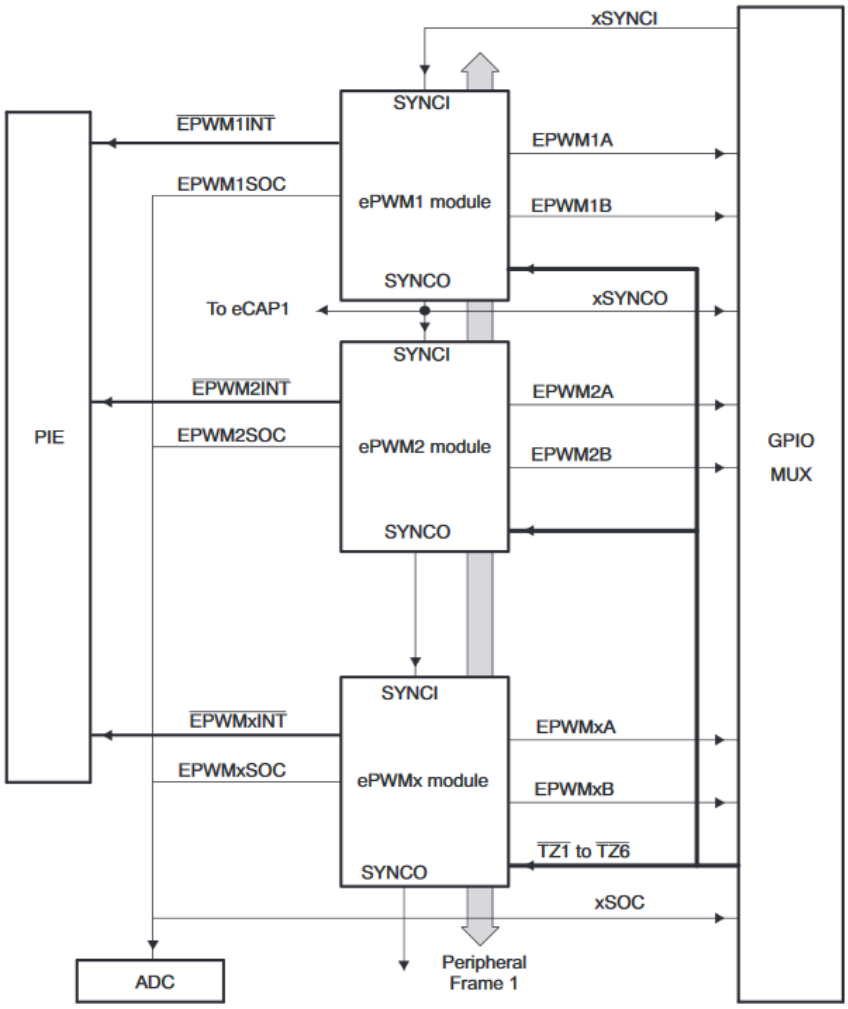
- **PIE úroveň**
  PIE blok multiplexuje osm periferních a externích přerušení do jedné žádosti o přerušení směrem k CPU. Tato přerušení jsou rozdělena do dvanácti skupin (PIE 1–PIE 12). Například skupina PIE 1 je multiplexována do skupiny přerušení CPU 1. Každá skupina přerušení na úrovni PIE má svůj příznakový (PIEIFRx) registr a povolovací (PIEIERx) registr (x = PIE 1–12). Každý bit, zde označovaný jako „y“, přísluší jednomu z osmi multiplexovaných zdrojů přerušení v rámci skupiny. Registry PIEIFRx.y a PIEIERx.y tedy odpovídají přerušení „y“ ve skupině „x“. Jakmile blok PIE obdrží informaci o aktivní žádosti o přerušení, je nastaven příslušný PIE příznak (PIEIFRx.y). Jestliže je toto přerušení povoleno (PIEIERx.y), PIE zkontroluje příslušný potvrzovací bit (PIEACKx), aby zjsitil, zdali je CPU připraveno obsloužit další přerušení z této skupiny. Pokud ano, pustí PIE blok žádost dále směrem k CPU.
- **CPU úroveň**
  Jakmile CPU obdrží žádost o přerušení, je nastaven příznak (IFR) příslušného zdroje (INTx) přerušení. Aby tato žádost o přerušení spustila proces samotné obsluhy, musí být povolena v povolovacím registru přerušení CPU (IER) a rovněž musí být povolen bit globální masky (INTM). Příslušný příznak přerušení v PIE bloku (PIEIFRx.y) je nulován automaticky po vyzvednutí adresy z tabulky vektorů přerušení. Avšak korespondující potvrzovací bit (PIEACKx) musí být nulován manuálně. Jen tehdy je možné obsloužit další žádosti o přerušení ze stejné skupiny.



Obrázek 4.2. Multiplexování zdrojů přerušení pomocí PIE bloku [14]

4.2.4 Enhanced Pulse Width Modulator (ePWM) Module

O generování řídicích signálů pro střídač se stará Enhanced PWM (ePWM). Ta obsahuje 6 ePWMx modulů a je navržena tak, aby bylo možné a uživatelsky snadné generovat i komplexní PWM signály a to při co nejnižším zatížení CPU prostředků. Každý z modulů může obsluhovat dva výstupy značené jako ePWMxA a ePWMxB. Moduly jsou propojeny synchronizačním hodinovým signálem a mohou tedy, pokud to aplikace vyžaduje, pracovat jako jeden systém. Vzájemné propojení je znázorněno na obr. 4.3.[15]



Obrázek 4.3. Propojení ePWM modulů [15]

Každý z ePWM modulů se dále skládá ze 7 submodulů, jejichž funkce bude vysvětlena dále. Jedná se o Time-Base (TB) modul, Counter-compare (CC) modul, Action-qualifier (AQ) modul, Dead-band (DB) modul, PWM-chopper (PC) modul, Event-trigger (ET) modul a Trip-zone (TZ) modul. Jednotlivé bloky submodulů v rámci ePWM společně s externími vstupními a výstupními signály jsou zachyceny na obr. 4.4.[15]

V následujícím výčtu jsou obsaženy stručné informace o každém submodulu:[15]

- **Time-base submodul (TB)** určuje časování všech událostí uvnitř modulu. Díky zabudované synchronizační logice může být toto časování sdíleno mezi několika moduly.

  Účel tohoto submodulu:
  - synchronizace hodin mezi moduly,
  - nastavení periody, resp. frekvence událostí na základě time-base counter (TBCTR) registru,
  - nastavení fázového posuvu vůči ostatním synchronizovaným ePWM modulům,
  - nastavení vzestupného, sestupného či obousměrného čítání,
  - generování události: Time-base counter se rovná periodě (TBCTR = TBPRD),
  - generování události: Time-base counter se rovná nule (TBCTR = 0x0000),
  - nastavení předděličky vstupních CPU systémových hodin (SYSCLKOUT) pro time-base counter.

- **Counter-compare submodul (CC)** neustále porovnává hodnotu TBCTR s hodnotami nahranými v komparačních registrech A (CMPA) a B (CMPB). Pokud dojde ke shodě, je vygenerována nastavená událost.

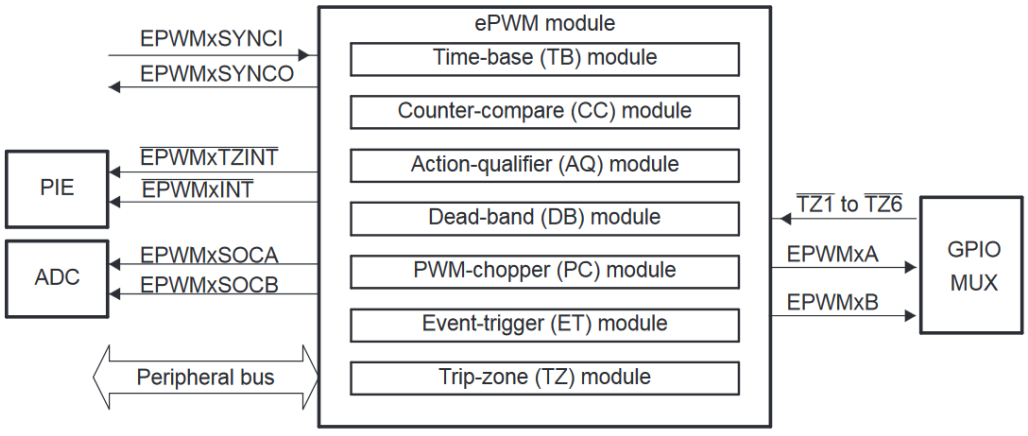
  Účel tohoto submodulu:
  - generace událostí v okamžicích daných hodnotami v CMPA a CMPB,
  - při vhodně nakonfigurovaném action-qualifier modulu kontroluje střídu PWM,
  - zajišťuje nahrání nové komparační hodnoty přes stínový registr (eliminuje hazard).

- **Action-qualifier submodul (AQ)** – jeho úkolem je:
  - generování akcí na příslušných pinech (vysoká úroveň, nízká úroveň, invertování logické úrovně) pokud time-base counter dosáhne své maximální hodnoty, nuly nebo pokud dojde ke shodě s hodnotami nahranými v CMPA a CMPB,
  - správa priority jestliže nastanou některé události současně,
  - umožňuje rozdílné nastavení událostí při shodě během čítání nahoru a dolů.

- **Dead-band submodul (DB)** umožnuje párování signálu na ePWMxB odvozené od signálu ePWMxA a zajišťuje generaci programovatelných mrtvých dob (dead-time). Dead-band submodul je umístěn za action-qualifier submodulem. DB submodul lze kompletně přemostit, pokud není v aplikaci potřeba.

- **PWM-Chopper submodul (PC)** umožňuje modulovat výstupní signál z AC a DB submodulů vysokofrekvenčním nosným signálem. Toho se v praxi využívá, pokud je driver polovodičového spínače založen na pulzním transformátoru. PC submodul lze kompletně přemostit, pokud není v aplikaci potřeba.

- **Trip-zone submodul (TZ)** zajišťuje předem naprogramovanou odezvu na poruchové signály. Každý ePWM module je připojen k šesti TZn (TZ1–TZ6) signálům. Tyto signály jsou přivedeny do TZ submodulu z GPIO multiplexoru a slouží k indikaci poruch mimo F28335.



Obrázek 4.4. ePWM submoduly [15]

4.2.5 Enhanced Quadrature Encoder Pulse (eQEP) module

Jednotka eQEP je určena pro zpracování signálu z inkrementálního enkodéru a v rámci vektorového řízení je použita k měření otáčivé rychlosti motoru. Mimo měření otáček, resp. frekvence lze eQEP použít pro vyhodnocování polohy hřídele.[16]

Modul eQEP podporuje několik digitálních vstupů – dva piny jsou vyhrazeny pro quadrature-clock mode (QEPA pro stopu A a QEPB pro stopu B) nebo direction-count mode (XCLK a XDIR), jeden pro index signál nebo indikátor „nulové“ polohy (QEPI) a jeden pro strobovací signál (QEPS).[16]

- *QEPA/XCLK a QEPB/XDIR*
  Tyto dva piny mohou být použity v quadrature-clock módu a direction-count módu.
  - *Quadrature-clock mode*
    Výstupem z inkrementálního enkodéru jsou dva obdélníkové signály se střídou 50 % (stopa A a stopa B) vzájemně posunuté o 90°. Kvadraturní dekodér na základě těchto signálů generuje tzv. kvadraturní hodiny (quadrature-clock) a ze sledu těcho signálů určuje směr otáčení. Relativní poloha hřídele, která je počítána od tzv. index pozice, je odvozena na základě napočítaných pulzů.
  - *Direction-count mode*
    Některá čidla polohy mají jako výstup přímo hodinový signál úměrný rychlosti otáčení hřídele a signál určující směr otáčení hřídele. QEPA pin pak slouží jako vstup pro hodinový signál a QEP pin jako vstup pro informaci o směru otáčení.

- *eQEPI: Index signál nebo indikátor nulové polohy*
  Enkodér jednotky eQEP využívá index signál k určení výchozí pozice, ze které se začínají čítat kvadraturní signály pro určení polohy. Signál na tomto pinu může sloužit k nulování čítače pulzů nebo naopak k jeho inicializaci či uložení jeho aktuální hodnoty na základě definované události.

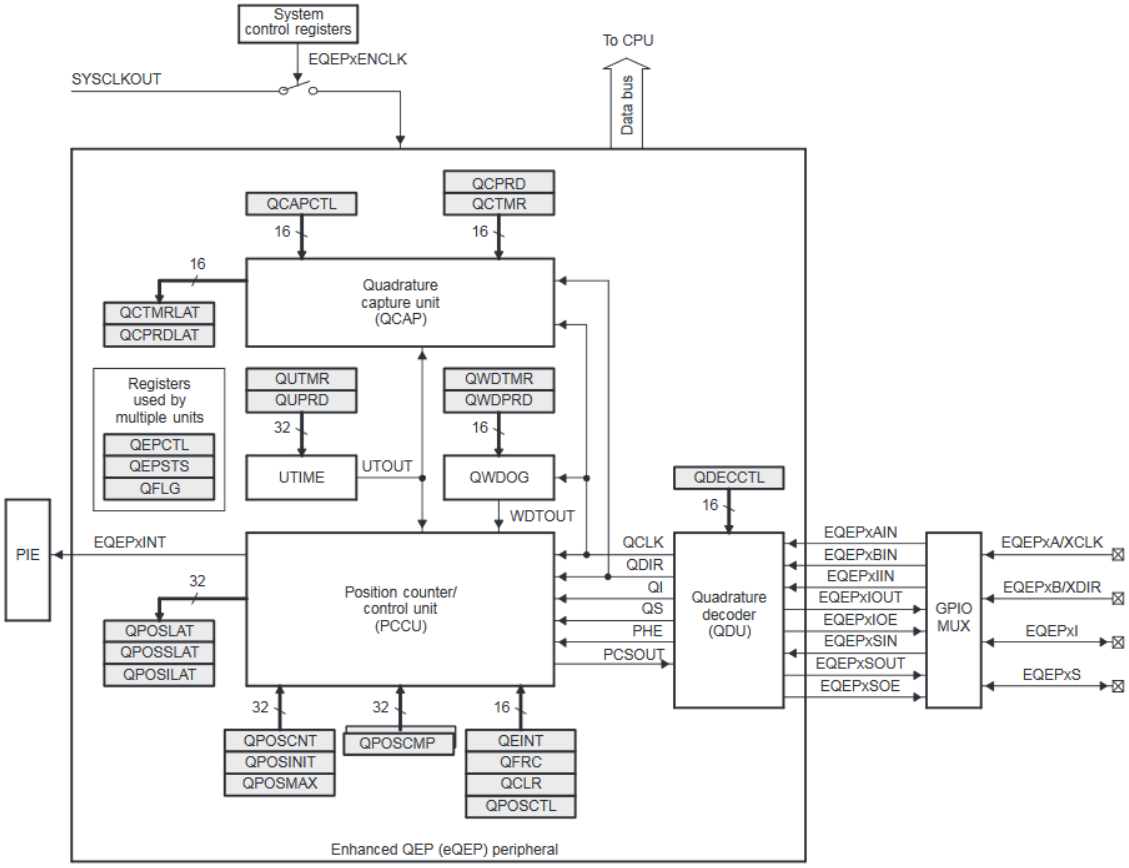
- *QEPS: Strobovací vstup*
  Strobovací vstup může na základě definované události na QEPS inicializovat čítač pulzů nebo uložit jeho aktuální hodnotu. Na tento vstup lze například přívést signál od koncového spínače, který dává informaci o dosažení požadované polohy.

Jednotka eQEP obsahuje tyto hlavní funkční bloky (obr. 4.5):[16]

- programovatelná kvalifikace vstupů pro každý pin (součást GPIO MUX – viz kap. 4.2.2),
- kvadraturní dekodér (QDU),
- čítač pulzů (resp. polohy) a řídicí jednotka pro měření polohy (PCCU),
- kvadraturní záchytná (capture) jednotka určená pro měření otáčivé rychlosti v oblasti nízkých otáček,
- časová základna (timer) pro měření frekvence/rychlosti (UTIME),
- watchdog timer pro detekování nulových otáček.

eQEP disponue vlastním 32bitovým timerem s programovatelnou předděličkou, jehož účel je generovat přerušení pro výpočet rychlosti. K vygenerování žádosti o přerušení (UTO) nastane v okamžiku shody hodnot timeru a registru periody (QUPRD). eQEP může být nakonfigurován tak, že při této události uloží hodnoty z některých registrů (position counter, capture timer a capture period) do svých tzv. „latch“ registrů.[16]

Důležitou funkcí kvadraturního dekodéru (QDU) je schopnost určit směr otáčení hřídele. Tato informace je uchovávána v bitu QEPSTS[QDF]. Na základě směru otáčení je pak inkrementován či dekrementován čítač polohy (QPOSCNT), který (v quadrature count módu) čítá každou vzestupnou i sestupnou hranu obou stop inkrementálního snímače. Registr QPOSCNT je 32bitový a neznaménkový. Pomocí příslušných řídicích registrů lze definovat, při jakých událostech bude QPOSCNT inicializován, či naopak resetován. eQEP podporuje celou řadu dalších konfigurací, kterými lze dosáhnout optimálního chování tohoto modulu v souladu s potřebami pro konkrétní aplikaci.[16]



Obrázek 4.5. Blokové schéma eQEP jednotky [16]

4.2.6 Analog-to-Digital Converter (ADC) Module

Analogově číslicový převodník je v rámci vektorového řízení používán k měření fázových proudů a stejnosměrného napětí v meziobvodu střídače. Převodník u F28335 je 12bitový se 16 kanály, které mohou být nakonfigurovány jako dva nezávislé 8kanálové moduly, a disponuje několika stupňovým předdělícím systémem, který umožňuje dosáhnout téměř libovolné časové základny. Blokové schéma analogově-číslicového převodníku je znázorněno na obr. 4.6.[17]

Charakteristika a funkce ADC modulu:[17]

- 12bitový převodník se dvěma vzorkovacími (S&H) obvody,
- sekvenční nebo simultánní převod,
- vstupní napěťový rozsah 0–3 V (tj. 3 V a více odpovídá maximální a 0 V a méně pak minimální číslicové hodnotě),
- analogové multiplexory podporující až 16 vstupů,
- možnost až 16 autokonverzí v autosekvenčního převodu – každá z konverzí může jako vstup využívat libovolný ze 16 vstupních kanálů,
- sekvencer může operovat jako dva nezávislé 8stavové sekvencery nebo jako jeden velký 16stavový sekvencer (tj. dva kaskádně spojené 8stavové sekvencery),
- 16 registrů pro uchování výsledků převodu,
- množství zdrojů pro spuštění převodní sekvence (S/W – softwarový okamžitý start, ePWM 1–6, GPIO XINT2),
- možnost vyvolání přerušení na konci každé nebo každé druhé převodní sekvence,
- pro nastavení doby trvání S&H okénka je k dispozici vlastní předdělička.

Jak již bylo výše uvedeno, ADC sekvencer se skládá z dvou nezávislých 8stavových sekvencerů (SEQ1 a SEQ2), které mohou být kaskádně spojeny do jednoho 16stavového sekvenceru. Přívlastek „8stavový“ (resp. „16stavový“) znamená počet autokonverzí, které mohou být v rámci sekvence provedeny. V obou případech je převodník schopen provést po zaregistrování požadavku na započetí konverze (SOC) automaticky sérii převodů. Jako zdroj signálu pro každou konverzi může být vybrán libovolný z 16 dostupných analogových vstupů. Po ukončení konverze je číslicová hodnota odpovídající napětí na příslušném vstupu uložena do jednoho z registrů určených pro uchování výsledku převodu (ADCRESULTn – první výsledek je uložen do registru ADCRESULT0, druhý do ADCRESULT1 atd.). Je rovněž možné převést vícekrát stejný kanál, což v některých případech může zvyšovat přesnost výsledné hodnoty.[17]

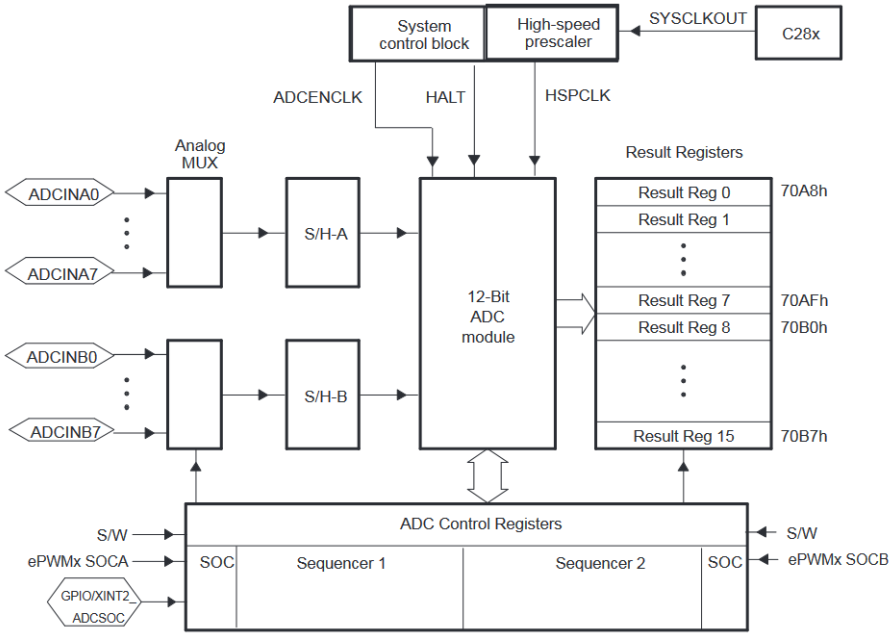
Analogově-číslicový převodník může pracovat rovněž v simultánním vzorkovacím módu (simultaneous sampling mode) nebo sekvenčním vzorkovacím módu (sequential sampling mode). Pro každou konverzi (resp. každý pár konverzí v simultánním vzorkovacím módu) je v bitovém poli CONVxx („xx“ značí číslo konverze) definována informace o tom, jaký vstupní kanál (rep. pár kanálů v simultánním vzorkovacím módu) má být navzorkován a převeden. CONVxx obsahuej 4 bity, které jsou v sekvenčním módu všechny využity pro výběr vstupního kanálu. MSB říká, který S&H buffer (S&H-A nebo S&H-B) bude vybrán a tři LSB definují offset. Například hodnota 0101b („b“ značí, že jde o číslo v binární soustavě) v CONVxx udává, že má být pro tuto konverzi vybrán vstup ADCINA5. Pokud je dále např. v CONVxx uložena hodnota 1011b, tak jako vstup bude vybrán ADCINB3. V simultánním módu je MSB ignorován a informaci o tom, jaký pár kanálů bude převeden, nesou tři LSB. Tak například, pokud je v registru CONVxx nahrána hodnota 0110b, je pomocí S&H-A vzorkován vstupní pin ADCINA6 a pomocí S&H-B pak ADCINB6 . Pokud tuto hodnotu změníme na 1001b, je přes S&H-A navzorkován vstup ADCINA1 a přes S&H-B vstup ADCINB1. Po odebrání vzorků z jednoho páru vstupů je nejdříve převedena hodnota napětí S&H-A a poté hodnota S&H-B. Výsledek odpovídající napětí na S&H-A je uložen do aktuálního registru ADCRESULTn, výsledek odpovídající napětí na S&H-B potom do registru následujícího.[17]

Další z možných módů, ve kterém může převodník pracovat, je tzv. nepřetržitý autosekvenční mód (uninterrupted autosequential mode). V tomto režimu mohou SEQ1 a SEQ2 automaticky převést až osm libovolných kanálů (případně 16 při kaskádním spojení) během jedné sekvence. Výsledky konverzí jsou v případě SEQ1 uloženy v registrech ADCRESULT0–ADCRESULT7 a v případě SEQ2 do ADCRESULT8–ADCRESULT15. Počet konverzí v jedné sekvenci se řídí bitovým polem MAX_CONVn z registru ADCMAXCONV. MAX_CONVn může obsahovat hodnotu od nuly do sedmi (případně od nuly do patnácti pri kaskádním spojení SEQ1 a SEQ2). Počet konverzí dokončených při jedné autosekvenci je roven číslu (MAX_CONVn + 1).[17]

Jakýkoliv sekvencer může dále operovat v tzv. Start/Stop módu, který je synchronizován k více časově doděleným SOC zdrojům. V tomto módu není v obsluze přerušení sekvencer po skončení jedné sekvence resetován do výchozího stavu (tj. CONV00). Jinými slovy při dokončení sekvence zůstává sekvencer v aktuálním stavu. Pro zakázání tohoto módu musí být bit CONT_RUN v ADCTRL1 registru nastaven do nuly.[17]

ADC modul musí být před užíváním zkalibrován. Kalibrační funkce ADC_cal() je nahrána od výrobce do vlastní OTP paměti. Boot ROM automaticky spouští tuto kalibrační funkci. Ta inicializuje registry ADCREFSEL a ADCOFFTRIM kalibračními údaji specifickými pro konkrétní zařízení. Tento proces za normálních okolností probíhá automaticky a není tedy potřeba zásah uživatele. Pokud ale během vývoje CCS

tuto Boot ROM nepoužívá, je nutné provést inicializaci registru ADCREFSEL a registru ADCOFFTRIM v rámci uživatelského kódu, což je detailněji popsáno v referenční příručce k ADC modulu.[17]



Obrázek 4.6. Blokové schéma analogově-číslicového převodníku [17]

4.3 Propojení eZdsp F28335 s počítačem, debugging

Propojení eZdsp a počítače s nainstalovaným CCS IDE bylo realizováno pomocí XDS100v3 USB JTAG emulátoru. XDS100v3 je v souladu s IEEE 1149.7 a slouží pro emulaci a debugging zařízení od Texas Instruments pomocí standardního 14pinového JTAG rozhraní. Funkcí zařízení je zejména zahájení/ukončení emulace, čtení a zápis do pamětí, čtení registrů; dále nahrání, spuštění, zastavení a krokování programu, podpora hardwarových a softwarových breakpointů a real-time mód (procesor není při čtení či zápisu dat přes emulátor pozastaven). XDS100v3 je kompatibilní s CCS IDE a připojení k nadřazenému počítači je realizováno pomocí USB.[18]

4.4 Propojení eZdsp F28335 s měničem a čidly

Propojení mezi eZdsp, měničem a čidly zajišťuje deska, která byla poskytnuta, doktorem Bauerem, vedoucím této práce. Toto fyzické rozhraní zajišťuje potřebnou konektivitu a napěťové přizpůsobení F28335 a k němu připojeného hardwaru.

4.5 Code Composer Studio

Kód pro F28335 byl psán v programu Code Composer Studio (CCS), což je integrované vývojové prostředí (IDE) určené pro vývoj aplikací pro mikrokontroléry a vestavěné procesory od Texas Instruments. Součástí IDE jsou nástroje pro vývoj a debugging – jmenovitě kompilátor optimalizovaný pro C/C++ kód, editor zdrojového kódu, prostředí pro překlad, debugger a mnoho dalšího.[19]

Veškeré relevantní materiály k CCS IDE včetně odkazu ke stažení lze nalézt na [19].

4.6 C2000Ware

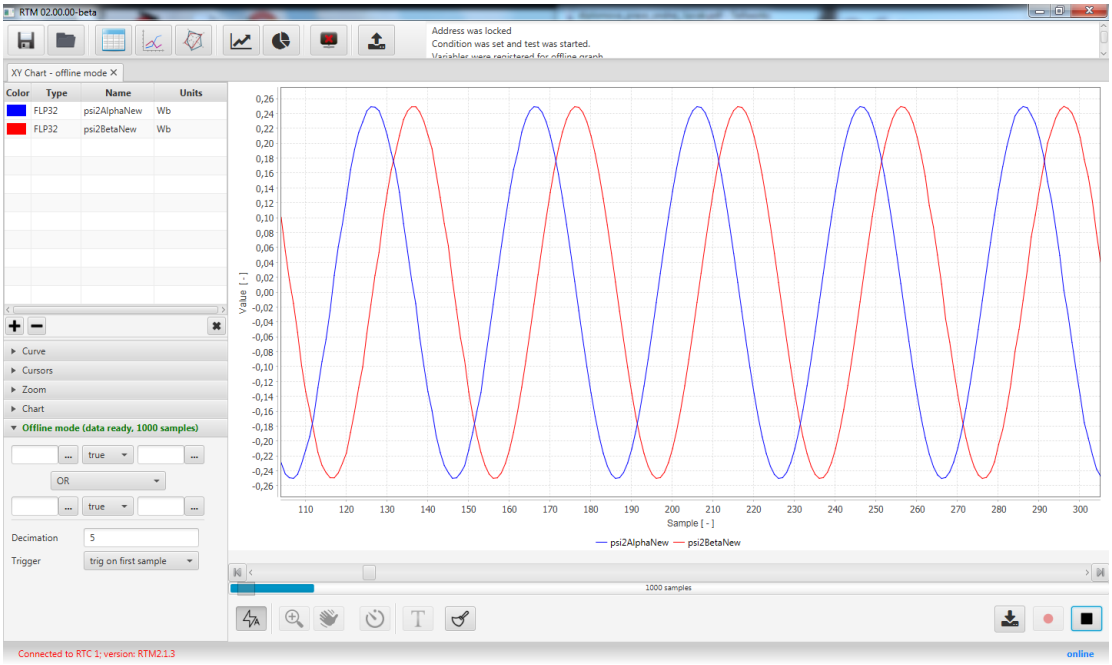
C2000Ware pro rodinu mikrokontrolérů C2000 je balíček, který obsahuje dodatečný software s dokumentací, který může uživateli pomoci s vývojem jeho programu a rovněž zkrátit čas pro tento vývoj nutný. C2000Ware obsahuje množství nástrojů pro usnadnění – od ovladačů a knihoven pro konkrétní zařízení až po různé příklady nastavení a použití periferií.[20]

Pro používání C2000Ware je nejdříve nutné C2000Ware nainstalovat. To je možné provést přes tzv. *Resource Explorer* v prostředí Code Composer Studio. Nejdříve spustíme CCS, poté otevřeme *Resource Explorer* (*View* → *Resource Explorer*), následně v levém stromu nabídky pod záložkou *Software* rozklikneme *C2000Ware*. V nově otevřeném okénku je možné kliknout na ikonku *Download and Install*, která tento balíček stáhne a posléze nainstaluje.

Některé součásti C2000Ware jsou dále využity během vektorového řízení.

4.7 Real-time monitor (RTM)

Ve spolupráci s katedrou elektrotechnologie je vyvíjen tzv. real-time monitor (RTM), který zprostředkovává vizualizaci interních proměnných signálového procesoru F28335. Součástí RTM jsou knihovny pro F28335 zajišťující komunikaci s PC a JAVA desktop aplikace pro samotnou vizualizaci. V RTM JAVA aplikaci lze monitorovat proměnné v offline i online módu. K dispozici je jednak standardní ortogonální graf a pak také graf polární. Pomocí RTM lze rovněž modifikovat proměnné uvnitř F28335. RTM byl použit pro debugging a ladění během vývoje vektorového řízení. Na jeho tvorbě se autor této diplomové práce nepodílel, avšak knihovny a funkce zajišťující komunikaci s osobním počítačem byly implementovány vedle kódu pro vektorové řízení.



Obrázek 4.7. Real-Time Monitor

Kapitola 5
Konfigurace a inicializace TMS320F28335

5.1 Spuštění a přenositelnost vytvořeného CCS projektu

Všechny cesty, které potřebuje compiler a linker pro vyhledání příslušných souborů, jsou rešeny jako relativní. Jediné, co musí uživatel udělat po rozbalení přiloženého souboru s projektem, aby mohl dále s projektem pracovat, je mít nainstalován Code Composer Studio (autor práce používal verzi 7.2.0.00013), C2000Ware a poté nastavit enviromentální proměnnou pro místo, kde je C2000Ware nainstalován.

Nejdříve nainstalujeme Code Composer Studio. Pak přidáme do CCS rozbalenou složku s projektem (*File* → *Import*, pod záložkou *Code Composer Studio* klikneme na *CCS Projects*, klikneme na *Next* a v dalším okně vyhledáme umístění rozbaleného kořenového adresáře projektu), poté nainstalujeme C2000Ware (viz kap. 4.6). Následně přidáme proměnnou pro adresář C2000Ware – v okně *Projects Explorer* klikneme pravým tlačítkem na importovaný projekt a vybereme *Properties*. Vyskočí nám nové okno – ze stromu nabídky vybereme *General* a následně v záložce *Products* zaškrtneme *C2000Ware*. Nyní by měly být všechny cesty pro compiler a linker známy a program by mělo být možné bez problému nahrát do zařízení a spustit.

5.2 Optimalizovaná knihovna pro matematické výpočty C28x Floating Point Unit fastRTS Library

V průběhu vektorového řízení je realizováno větší množství složitějších matematických operací využívajících např. goniometrické funkce. Pro urychlení výpočtů byla do CCS projektu zakomponována rychlá matematická knihovna fastRTS Library.

Texas Instruments TMS320C28x Floating Point Unit Fast Run-Time Support (RTS) Library obsahuje sadu matematických funkcí pro výpočty v plovoucí řádové čárce. Knihovna je určena pro zařízení s C28x FPU. Knihovna obsahuje C kompatibilní optimalizovanou verzí funkcí, které se nacházejí ve standardních knihovnách C kompilátoru. Použití těchto optimalizovaných funkcí je cíleno pro náročné aplikace pracující v reálném čase, kde lze dosáhnout výraznou úsporu výpočetního času. Všechny potřebné soubory včetně dokumentace s návodem pro zapracování knihovny do CCS projektu lze nalézt v rámci nainstalovaného C2000Ware v podadresáři `\libraries\math\FPUfastRTS\c28`.[21]

5.3 Organizace programu

Celý projekt je psán v jazyce C a je pro přehlednost a snadnější implementaci organizován modulárně. Funkční bloky vektorového řízení jsou rozděleny do jednotlivých hlavičkových souborů, které jsou následně vtahovány do hlavního zdrojového souboru.

Samostatný hlavičkový soubor je vytvořen také pro deklaraci konstant pomocí makra `#define` používaných globálně v projektu. Pro inicializaci řídicích periferií jsou vytvořeny samostané zdrojové (source) soubory s inicializačními funkcemi, jejichž prototypy jsou deklarovány v hlavním souboru.

Hlavičkové soubory (`.h`):

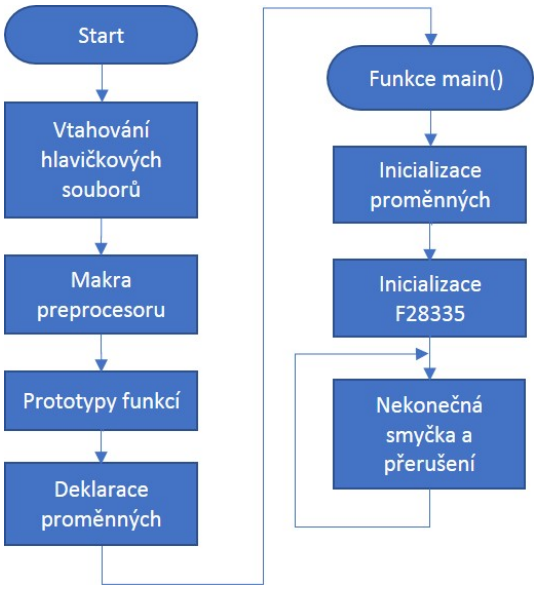
- Clarkové a Parkova transformace,
- globální konstanty,
- modulace prostorového vektoru,
- regulátory,
- model motoru,
- výpočet rychlosti otáčení.

Zdrojové soubory (`.c`):

- inicializace ePWM modulu,
- inicializace analogově–číslicového převodníku,
- inicializace eQEP jednotky.

5.4 Struktura hlavního zdrojového souboru

Na obr. 5.1 je znázorněna vývojovým diagramem struktura hlavního zdrojového souboru (`main.c`) projektu. Nejdříve jsou direktivami `#include` vtaženy hlavičkové soubory. Dále jsou pomocí makra `#define` definovány konstanty použité v hlavním souboru. Poté jsou uvedeny prototypy použitých funkcí včetně obsluh přerušení a deklarovány proměnné. Následuje obligátní funkce `main()`, ve které je nejdříve provedena inicializace všech deklarovaných proměnných a dále konfigurace periferií F28335 do požadovaného stavu. Následně je pomocí příkazu `while(1){}` vytvořena nekonečná smyčka (pozadí programu), ze které program periodicky odskakuje obsloužit žádosti o přerušení.



Obrázek 5.1. Struktura hlavního zdrojového souboru

5.5 EALLOW chráněné registry

Některé řídicí registry jsou chráněné tzv. EALLOW ochranným mechanismem proti falešným CPU zápisům. To, jestli je ochrana zapnuta či vypnuta, indikuje EALLOW bit v stavovém registru 1 (ST1). Tento bit je po resetu nastaven do nuly, tj. je povolen chráněný režim. V chráněném režimu jsou zakázány jakékoliv zápisy do EALLOW registrů, povoleno je pouze čtení. Pokud chceme v těchto registrech modifikovat bity, je nutné nejdříve příkazem `EALLOW` tento zápis umožnit a po provedených úpravách jej příkazem `EDIS` zakázat.[14]

Příklad takového zápisu:

```
EALLOW;
GpioCtrlRegs.GPAMUX2.bit.GPIO24=2;
EDIS;
```

5.6 Popis konfigurace F28335 a periferií

5.6.1 Obecná inicializace F28335

Nejzákladnější inicializace F28335 je již provedena v knihovnách autorů RTM. K tomuto účelu byly použity kromě vlastního kódu autorů také volně dostupné příklady konfigurací TMS320F28335 od společnosti Texas Instruments. Tyto příklady lze najít v již zmíněném balíčku C2000Ware (např. soubory *DSP2833x_SysCtrl.c, DSP2833x_PieCtrl.c, DSP2833x_Gpio.c, DSP2833x_PieVect.c*).

Nejzákladnější inicializace zahrnuje nastavení hodiny CPU na 150 MHz pomocí PLL, zakázaní WatchDog obvodů, povolení periferních hodin, nastavení GPIO do definovaného stavu, inicializace PIE řídicích registrů do definovaného stavu a inicializace tabulky vektorů přerušení.

5.6.2 Inicializace PWM

Popis konfigurace PWM je rozdělen dle jednotlivých submodulů:

- **Time-base submodul**

  Všechny submoduly uvnitř PWM modulu jsou časovány signálem TBCLK, který je odvozen od systémových hodin procesrou (SYSCLKOUT). TBCLK určuje rychlost inkrementace případně dekrementace time-base counteru. Hodiny TBCLK jsou vypočteny následujícím způsobem:[15]

$$\mathrm{TBCLK} = \frac{\mathrm{SYSCLKOUT}}{\mathrm{HSPCLKDIV} \cdot \mathrm{CLKDIV}}, \tag{5.1}$$

  kde TBCLK je frekvence hodin time-base counteru v Hz, SYSCLKOUT je frekvence hodin CPU v Hz a HSPCLKDIV a CLKDIV jsou předdělící poměry, které lze nastavit v registru TBCTL. Frekvence systémových hodin SYSCLKOUT je nastavena na 150 MHz, předdělící poměr CLKDIV je zvolen jako 1 a HSPCLKDIV jako 2. Po dosazení těchto hodnot do vztahu (5.1) vychází frekvence TBCLK 75 MHz.[15]

  Frekvence, resp. perioda PWM se řídí zvoleným módem čítání a hodnotou v registru TBPRD, která určuje maximální hodnotu pro time-base counter. Je zvolen tzv.

up-down-count mód, ve kterém time-base counter čítá od nuly do své maximální hodnoty (TBPRD) a poté z maximální hodnoty zpět do nuly. Pro výslednou frekvenci PWM v up-down-count módu platí následující vztah:[15]

$$f_{\mathrm{PWM}} = \frac{1}{2} \cdot \frac{f_{\mathrm{TBCLK}}}{\mathrm{TBPRD}}. \tag{5.2}$$

Pro požadovanou $f_{\mathrm{PWM}} = 10$ kHz lze ze vztahu (5.2) vypočítat hodnotu, kterou je nutné nahrát do registru TBPRD. Ta po dosazení a vyjádření vychází 3 750.

Pro vytvoření napájení pomocí třífázového střídače je nutné použít tři ePWMx moduly. Každý modul ovládá dvěma svými výstupy horní a spodní tranzistor v jedné větvi. Aby se docílilo vygenerování požadovaného vektoru napětí pomocí PWM je nutné, aby čítače jednotlivých ePWMx modulů byly mezi sebou synchronizovány. Každý ePWMx modul má svůj synchronizační vstup a výstup (pro ePWM1 je to externí signál), které lze vzájemně propojit. Konfigurace je provedena tak, že ePWM1 vygeneruje synchronizační signál EPWM1SYNCO, když time-base counter dosáhne nulové hodnoty. Událost, při které dojde k vygenerování synchronizačního signálu, řídí bitové pole SYNCOSEL registru TBCTL. EPWM1SYNCO je zároveň vstupem pro synchronizační signál ePWM2 (EPWM2SYNCI), který dále generuje výstup (EPWM2SYNCO) pro synchronizační signál ePWM3 (EPWM3SYNCI). Modul ePWM1 se tak chová jako „master“ a moduly ePWM1 a ePWM2 jako „slave“. V momentě, kdy je vygenerován synchronizační signál lze u jednotlivých modulů bitem PHSEN stanovit, zda bude do TBCTR nahrána hodnota z registru TBPHS, která řídí fázový posuv jednotlivých PWM signálů. Pro vektorové řízení je požadováno, aby všechny PWM výstupy byly ve fázi. Aby byla do TBCTR modulů ePWM2 a ePWM3 nahrána nulová hodnota (tj. aby byl zajištěn nulový fázový posuv) je nutné nastavit u každého modulu bit PHSEN a do TBPHS nahrát 0.[15]

- **Counter-Compare submodul**
  U CC submodulu není provedena žádná zvláštní konfigurace. Hodnoty v řídicím registru CPMCTL po resetu již zajistí požadované vlastnosti, tj. nahrání nové hodnoty do compare registrů přes pomocné stínové registry v momentě, kdy je TBCTR roven nule. Jak již bylo zmíněno v kap. 4.2.4, každý CC submodul obsahuje dva compare registry (CMPA a CMPB), které generují události pro action-qualifier submodul. Díky konfiguraci AQ a DB submodulů (viz dále) je však pro generování požadovaného PWM průběhu použit pouze registr CMPA.[15]

- **Action-Qualifier submodul**
  V registrech AQCTLA a AQCTLB lze pro oba dva PWM výstupy nastavit akci na příslušném pinu (vysoká úroveň, nízká úroveň, invertování logické úrovně) a událost, při níž má k této akci dojít. Výstup „B“ je pomocí DB modulu nastaven jako komplementární, tudíž je konfigurován pouze registr AQCTLA a to tak, že při shodě CMPA a TBCTR při čítání nahoru dojde k nastavení výstupu „A“ do vysoké úrovně a při shodě během čítání dolů naopak do úrovně nízké.[15]

- **Dead-Band submodul**
  V registru DBCTL je nastaven tzv. active high complementary (AHC) mód. To znamená, že výstup EPWMxB je vůči výstupu EPWMxA invertován. Dále je plně povoleno mrtvé pásmo (dead-band) pro náběžnou hranu na výstupu EPWMxA a pro sestupnou hranu na EPWMxB. Nakonec je ponecháno výchozí nastavení EPWMxA jako zdroje pro zpoždění náběžné a sestupné hrany na EPWMxB. Dead-time lze

nastavit zvlášť pro náběžnou hranu (registr DBRED) a sestupnou hranu (registr DBFED). Hodnoty v registrech DBRED a DBFED jsou převedny na časové zpoždění následujícím způsobem:[15]

$$\mathrm{FED} = \mathrm{DBFED} \cdot T_{\mathrm{TBCLK}}, \tag{5.3}$$

$$\mathrm{RED} = \mathrm{DBRED} \cdot T_{\mathrm{TBCLK}}, \tag{5.4}$$

kde T_{TBCLK} je perioda TBCLK v s a FED a RED je zpoždění náběžné a sestupné hrany rovněž v s. Hodnoty DBRED a DBFED jsou zvoleny jako 110. Dosazením do vztahů (5.3) a (5.4) vychází zpoždění náběžné a sestupné hrany (dead-time) zhruba 1,5 μs.

- **PWM-Chopper submodul**
  Není ve vektorovém řízení použit.

- **Trip-Zone submodul**
  Rovněž není ve vektorovém řízení použit. Do budoucna je však předpokládána implementace zpracování poruchových signálů z měniče.

- **Event-Trigger submodul**
  Proud je potřeba vzorkovat uprostřed okamžiku, kdy je na motor aplikováno nulové napětí (zdůvodnění viz kap. 6.1). Ten nastává buď když TBCTR = 0, nebo TBCTR = TBPRD. Je tedy nutné nějakým způsobem synchronizovat AD převodník s PWM. To umožňuje právě ET submodul. V registru ETSEL je bitem SOCAEN povolen signál započetí konverze (EPWM1SOCA). Dále je specifikováno v bitovém poli SOCASEL, že EPWM1SOCA má být vygenerován, když TBCTR = TBPRD.

Výsledná společná konfigurace pro všechny ePWMx moduly (nutno změnit číslo modulu pro ePWM1 a ePWM2):

```
EPwm1Regs.TBCTL.bit.PRDLD=0;
EPwm1Regs.TBCTL.bit.CLKDIV=0;
EPwm1Regs.TBCTL.bit.HSPCLKDIV=1;
EPwm1Regs.TBCTL.bit.CTRMODE=2;
EPwm1Regs.TBPRD=PWM_TBPRD;
EPwm1Regs.CMPCTL.bit.SHDWAMODE=0;
EPwm1Regs.CMPCTL.bit.LOADAMODE=0;
EPwm1Regs.AQCTLA.bit.CAU=2;
EPwm1Regs.AQCTLA.bit.CAD=1;
EPwm1Regs.DBCTL.bit.OUT_MODE=3;
EPwm1Regs.DBCTL.bit.POLSEL=2;
EPwm1Regs.DBCTL.bit.IN_MODE=0;
EPwm1Regs.DBRED=RISING_EDGE_DELAY;
EPwm1Regs.DBFED=FALLING_EDGE_DELAY;
```

Dodatečná konfigurace pro ePWM1:

```
EPwm1Regs.ETSEL.bit.SOCAEN=1;
EPwm1Regs.ETSEL.bit.SOCASEL=2;
EPwm1Regs.ETPS.bit.SOCAPRD = 1;
EPwm1Regs.TBCTL.bit.PHSEN=0;
EPwm1Regs.TBCTL.bit.SYNCOSEL=1;
```

Dodatečná konfigurace pro ePWM2:

```
EPwm2Regs.TBPHS.half.TBPHS=0;
EPwm2Regs.TBCTL.bit.PHSEN=1;
EPwm2Regs.TBCTL.bit.SYNCOSEL=0;
```

Dodatečná konfigurace pro ePWM3:

```
EPwm3Regs.TBPHS.half.TBPHS=0;
EPwm3Regs.TBCTL.bit.PHSEN=1;
```

Zde je ještě pro úplnost ukázáno, jak povolit na příslušné piny ePWM1 výstup:

```
EALLOW;
GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 1;
GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 1;
EDIS;
```

A zde jak ePWM1 „vypnout“ a nastavit piny do nízké úrovně:

```
GpioDataRegs.GPACLEAR.bit.GPIO4=1;
GpioDataRegs.GPACLEAR.bit.GPIO5=1;
EALLOW;
GpioCtrlRegs.GPADIR.bit.GPIO4=1;
GpioCtrlRegs.GPADIR.bit.GPIO5=1;
GpioCtrlRegs.GPAMUX1.bit.GPIO4=0;
GpioCtrlRegs.GPAMUX1.bit.GPIO5=0;
EDIS;
```

5.6.3 Inicializace ADC

Pro konfiguraci ADC slouží tři řídicí registry – ADCTRL1, ADCTRL2 a ADCTRL3. Je zvolen kaskádní mód, to znamená, že k dispozici je jeden 16stavový sekvencer. V minulé kapitole bylo zmíněno, že převodní sekvence ADC má být spuštěna signálem vygenerovaným ePWM1 modulem a to v okamžiku, kdy time-base counter dosáhne své maximální hodnoty (TBCTR = TBPRD). Tato možnost však musí být dále povolena i uvnitř ADC. Je tedy nutné specifikovat, že ADC má operovat v start/stop módu, kdy po ukončení sekvence ADC čeká na další spuštění. Důležitým aspektem při vzorkování je šířka S&H okénka, tj. doba, po kterou je příslušný analogový vstup spínačem připojen na vzorkovací kondenzátor. Jelikož je proud rychle měnící se veličina, která navíc kvůli pulsně-šířkové modulaci obsahuje mnoho vyšších harmonických, tak je účelné, aby tato doba byla co nejmenší. Na druhou stranu musí ale umožnit plné nabití vzorkovacího kondenzátoru. Zvolena je délka tří period hodin ADCLK.

Napěťové signály odpovídající fázovým proudům jsou přivedeny na analogové vstupy ADCA0, ADCB0 a ADCA1. Signál odpovídající napětí na kondenzátorech ve stejnosměrném meziobvodu pak na vstup ADCA2. Je tedy nutné provést tři páry konverzí v jedné sekvenci. Dále je nastaveno, že je vždy simultánně vzorkován jeden pár kanálů, nejdříve ADCA0 a ADCB0, pak ADCA1 a ADCB1 a nakonec ADCA2 a ADCB2. To, který pár kanálů bude v určité konverzi vzorkován, je nastaveno pomocí registru ADCCHSELSEQ1 (ADCA0 a ADCB0 v konverzi CONV00, ADCA1 a ADCB1 v konverzi CONV01, ADCA2 a ADCB2 v konverzi CONV02). Bitovým polem MAX_CONVn registru ADCMAXCONV je pak specifikováno, že maximální počet konverzí je 3.

Je požadováno, aby po skončení převodní sekvence došlo k vyvolání přerušení, na jehož začátku jsou do proměnných uloženy výsledky převodů z registrů ADCRESULT0,

ADCRESULT1, ADCRESULT2 a ADCRESULT4. Výsledky jsou čteny z tzv. mirror registrů (např. AdcMirror.ADCRESULT0), neboť jsou v nich výsledky zarovnány doprava a není tedy při čtení nutné provádět bitový posuv. V přerušení od AD převodníku pak probíhá po uložení výsledků celý algoritmus vektorového řízení.

Hodiny AD převodníku ADCLK jsou odvozeny od periferních hodin HSPCLK. Bitovým polem ADCCLKPS v registru ADCTRL3 lze vybrat dělicí poměr, který lze dále dvakrát zvýšit bitem CPS v registru ADCTRL1. HSPCLK je roven 75 MHz, ADCCLKPS = 3 a CPS = 0. Z těchto hodnot vychází celkový dělicí poměr, kterým jsou děleny hodiny HSPCLK, jako 6. Výsledná frekvence ADCLK je tedy 12,5 MHz. Reference pro AD převodník je ponechána vnitřní – z toho důvodu je pak v souladu s doporučením od výrobce pin ADCREFIN připojen na analogovou zem.

Mimo konfigurační funkci vytvořenou autorem práce je dále při inicializaci použita funkce `InitAdc()` z balíčku C2000Ware od Texas Instruments, která se nachází v souboru DSP2833x_Adc. Ta zajišťuje povolení hodin ADCLK, volání assemblerovské kalibrační funkce `ADC_cal()`, spuštění AD převodníku včetně interní napěťové reference a nakonec vložení 5ms časového zpoždění nutného pro ustálení analogových obvodů před prvním převodem.

Vytvořený kód pro konfiguraci ADC:

```
AdcRegs.ADCTRL1.bit.ACQ_PS=3;
AdcRegs.ADCTRL1.bit.CPS=0;
AdcRegs.ADCTRL1.bit.CONT_RUN=0;
AdcRegs.ADCTRL1.bit.SEQ_CASC=1;
AdcRegs.ADCTRL2.bit.EPWM_SOCA_SEQ1=1;
AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1=1;
AdcRegs.ADCTRL2.bit.RST_SEQ1=1;
AdcRegs.ADCTRL3.bit.ADCCLKPS=3;
AdcRegs.ADCTRL3.bit.SMODE_SEL=1;
AdcRegs.ADCCHSELSEQ1.bit.CONV00=0;
AdcRegs.ADCCHSELSEQ1.bit.CONV01=1;
AdcRegs.ADCCHSELSEQ1.bit.CONV02=2;
AdcRegs.ADCMAXCONV.bit.MAX_CONV1=2;
```

5.6.4 Inicializace eQEP

Pro měření otáčivé rychlosti je použita metoda, která je kombinací počítání pulzů ve fixní periodě (dobrá přesnost při vyšších rychlostech) a měření časového intervalu mezi dvěma po sobě jdoucími hranami (dobrá přesnost při nižších rychlostech). Samotná metoda společně s výpočtem otáček je vysvětlena v kap. 6.3. Zde jsou uvedeny pouze informace relevantní ke konfiguraci eQEP jednotky.

Pro možnost čítat pulzy enkodéru je nutné povolit eQEP position counter. Dále je specifikováno, že position counter má být resetován v okamžiku, kdy dosáhne své maximální hodnoty. Tu je možné nastavit, a to v registru QPOSMAX, který je 32bitový. Maximální hodnota je zvolena stejná, jako je maximální zobrazitelná hodnota v registru position counteru QPOSCNT (0xFFFFFF). Registr QPOSCNT je možné inicializovat počáteční hodnotou, která je zvolena jako 0.

Vzhledem k typu metody měření otáček je nutné rovněž použít eQEP timer, který bude vyvolávat přerušení v přesně definovaném okamžiku. Hodiny timeru jsou odvozeny od hodin SYSCLKOUT, dělicí poměr je zvolen 1. V registru QUPRD, který specifikuje periodu eQEP timeru je nahrána taková hodnota (150 000), aby docházelo k vyvolání přerušení každou 1 ms. Je nastaveno, že při shodě timeru (QUTMR) a registru QUPRD mají být hodnoty z position counteru a caption timeru přesunuty do svých latch registrů,

ze kterých je posléze možné tyto hodnoty přečíst a dále s nimi pracovat. Při této shodě je rovněž vyvoláno přerušení, ve kterém probíhá samotný výpočet otáček. Dále je povolena capture jednotka a je nastaveno, že předdělička čítaných hran má být 1. Watchdog timer jednotky eQEP pro detekci nulových otáček není použit a je tedy zakázán.

F28335 obsahuje dvě eQEP jednotky. Použita je eQEP2. Výsledný kód pro konfiguraci eQEP včetně povolení příslušných vstupů a hodin:

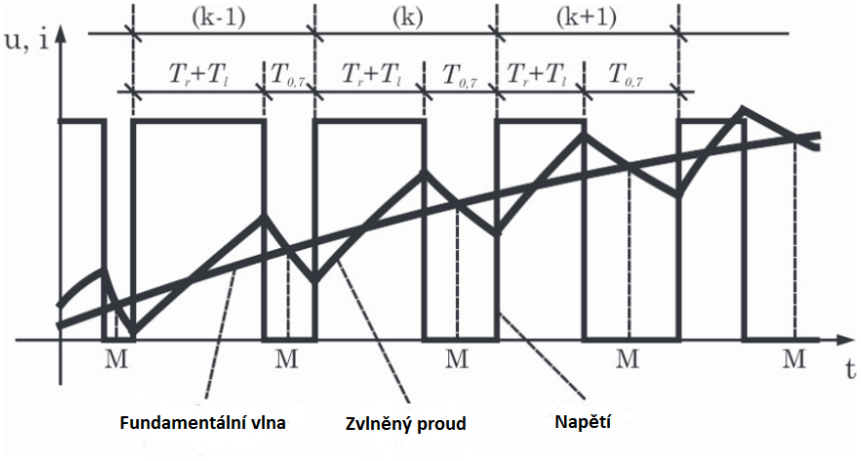
```
EALLOW;
GpioCtrlRegs.GPAMUX2.bit.GPIO24=2;
GpioCtrlRegs.GPAMUX2.bit.GPIO25=2;
EDIS;

EQep2Regs.QEPCTL.bit.FREE_SOFT=3;
EQep2Regs.QEPCTL.bit.PCRM=1;
EQep2Regs.QEPCTL.bit.QPEN=1;
EQep2Regs.QEPCTL.bit.QCLM=1;
EQep2Regs.QEPCTL.bit.UTE=1;
EQep2Regs.QEPCTL.bit.WDE=0;
EQep2Regs.QCAPCTL.bit.CEN=1;
EQep2Regs.QCAPCTL.bit.CCPS=0;
EQep2Regs.QCAPCTL.bit.UPPS=0;
EQep2Regs.QPOSINIT=0;
EQep2Regs.QEPCTL.bit.SWI=1;
EQep2Regs.QEPCTL.bit.SWI=0;
EQep2Regs.QPOSMAX=0xFFFFFFFF;
EQep2Regs.QUPRD=QUPRD_VAl;
EQep2Regs.QEINT.bit.UTO=1;

EALLOW;
SysCtrlRegs.PCLKCR1.bit.EQEP2ENCLK=1;
EDIS;
```

Kapitola 6
Zpracování měřených veličin

6.1 Měření proudu

Celý algoritmus vektorového řízení bude fungovat jedině v případě, že změříme fázové proudy motoru s určitou minimální přesností. Jejich znalost je nutná prakticky pro všechny části vektorového řízení. Mimo to je využita např. u softwarových proudových ochran. Z uvedeného plyne, že je nutné věnovat výběru a návrhu hardwaru i softwaru pro měření proudu náležitou pozornost.

V předchozích kapitolách bylo zmíněno, že je potřeba vzorkovat proud v okamžiku, kdy PWM time-base counter dosáhne své maximální hodnoty, tj. v okamžiku, kdy je na motor připojeno nulové napětí. Tento požadavek, který plyne z obr. 6.1, je zde zpětně zdůvodněn. Proud obsahuje vlivem PWM mimo základní harmonickou i harmonické vyšší. Jestliže však odebereme vzorek proudu uprostřed pulzu, kdy je fázové napětí nulové, získáme v idealizovaném případě informaci pouze o fundamentální vlně a zamezíme tak aliasingu.[7]



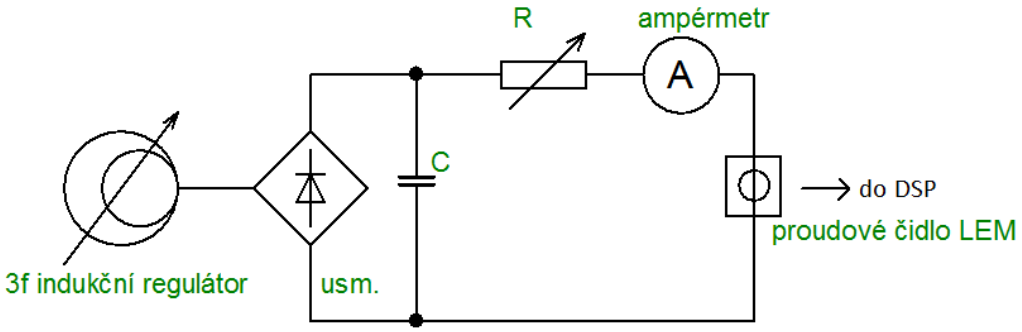
Obrázek 6.1. Vysvětlení okamžiku vzorkování proudu [7] (upraveno)

6.1.1 Kalibrace převodní konstanty proudového čidla

Pro správný výpočet hodnoty proudu je kalibrací nutné zjistit proudovou převodní konstantu navzorkovaného napětí na skutečný proud motoru. K tomuto účelu bylo využito schéma zapojení, které je vyznačeno na obr. 6.2.

Pomocí indukčního regulátoru a reostatu byl nastavován proud obvodem, a to od -25 A do 25 A s krokem 5 A. Pro F28335 byl vytvořen jednoduchý program, který na pokyn v daném okamžiku (zadávaný změnou proměnné přes debugger) začal v každém přerušení AD převodníku (tj. s periodou 100 μs) plnit pole 1 000 vzorky odpovídajícími měřenému proudu. Po jeho naplnění byl ze všech změřených hodnot v programu vypočítán aritmetický průměr, který byl následně zaznamenán. Závislost proudu na

aritmetickém průměru navzorkovaných hodnot byla vynesena do grafu a proložena pomocí metody nejmenších čtverců přímkou. Směrnice takto vzniklé přímky pak odpovídá rovnou převodní konstantě. Tento postup byl aplikován na oba dva měřené fázové proudy.



Obrázek 6.2. Schéma zapojení pro kalibraci proudového čidla

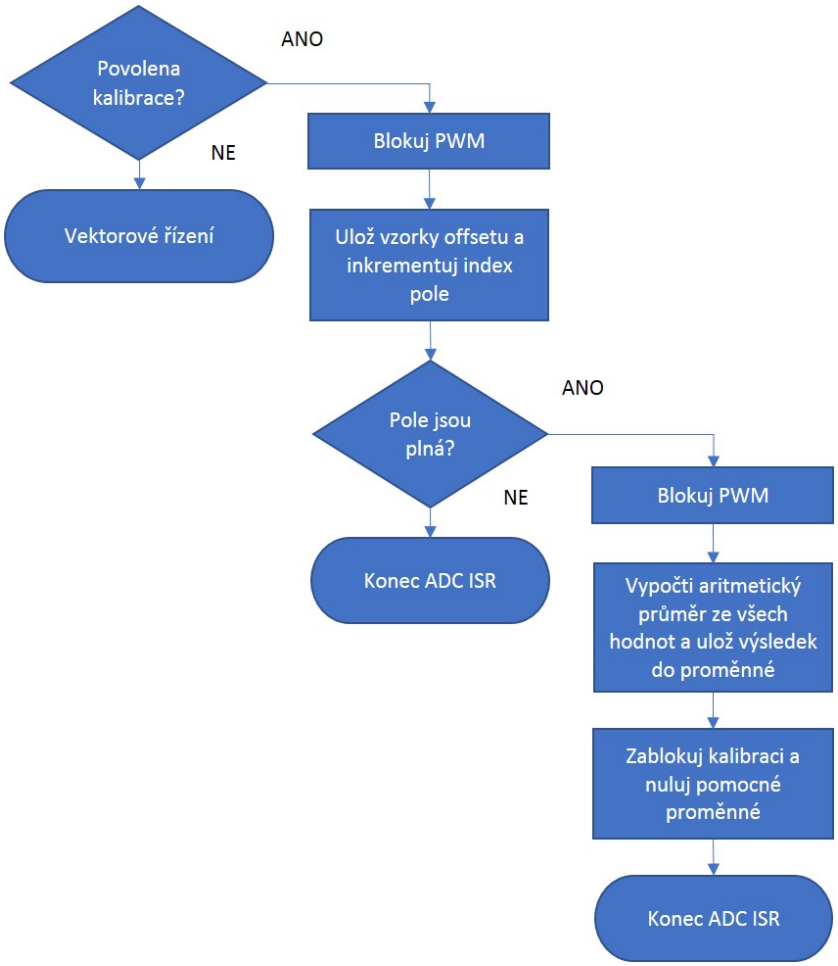
6.1.2 Odstranění offsetu přizpůsobovacích obvodů proudových čidel

Měřený proud dosahuje obou polarit, tudíž i napětí odpovídající měřenému proudu bude oboupolaritní. AD převodník dokáže ale zpracovat pouze kladný napěťový signál. Proto je nutné signál odpovídající měřenému proudu podložit kladnou stejnosměrnou složkou o velikosti zhruba poloviny vstupního rozsahu AD převodníku (1,5 V). To je zajištěno pomocí obvodů s operačními zesilovači na propojovací desce. Jak je ale všeobecně známo, analogové obvody vykazují mimo teplotní závislost svých parametrů a drift. Z uvedeného plyne, že generovaná stejnosměrná složka zajišťující podložení oboupolartiního signálu nebude konstatní. Tento problém je navíc umocněn jednak tím, že poměrně velkému rozsahu měřených proudů (−40 až +40 A) odpovídá poměrně malý rozsah signálu přiváděného na AD převodník (0 – 3 V) a jednak tím, že na tuto stejnosměrnou složku je ještě superponován šum. Proto je nutné před začátkem vektorového řízení stanovit offset a z něho následně vypočítat číslo, které se bude odečítat od převedené hodnoty proudu.

Po konfiguraci a inicializaci F28335 je provedena kalibrace offsetu obvodů proudových čidel. Ta probíhá tím způsobem, že každou vzorkovací periodu jsou do dvou připravených polí ukládány vzorky odpovídající napěťovému offsetu se superponovaným šumem. Po naplnění polí 1 000 vzorky je z těchto spočítán aritmetický průměr. Tato hodnota je poté použita při výpočtu skutečného proudu motoru. Aby měření bylo validní, je nutné zajistit, aby motorem neprotékal žádný proud. Před začátkem měření jsou zablokovány PWM výstupy a samotný výpočet algoritmu vektorového řízení a rovněž je zajištěno, že je uživatel až do skončení výpočtu nemůže povolit. Kalibrace probíhá pouze po spuštění programu a není ji již možné poté spustit znovu. Na obr. 6.3 je znázorněn vývojový diagram kalibrace offsetu.

6.1.3 Výpočet hodnoty proudu

V předchozích dvou kapitolách byl popsán postup, kterým byly získány převodní konstanty výstupního napětí z proudových čidel na skutečný proud a způsob kalibrace offsetu. Ná základně znalosti těchto konstant lze z čísla, které odpovídá převedené hodnotě napětí na vstupu AD převodníku, získat podle následujícího vztahu skutečnou



Obrázek 6.3. Vývojový diagram kalibrace offsetů z proudových čidel

okamžitou hodnotu fázového proudu motoru:

$$i = (\text{ADCRESULTn} - \text{OFFSET}) \cdot C_{\text{current}}, \tag{6.1}$$

kde i je výsledná hodnota proudu v A, ADCRESULTn je příslušný registr s výsledkem převodu, OFFSET je hodnota odpovídající vypočtenému offsetu a C_{current} je zjištěná převodní konstanta na skutečný proud, která má rozměr A.

6.2 Měření napětí

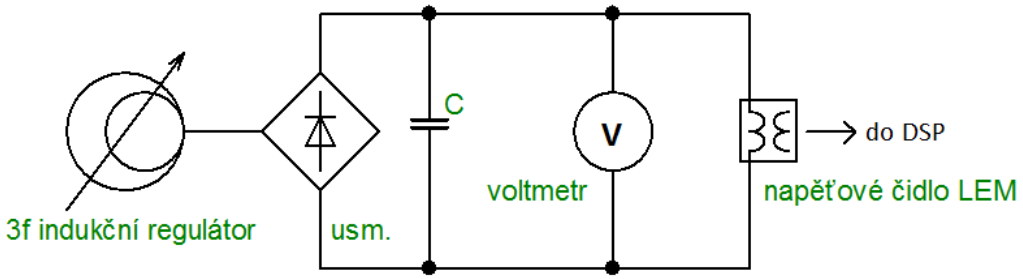
V rámci implementovaného vektorového řízení je měřeno napětí ve stejnosměrném meziobvodu třífázového střídače. Znalost tohoto napětí není pro řízení tak esenciální, jako je tomu v případě fázových proudů, nicméně má své opodstatnění. Hlavní využití je při ochraně samotného střídače při brzdění a reverzaci pohonu, kdy motor přechází do generátorického chodu a zvyšuje tak napětí na filtračním kondenzátoru. Dále je pak hodnota stejnosměrného napětí využita pro omezení žádaného napěťového vektoru, který je výstupem z proudových regulátorů.

6.2.1 Kalibrace převodní konstanty napěťového čidla

Postup kalibrace převodní konstanty napěťového čidla je analogický tomu u proudového čidla. Na obr. 6.4 je znázorněno schéma zapojení, které bylo ke kalibraci využito. Indukčním regulátorem bylo nastavováno stejnosměrné napětí na kondenzátoru

za usměrňovačem. Měření probíhalo od 0 do 600 V s krokem 50 V. V každém kroku bylo odečteno napětí na voltmetru a následně byl na F28335 spuštěn (přes debugger) stejný program jako u kalibrace proudového čidla, který uložil do předpřipraveného pole 1 000 vzorků napětí. Z těchto byl následně spočítán aritmetický průměr a tato číselná hodnota byla zapsána spolu s hodnotou napětí na voltmetru.

Závislost napětí na aritmetickém průměru navzorkovaných hodnot byla vynesena do grafu a proložena pomocí metody nejmenších čtverců přímkou. Směrnice takto vzniklé přímky pak odpovídá, stejně jako u proudových čidel, rovnou převodní konstantě.



Obrázek 6.4. Schéma zapojení pro kalibraci napěťového čidla

6.2.2 Odstranění offsetu přizpůsobovacích obvodů napěťových čidel

Signál z napěťového čidla je rovněž podložen stejnosměrnou složkou odpovídající zhruba polovině rozsahu AD převodníku, i když to v tomto případě není nutné, neboť napětí ve stejnosměrném meziobvodu střídače může dosahovat pouze kladných hodnot. Kalibrace offsetu ale není prováděna při každém spuštění programu jako u proudových čidel, protože na rozdíl od proudových obvodů, které lze zablokováním PWM vypnout, se může na kondenzátoru nacházet zbytkové napětí, které by mohlo do měření vnést nezanedbatelnou chybu. Navíc znalost napětí není požadována s takovou přesností, jako je tomu u proudů.

Offset přizpůsobovacích obvodů napěťového čidla je odečten z lineární převodní charakteristiky hodnot AD převodníku na napětí, jejíž stanovení bylo popsáno v předchozí kapitole. Zjištěná hodnota offsetu je v programu brána jako konstantní.

6.2.3 Výpočet hodnoty napětí

Napětí lze vypočítat naprosto analogicky jako proud podle vztahu (6.1). Místo offsetu pro proud dosadíme offset pro napětí, místo převodní konstanty proudu převodní konstantu napětí a místo výsledné hodnoty AD převodníku ze vstupů proudových přizpůsobovacích obvodů pak výslednou hodnotu AD převodníku ze vstupu napěťových přizpůsobovacích obvodů.

Za výpočtem hodnoty napětí je realizováno omezení pouze na nezáporné hodnoty napětí, neboť pokud není kondenzátor nabit, tak vlivem šumu dostáváme i nesmyslné malé záporné hodnoty napětí. Tato úprava je čistě kosmetická a nemá na fungování dalších programových bloků žádný vliv.

6.3 Měření otáček

Znalost otáčivé rychlosti motoru je společně se znalostí dvou fázových proudů nutná pro výpočet složek rotorového magnetického toku v souřadnicích svázaných se statorem

$\alpha\beta$. Tento výpočet probíhá řešením diferenciálních rovnic proudového I-n modelu. Dále na základě otáček probíhá v několika dalších blocích řízení různé blokování, nulování či nastavování proměnných. V neposlední řadě jsou vstupem pro otáčkový regulátor a otáčkovou ochranu. Správná znalost otáčivé rychlosti hřídele je tedy pro implementované vektorové řízení naprosto zásadní. Funkce se svým prototypem pro výpočet otáček je společně s deklarací pomocných proměnných použitých při výpočtu umístěna v samostatném hlavičkovém souboru s názvem `speed_calc.h`. Pomocí klíčového slova `extern` jsou zde také deklarovány proměnné založené v hlavním zdrojovém souboru nesoucí informaci o aktuálních otáčkách.

6.3.1 Metoda pro výpočet otáček

Existují dva základní přístupy pro zjišťování otáček motoru pomocí kvadraturního enkodéru. Prvním je měření počtu pulsů enkodéru ve fixním časovém intervalu. Přesnost této metody je nepřímo úměrně závislá na rozlišení (počet pulzů na otáčku) enkodéru a na délce časového intervalu, ve kterém se pulzy počítají. Tato metoda je někdy označována jako „M“ metoda a hodí se zejména pro oblast vyšších otáček, neboť při nízkých otáčkách počet pulzů detekovaných ve vzorkovacím intervalu klesá a v některých úsecích může být i nulový.[16, 22]

Druhým ze základních přístupů je měřit časový interval mezi dvěma po sobě následujícími pulzy kvadraturního enkodéru. Tato metoda je někdy také označována jako „T“ metoda a hodí se pro oblast nižších otáček, neboť trpí přesně opačnou nevýhodou v porovnání s metodou předchozí. Kombinací vysokého rozlišení enkodéru a relativně vyšší rychlosti otáčení dojde k tomu, že časový interval mezi dvěma následujícími hranami kvadraturního enkodéru je malý a tím pádem je výpočet mnohem více ovlivněn rozlišením časovače, což může při vyšších rychlostech způsobit nezanedbatelnou chybu.[16, 22]

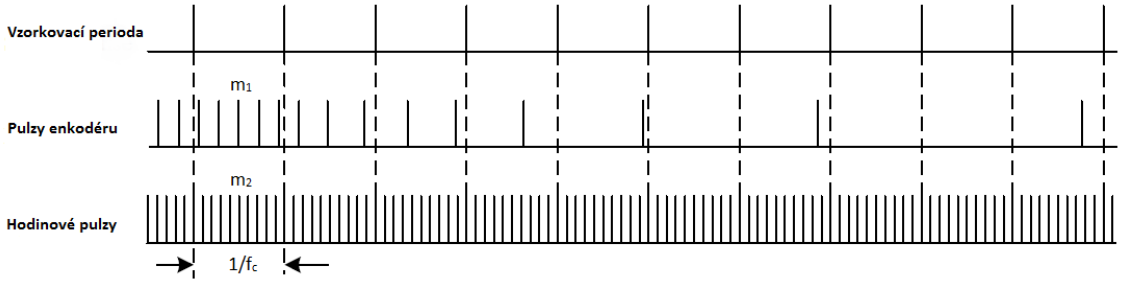
Metoda, která je v rámci vektorového řízení použita pro měření otáček, je v principu kombinací dvou výše zmíněných metod a je označována jako „M/T“ metoda. Poprvé byla představena v [23]. U této metody je měřen nejen počet pulzů enkodéru, ale rovněž počet pulzů hodin časovače mezi prvním a posledním pulzem v daném fixním intervalu. Tím dojde k výraznému zpřesnění v oblasti nízkých rychlostí. Přesnost metody není závislá na otáčivé rychlosti motoru. Navíc je metoda díky eQEP jednotce jednoduše realizovatelná. Na druhou stranu podobně jako „M“ metoda má „M/T“ metoda problém při relativně velmi nízkých otáčkách, kdy počet načítaných pulzů v daném časovém intervalu začne být nulový.[22–23]

Rychlost otáčení hřídele lze při použití „M/T“ metody spočítat pomocí následujícího vztahu:[23]

$$n = \frac{60 f_c m_1}{P m_2}, \tag{6.2}$$

kde n je rychlost motoru v min^{-1}, f_c je frekvence hodin časovače v Hz, P je počet pulzů enkodéru na otáčku, m_1 je počet pulzů enkodéru v měřeném intervalu a m_2 je počet pulzů časovače v měřeném intervalu (myšleno od hrany prvního do hrany posledního pulzu). Princip metody je znázorněn na obr. 6.5.

V [24] jsou uvedené modifikace „M/T“ metody, kterými lze dosáhnout zpřesnění výpočtu, zejména v oblasti nízkých rychlostí. Vztahy pro výpočet otáček jsou totožné se vztahem (6.2). Metody se liší tím, že čítají hodinové pulzy v proměnných intervalech. Jedná se o metodu označovanou jako „Variable M/T method“ a dále pak o metodu označovanou „Leap variable M/T method“. Nicméně klasická „M/T“ metoda je pro účely prezentovaného vektorového řízení naprosto dostačující.



Obrázek 6.5. Princip M/T metody pro měření otáček [24] (autorem)

6.3.2 Výpočet otáček

V předchozí kapitole byla popsána metoda výpočtu otáček a taktéž byl uveden vztah, podle něhož je možné otáčky spočítat (rovnice (6.2)). V této kapitole se zaměříme na konkrétní implementaci metody pomocí eQEP jednotky.

Předpokládáme, že eQEP modul je nakonfigurován v souladu s postupem uvedeným v kap. 5.6.4. Jednotka eQEP pak vyvolává přerušení s periodou 1 ms, ve kterém je volána funkce `speedCalc()`, která na základě hodnot uložených v tzv. „latch“ registrech provede výpočet otáček. Rovnici (6.2) lze přepsat do názornějšího tvaru:

$$\Omega(k) = C_{\text{omega}} \frac{\text{QPOSLAT}(k) - \text{QPOSLAT}(k-1)}{\text{QUPRD} - \text{QCTMRLAT}(k) + \text{QCTMRLAT}(k-1)}, \tag{6.3}$$

kde QPOSLAT je registr, ve němž je uchován počet načítaných pulzů kvadraturního enkodéru, QCTMRLAT je registr, ve kterém je uložena hodnota capture timeru odpovídající rozdílu mezi posledním detekovaným pulzem a okamžikem vyvolání přerušení, QUPRD je registr, kde je uložena hodnota periody výpočtu otáček a C_{omega} je konstanta rozměru s^{-1} převádějící načítané pulzy na mechanickou úhlovou rychlost hřídele. Symbolem (k) je značena aktuální hodnota a symbolem $(k-1)$ hodnota předcházející (ve vztahu k přerušení eQEP jednotky).

Převodní konstanta C_{omega} je závislá na použitém enkodéru a na konfiguraci eQEP jednotky. V našem případě ji lze spočítat takto:

$$C_{\text{omega}} = \pm \frac{2\pi \text{SYSCLKOUT}}{4P}, \tag{6.4}$$

kde SYSCLKOUT je frekvence systémových hodin v Hz a P je počet pulzů enkodéru na otáčku. SYSCLKOUT má hodnotu 150 MHz a je ve vztahu proto, že předdělička hodin eQEP capture timeru je zvolena jako 1. Počet pulzů na otáčku je ve jmenovateli násoben čtyřmi z toho důvodu, že eQEP quadrature counter čítá každou náběžnou i sestupnou hranou obou stop kvadraturního enkodéru, což ve výsledku zdánlivě zčtyřnásobí počet pulzů enkodéru na otáčku. Čitatel je násoben konstantou 2π, abychom výraz převedli na úhlovou frekvenci. Konstanta může být kladná i záporná. Kladná otáčivá rychlost hřídele odpovídá kladnému smyslu otáčení prostorových vektorů v komplexní rovině $\alpha\beta$, záporná otáčivá rychlost odpovídá naopak zápornému smyslu otáčení. Směr otáčení indikuje bit QDF v registru QEPSTS. Pokud je roven nule, tak je výsledná rychlost kladná, pokud je roven jedničce, je výsledná rychlost záporná. Přiřazení bitové hodnoty kladnému nebo zápornému smyslu otáčení bylo provedeno experimentálně.

V přerušení eQEP jednotky jsou dále otáčky v jednotkách s^{-1} převedeny pro účely zobrazení na panelu vytvořeného ovládacího GUI Composer prostředí na min^{-1}. Veškerá regulace, omezení a ochrany ale pracují s hodnotou v s^{-1} . Dále je v přerušení

provedeno potlačení „záporné nuly“. Při stojícím motoru byla totiž výsledná otáčivá rychlost v některých okamžicích indikována jako -0.0. Ukázalo se, že tato hodnota není ekvivalentní s hodnotou 0.0. Výsledek se „zápornou nulou“ bude pravděpodobně souviset s použitými datovými typy a s výpočty v plovoucí řádové čárce. Ošetření je jednoduché – pokud je výsledná otáčivá rychlost rovna -0.0, je tato hodnota pomocí podmínky změněna na 0.0.

Funkce pro výpočet otáček má následující podobu:

```
Uint32 curPosCount,oldPosCount;
Uint16 curDelta,oldDelta;

extern float32 omega;
extern int16 rpm;

void speedCalc(void)
{
    curPosCount=EQep2Regs.QPOSLAT;
    curDelta=EQep2Regs.QCTMRLAT;
    if(EQep2Regs.QEPSTS.bit.QDF)
    {
        if(curPosCount>=oldPosCount)
        {
            omega=-1.0f*OMEGA_CONST*((curPosCount-oldPosCount)/
            (QUPRD_VAl-curDelta+oldDelta));
        }
    } else
    {
        if(oldPosCount>=curPosCount)
        {
            omega=OMEGA_CONST*((oldPosCount-curPosCount)/
            (QUPRD_VAl-curDelta+oldDelta));
        }
    }
    if(omega==-0.0) omega=0.0f;
    oldPosCount=curPosCount;
    oldDelta=curDelta;
    rpm=OMEGA_TO_RPM*omega+0.5f;
}
```

Ještě je třeba vysvětlit vnořené podmínky uvnitř konstrukce `if else` pro rozlišení směru otáčení (bit QDF). Toto opatření bylo nutné přijmout z důvodu mírného zakmitávání rotoru po nabuzení stroje, kdy byly ještě žádané otáčky rovny nule. Kvůli velké citlivosti otáčkového čidla toto zakmitání v některých případech vedlo k výpočtu enormně vysoké rychlosti. Příčinou tohoto nežádoucího jevu bylo zřejmě nesprávné či zpožděné vyhodnocení směru, které vedlo k tomu, že docházelo k odčítání neznaménkové 32bitové proměnné od o několik málo jednotek nižší 32bitové neznaménkové proměnné. Výpočet probíhá v aritmetice modulo $2^{32} - 1$, což v tomto případě dává výsledky blízko maximálnímu rozsahu proměnných. Opatření je sice funkční, ale není bez následků, neboť dojde k přeskočení výpočtu otáček při přetékání rozsahu registru QPOSLAT. Nicméně mechanická časová konstanta je oproti periodě výpočtu otáček násobně delší, takže toto opomenutí nevede k žádným pozorovatelným změnám v chování pohonu.

Kapitola 7
Softwarová implementace bloků vektorového řízení

V této kapitole je popsána programová realizace funkčních bloků navrženého vektorového řízení. Celý algoritmus je založen na pojmenovaných jednotkách v základním tvaru (A, V, Wb, s^{-1}...). Pro názornost a lepší pochopení jsou zde uvedeny výstřižky výsledného kódu[1].

7.1 Stavové bity, řídicí bity a bity kontrolující tok programu

Pro účely řízení toku programu, zpracování vnějších poruch a ovládání celého systému jsou v deklarační části programu založeny pomocí konstrukce `union` nové datové typy. `Union` obsahuje dva prvky. Prvním je bitová struktura s jednotlivými významovými bity a druhým je proměnná s názvem `„all"`, která překrývá všechny bity předešlé bitové struktury. Pokud např. chceme v programu kontrolovat pouze to, že nastala nějaká porucha, můžeme se díky takto deklarovaným typům rovnou podívat na všechny bity najednou a nemusíme testovat jednotlivé poruchové bity zvlášť.

Syntaxe je následující:

```
union
{
    struct
    {
        Uint16 prvniBit:1;
        Uint16 druhyBit:1;
        Uint16 tretiBit:1;
        ...
    };
    Uint16 all;
} nazev_unionu;
```

7.1.1 CONTROLbits

Obsahuje bity, jimiž lze ovládat některé funkce programu.

- `pwmEnable` – povoluje (1) či blokuje (0) ePWM výstupy. Pokud je tento bit v nule, jsou rovněž zablokovány regulátory, výpočty zpětných transformací a modulace prostorového vektoru,
- `faultAcknowledge` – slouží ke kvitaci poruch,
- `startRamp` – spouští otáčkovou rampu.

[1] Kód je v některých případech z důvodu omezeného místa při výpisu zalomen do více řádků.

7.1.2 FAULTbits

Obsahuje bity, které signalizují, že na motoru nastal definovaný poruchový stav. Jedná se o interní proměnné.

- `maxSpeed` – překročení maximálních otáček,
- `maxCurrent` – překročení maximálního proudu.

7.1.3 PROGRAMFLOWbits

Zahrnuje bity, které interně řídí tok programu.

- `adcCalibration` – při startu programu nastaven do jedničky pro spuštění kalibrace offsetu. Po kalibraci trvale nulový,
- `rampON` – signalizuje v programu běžící rampování otáček,
- `pwmOFFDuringMotion` – slouží pro indikaci, že došlo k vypnutí PWM za běhu motoru. Pokud je v jedničce, nelze opětovně zapnout PWM. Po doběhu motoru je vynulován a PWM může být opět povolena.

7.2 Blok Clarkové, Parkovy a inverzní Parkovy transformace

Kód zajišťující provedení Clarkové, Parkovy a inverzní Parkovy transformace se nachází v samostatném hlavičkovém souboru s názvem `clarke_park.h`. Zde jsou po vtáhnutí dalších headrů a definování prototypů funkcí dále pomocí operátoru `typedef` vytvořeny struktury s názvem `CLARKE` a `PARK`. Prvkami těchto struktur jsou vstupní a výstupní parametry každé z transformací. V případě Clarkové transformace jsou vstupními veličinami dva měřené fázové proudy (smybolické značení `A`, `B`), výstupem jsou pak transformované proudy v souřadnicovém systému $\alpha\beta$ (symbolické značení `Alpha`, `Beta`). Do Parkovy transformace vstupují transformované proudy v souřadnicovém systému $\alpha\beta$ a transformační úhel, tj. úhel mezi $\alpha\beta$ a dq (symbolické značení `Alpha`, `Beta`, `Theta`). Výstupem jsou pak transformované proudy v dq (symbolické značení `D`, `Q`). Struktura `PARK` je využita i pro inverzní Parkovu transformaci, neboť její prvky jsou totožné, jedinou změnou je, že veličiny v souřadnicovém systému dq jsou nyní vstupem a veličiny v systému $\alpha\beta$ výstupem. Inverzní Clarkové transformace není v implementovaném vektorovém řízení využita. Následně jsou pomocí nových typů vytvořené proměnné, se kterými se dále pracuje při kalkulaci všech transformací. Hlavičkový soubor pak uzavírají funkce pro konkrétní výpočty. Matematické vyjádření transformací bylo uvedeno v rámci teoretické části v kap. 2.4.3 a 2.4.4. Transformační konstanty jsou pro úsporu výpočetního času spočítány předem a deklarovány v souboru `global_constants.h`.

Pro názornost je zde uveden výřez kódu z hlavičkového souboru (`CLARKE_DEFAULT` a `PARK_DEFAULT` jsou vytvořeny pomocí makra `#define` a obsahují nulové prvky pro počáteční inicializaci proměnných reprezentující jednotlivé struktury):

```
typedef struct
{
    float32 A;
    float32 B;
    float32 Alpha;
    float32 Beta;
}CLARKE;
```

```
typedef struct
{
    float32 Alpha;
    float32 Beta;
    float32 D;
    float32 Q;
    float32 Angle;
}PARK;

CLARKE Clarke=CLARKE_DEFAULT;
PARK Park=PARK_DEFAULT;
PARK ParkInv=PARK_DEFAULT;

void ClarkeCalc(void)
{
    Clarke.Alpha=Clarke.A;
    Clarke.Beta=ONE_OVER_SQRT3*Clarke.A+TWO_OVER_SQRT3*Clarke.B;
}

void ParkCalc(void)
{
    Park.D=Park.Alpha*cos(Park.Angle)+Park.Beta*sin(Park.Angle);
    Park.Q=-1.0f*Park.Alpha*sin(Park.Angle)+Park.Beta*cos(Park.Angle);
}

void ParkInvCalc(void)
{
    ParkInv.Alpha=ParkInv.D*cos(ParkInv.Angle)-
    1.0f*ParkInv.Q*sin(ParkInv.Angle);
    ParkInv.Beta=ParkInv.D*sin(ParkInv.Angle)
    +ParkInv.Q*cos(ParkInv.Angle);
}
```

Funkce zajišťující transformace jsou volány z hlavního zdrojového souboru z přerušení od AD převodníku, kde je vykonávána naprostá většina algoritmů týkající se vektorového řízení. Konkrétní proces bude demonstrován na Clarkové transformaci. Pro ostatní transformace je postup naprosto analogický.

Nejdříve je potřeba do proměnné, resp. struktury `Clarke` předat vstupní parametry (`ia`, `ib` jsou proměnné uchovávající naměřené hodnoty dvou fázových proudů):

```
Clarke.A=ia;
Clarke.B=ib;
```

Následuje zavolání funkce, která provede transformaci:

```
ClarkeCalc();
```

Nakonec je třeba výsledek transformace uložit do proměnných pro další zpracování programu:

```
i1alpha=Clarke.Alpha;
i1beta=Clarke.Beta;
```

7.3 Blok modelu motoru – výpočet rotorového toku a transformačního úhlu

Aby bylo možně transformovat naměřené proudy ze souřadnicového systému $\alpha\beta$ do souřadnicového systému dq, je nutné znát transformační úhel, tj okamžitý úhel mezi těmito systémy. Ten je možné zjistit ze složek α, β rotorového toku jednoduchým výpočtem přes arkus tangens (viz kap. 3.4.1). Pro odhad tohoto toku je vytvořen samostatný hlavičkový soubor s názvem `motor_model.h`. Header obsahuje deklaraci pomocných proměnných použitých během výpočtu společně s příslušnými funkcemi a jejich prototypy. Pomocí klíčového slova `extern` je zde, podobně jako v případě headru sloužícího k výpočtu otáček, umožněno používání některých proměnných z hlavního zdrojového souboru.

7.3.1 Numerické řešení rovnic matematického modelu

Mějme obecně zadanou obyčejnou diferenciální rovnici 1. řádu společně s počáteční podmínkou

$$\begin{aligned} y'(x) &= f(x, y(x)), \\ y(x_0) &= y_0. \end{aligned} \tag{7.1}$$

V uvedené definici je $f : \mathbb{R}^2 \to \mathbb{R}$ funkce definující diferenciální rovnici a $y : \mathbb{R} \to \mathbb{R}$ je hledaná funkce nezávislé reálné proměnné x.[25] Odpověď na otázky ohledně řešitelnosti lze nalézt např. v [26].

Cílem numerických metod je nalezení řešení na nějakém intervalu $< a, b >$ [25]

$$a = x_0 < x_1 < \cdots < x_n = b.$$

Krokem metody h_i rozumíme [25]

$$x_{i+1} - x_i = h_i. \tag{7.2}$$

Pokud je $h_i = h = const.$, říkáme, že se jedná o metodu s pevným krokem.[25]

- **Eulerova metoda**

  Eulerova metoda je nejjednodušší metodou pro řešení diferenciálních rovnic. Patří mezi jednokrokové metody, tj. pro odhad hodnoty f_{i+1} v dalším kroku vyžaduje pouze vypočtený odhad f_i v kroku předchozím. Lze ji odvodit např. ze základního vzorce pro numerickou aproximaci derivace dopřednou diferencí [25]

  $$y'(x) \approx \frac{y(x+h) - y(x)}{h}. \tag{7.3}$$

  Po dosazení za $y'(x)$ z (7.1) a přeskupení členů získáváme předpis pro aproximaci nové hodnoty funkce y na řešeném intervalu $< x_i, x_{i+1} >$ [25]

  $$y_{i+i} = y_i + hf(x_i, y_i). \tag{7.4}$$

  Tato metoda je řádu 1 (definice řádu metody např. v [25]). Její výhodou je malá výpočetní náročnost. U složitějších typů rovnic poskytuje uspokojivé výsledky pouze při volbě dostatečně malého kroku h.[5]

Metoda Runge-Kutta 4. řádu

Runge-Kuttovy metody patří, stejně jako Eulerova metoda, mezi metody jednokrokové. Obecně využívají derivace (směrnice) vypočtené v několika různých místech řešeného intervalu $< x_i, x_{i+1} >$, kterým přiřazují různé váhy α_j [25]

$$y_{i+1} = y_i + h_i(\alpha_1 k_1 + \cdots + \alpha_r k_r) = y_i + h_i \sum_{j=1}^{r} \alpha_j k_j,$$

kde k_j jsou odhady derivací, pro které platí [25]

$$\begin{aligned} k_1 &= f(x_i, y_i) \\ &\vdots \\ k_j &= f(x_i + \lambda_j h_i, y_i + \mu_j h_i k_{j-1}) \quad \text{pro } j > 1 \end{aligned}$$

Jednotlivé metody Runge-Kutta získáme speciální volbou parametrů α, λ a μ.[25]

Nejpoužívanější Runge-Kutta 4. řádu (RK4) je dána předpisem (zde předpokládáme pevný krok h):[25]

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= f(x_i, y_i) \\ k_2 &= f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1) \\ k_3 &= f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2) \\ k_4 &= f(x_{i+1}, y_i + hk_3). \end{aligned}$$

Tato metoda je řádu 4. Čtyři je také počet kroků provedených k získání odhadu k.[26] Níže je podrobněji rozepsán algoritmus výpočtu:[26]

Algoritmus metody RK4

Zadána počáteční úloha ve tvaru $y'(x) = f(x, y(x))$ na intervalu $< x_0, x_0 + T >$ s počáteční podmínkou $y_0 = y(x_0)$, $n \in \mathbb{N}$, krok $h = \frac{T}{n}$. První hodnota y_0 je určena ze zadání. Pro $i = 0, \ldots, n-1$ máme:

1. Odhad $y'(x_i) : k_1 = f(x_i, y_i)$.
2. Odhad $y(x_i + \frac{1}{2}h) : y^*_{i+1/2} = y_i + \frac{1}{2}k_1 h$ a následně směrnici $y'(x_i + \frac{1}{2}h)$: $k_2 = f(x_i + \frac{1}{2}h, y^*_{i+1/2})$.
3. Znovu odhad $y(x_i + \frac{1}{2}h) : y^{**}_{i+1/2} = y_i + \frac{1}{2}k_2 h$ a následně směrnici $y'(x_i + \frac{1}{2}h)$: $k_3 = f(x_i + \frac{1}{2}h, y^{**}_{i+1/2})$.
4. Odhad $y(x_i + h) : y^*_{i+1} = y_i + k_3 h$ a následně smernici $y'(x_{i+1})$: $k_4 = f(x_{i+1}, y^*_{i+1})$.
5. Vypočteme $y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$.

Grafická demonstrace celého postupu je uvedena v příloze C.

Metoda poskytuje velice dobrou stabilitu a přesnost řešení. V praxi je velmi oblíbená, neboť poskytuje dobrý kompromis mezi výpočetní náročností a přesností výsledné aproximace.[26]

7.3.2 Aplikace metody RK4 pro řešení rovnic matematického modelu

Zde je demonstrováno „nasazení“ metody RK4 na soustavu (3.5) a její implementace do F28335.

Přepis rovnic (3.5) do jazyka C:

```
float32 derPsi2Alpha(float32 i1alpha_1, float32 psi2Alpha_1,
float32 psi2Beta_1, float32 omega_1)
{
    static float32 res1=0;
    res1=C1*i1alpha_1+C2*psi2Alpha_1-(POLE_PAIRS*omega_1*psi2Beta_1);
    return res1;
}

float32 derPsi2Beta(float32 i1beta_1, float32 psi2Alpha_1,
float32 psi2Beta_1, float32 omega_1)
{
    static float32 res2=0;
    res2=C1*i1beta_1+C2*psi2Beta_1+(POLE_PAIRS*omega_1*psi2Alpha_1);
    return res2;
}
```

Jedná se o dvě funkce, jejichž parametry jsou složky vektoru statorového proudu a rotorového toku v souřadnicích $\alpha\beta$ a mechanická úhlová rychlost. Funkce pak vrací odpovídající derivace složek rotorového toku. Význam jednotlivých proměnných je zřejmý z jejich mnemotechnického pojmenování. Konstanty `C1` a `C2` v sobě zahrnují podíly a součiny konstantních koeficientů ze soustavy (3.5).

Nyní je již možné provést výpočet nových hodnot rotorového toku v souladu s algoritmem uvedeným v kapitole 7.3.1. Pro tento účel je vytvořena funkce bez návratové hodnoty, které pouze modifikuje už existující proměnné reprezentující složky rotorového toku. Krok řešení h odpovídá vzorkovací periodě, která je 100 μs.

Metoda byla demonstrována na případu obyčejné diferenciální rovnice 1. řádu, kde derivace hledané funkce je funkcí pouze jedné proměnné. V případě rovnic I-n modelu se jedná o soustavu dvou rovnic, kde v každé z nich se vyskytují tři nezávislé proměnné – dvě složky vektoru statorového proudu a elektrická úhlová rychlost rotoru. Navíc je potřeba znát hodnoty nezávislých proměnných v polovině řešeného intervalu, tj. mezi dvěma okamžiky vzorkování, kde informaci o proudech nemáme. To lze vyřešit tak, že neznámé hodnoty uprostřed vzorkovací periody aproximujeme aritmetickým průměrem hodnot proudů ze dvou po sobě následujících vzorkovacích period. Vzhledem k tomu, že vzorkovací perioda je relativně „malá“ oproti změně proudu, tak lze předpokládat, že chyba takovéto aproximace bude rovněž „malá“. Provázanosti rovnic se „zbavíme“ tak, že budeme postupně při výpočtu nových k_i dosazovat do obou funkcí vždy nový odhad obou složek.

Níže je uveden kód, kterým získáme nové složky rotorového toku pomocí RK4 (využijeme u toho výše uvedené funkce):

```
void calcPsi2AlphaBeta_Inmodel(void)
{
        k1Alpha=derPsi2Alpha(i1alphaOld, psi2AlphaOld,
        psi2BetaOld, omegaOld);
        k1Beta=derPsi2Beta(i1betaOld, psi2AlphaOld,
        psi2BetaOld, omegaOld);
```

```
        psi2AlphaAux=psi2AlphaOld+k1Alpha*T_HALF;
        psi2BetaAux=psi2BetaOld+k1Beta*T_HALF;
        k2Alpha=derPsi2Alpha(0.5f*(i1alphaOld+i1alpha),
        psi2AlphaAux, psi2BetaAux,0.5f*(omegaOld+omega));
        k2Beta=derPsi2Beta(0.5f*(i1betaOld+i1beta), psi2AlphaAux,
        psi2BetaAux,0.5f*(omegaOld+omega));
        psi2AlphaAux=psi2AlphaOld+k2Alpha*T_HALF;
        psi2BetaAux=psi2BetaOld+k2Beta*T_HALF;
        k3Alpha=derPsi2Alpha(0.5f*(i1alphaOld+i1alpha),psi2AlphaAux,
        psi2BetaAux,0.5f*(omegaOld+omega));
        k3Beta=derPsi2Beta(0.5f*(i1betaOld+i1beta), psi2AlphaAux,
        psi2BetaAux,0.5f*(omegaOld+omega));
        psi2AlphaAux=psi2AlphaOld+k3Alpha*T;
        psi2BetaAux=psi2BetaOld+k3Beta*T;
        k4Alpha=derPsi2Alpha(i1alpha, psi2AlphaAux,
        psi2BetaAux, omega);
        k4Beta=derPsi2Beta(i1beta, psi2AlphaAux,
        psi2BetaAux, omega);
        kResAlpha=ONE_OVER_SIX*(k1Alpha+2*k2Alpha+2*k3Alpha+k4Alpha);
        kResBeta=ONE_OVER_SIX*(k1Beta+2*k2Beta+2*k3Beta+k4Beta);
        psi2AlphaNew=psi2AlphaOld+kResAlpha*T;
        psi2BetaNew=psi2BetaOld+kResBeta*T;
        psi2AlphaOld=psi2AlphaNew;
        psi2BetaOld=psi2BetaNew;
        i1alphaOld=i1alpha;
        i1betaOld=i1beta;
        omegaOld=omega;
}
```

Vysvětlení symboliky použité v kódu:

- `i1alphaOld, i1betaOld` – složky α, β vektoru statorového proudu v minulé vzorkovací periodě,
- `omegaOld` – hodnota otáčivé rychlosti v minulé vzorkovací periodě[1],
- `psi2AlphaNew, psi2BetaNew` – nově vypočtené složky α, β rotorového magnetického toku,
- `psi2AlphaOld, psi2BetaOld` – složky α, β rotorového toku vypočtené v minulé vzorkovací periodě,
- `psi2AlphaAux, psi2BetaAux` – pomocné hodnoty pro uchování odhadů složek α, β rotorového toku v jednotlivých mezikrocích metody RK4,
- `kiAlpha, kiBeta`, kde i $= 1, 2, 3, 4$ – koeficienty k_i u RK4 pro složky α, β rotorového toku (viz kap. 7.3.1),
- `kResAlpha, kResBeta` – výsledné koeficienty k u RK4 pro složky α, β rotorového toku (viz kap. 7.3.1).

Konstanty u metody RK4 jsou v kódu předpočítány a jejich hodnota je zřejmá z jejich pojmenování. Aby byl další výpočet validní, tak je nutné před ukončením metody do proměnných reprezentující hodnoty veličin v minulém kroku uložit hodnoty získané v aktuálním kroku. Funkce `calcPsiAlphaBeta_Inmodel()` je volána po provedení Clarkové transformace každou vzorkovací periodu v přerušení od AD převodníku.

[1] Perioda výpočtu otáček je 10krát delší než perioda výpočtu složek proudů, tudíž se po relativně dlouhou dobu hodnota otáčivé rychlosti nemění.

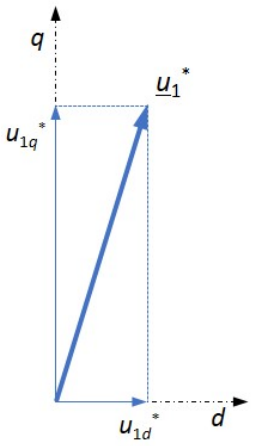
7.3.3 Výpočet transformačního úhlu

Jestliže máme k dispozici vypočtené obě složky rotorového magnetického toku v souřadnicích $\alpha\beta$, můžeme pomocí rovnice (3.19) jednoduše zjistit okamžitý úhel mezi systémy $\alpha\beta$ a dq.

Pro výpočet je nutné zavolat z matematické knihovny namísto klasické funkce `atan` funkci `atan2`. Ta v sobě již obsahuje kontrolu znaménka obou složek a zajistí, že transformační úhel θ bude vypočítán správně dle skutečné aktuální pozice vektoru rotorového magnetického toku. Funkce `atan2` je čtyřkvadrantová verze arkus tangens, která přijímá dva `float32` argumenty `(X,Y)` a vrací úhel rovněž datového typu `float32` v rozmezí $[-\pi, \pi]$ odpovídající podílu `X/Y`.[21]

7.4 Výpočet maximálních hodnot d a q složky žádaného statorového vektoru napětí

Je zřejmé, že výstupy proudových regulátorů musí být omezeny s ohledem na maximální dosažitelné napětí. Toto omezení je vhodné udělat proměnné, a to v závislosti na dostupném napětí ve stejnosměrném meziobvodu střídače. Pohon bude pak možné provozovat i při jiném napětí, než je jeho jmenovité. Na obr. 7.1 je znázorněn požadovaný vektor napětí v souřadnicích dq. Žádané napětí u_{1d}^* je výstupem regulátoru tokotvorné složky proudu, napětí u_{1q}^* pak výstupem regulátoru momentotvorné složky proudu. Předpokládáme, že tato napětí v sobě již obsahují přičtené odvazbovací členy dle rovnic (3.26) a (3.27).



Obrázek 7.1. Požadovaný vektor napětí v souřadnicích dq

Možných způsobů omezení je více. V této práci je použit geometrický způsob s přiřazením priority pro buzení. Maximální vektor napětí $|\underline{u}_{\max}|$, jehož velikost $U_{\mathrm{DC}}/\sqrt{3}$ (viz kap. 3.7) je snížena o násobek λ (viz kap. 3.7.5), je mezi složky d a q rozdělen tak, že saturace regulátoru tokotovorné složky je nastavena na 50 % modulu tohoto vektoru. Po zavolání regulátorů toku a tokotvorného proudu je na základě skutečné požadované hodnoty u_{1d}^* pomocí Pythagorovy věty dopočtena saturace pro regulátor momentotvorné složky:

$$\begin{aligned} u_{1d\max} &= -u_{1d\min} = \frac{1}{2}\lambda|\underline{u}_{\max}| \\ u_{1q\max} &= -u_{1q\min} = \sqrt{(\lambda|\underline{u}_{\max}|)^2 - {u_{1d}^*}^2}. \end{aligned} \tag{7.5}$$

7.5 Vytvořené regulační smyčky

Pro ovládání motoru bylo vytvořeno několik regulačních smyček, mezi kterými je možno za běhu programu přepínat. Jmenovitě jde o:

- otáčkovou regulační smyčku,
- otáčkovou regulační smyčku s rampou otáček ve tvaru „S“ křivky,
- momentovou regulační smyčku se zadáváním žádaného proudu i_q,
- polohovou regulační smyčku.

Programově realizované regulátory všech regulačních smyček se náchází v samostatném hlavičkovém souboru `speed_and_current_controllers.h`. Každý regulátor příslušné veličiny je zakomponován do samostatné funkce bez parametrů a návratové hodnoty. V hlavičkovém souboru jsou rovněž deklarovány prototypy jednotlivých funkcí a jsou zde také založeny proměnné, jež jsou v těchto funkcích využity při výpočtu nových hodnot.

Mimo výše uvedené regulační smyčky lze navíc pohon připnout na třífázovou síť 50 Hz simulovanou pomocí střídače, která byla vytvořena pro demonstrační účely.

7.5.1 Přepínání mezi regulačními smyčkami

Pro účel rozlišení regulačních smyček v programu a možnosti přepínání mezi nimi byl v hlavním zdrojovém souboru pomocí klíčového slova `typedef` vytvořen nový datový typ výčtového typu `enum`. Nový datový typ je pojmenován `control_loop`. Ve výčtovém typu jsou uvedeny symbolické názvy všech typů smyček. Následně jsou pomocí nového typu `control_loop` vytvořeny dvě proměnné `actualControlLoop` a dále pak `newControlLoop`. První ze zmíněných nese informaci o akutální vybrané regulační smyčce a je rozhodující pro její vykonání během vektorového řízení (pozor, neměnit za chodu!). Druhá nese infromaci o nové uživatelem či programátorem žádané smyčce a může za určitých podmínek předat tuto informaci do proměnné `actualControlLoop` (více v kapitole 7.10.4). Obě proměnné jsou inicializovány hodnotou `SPEED_LOOP`.

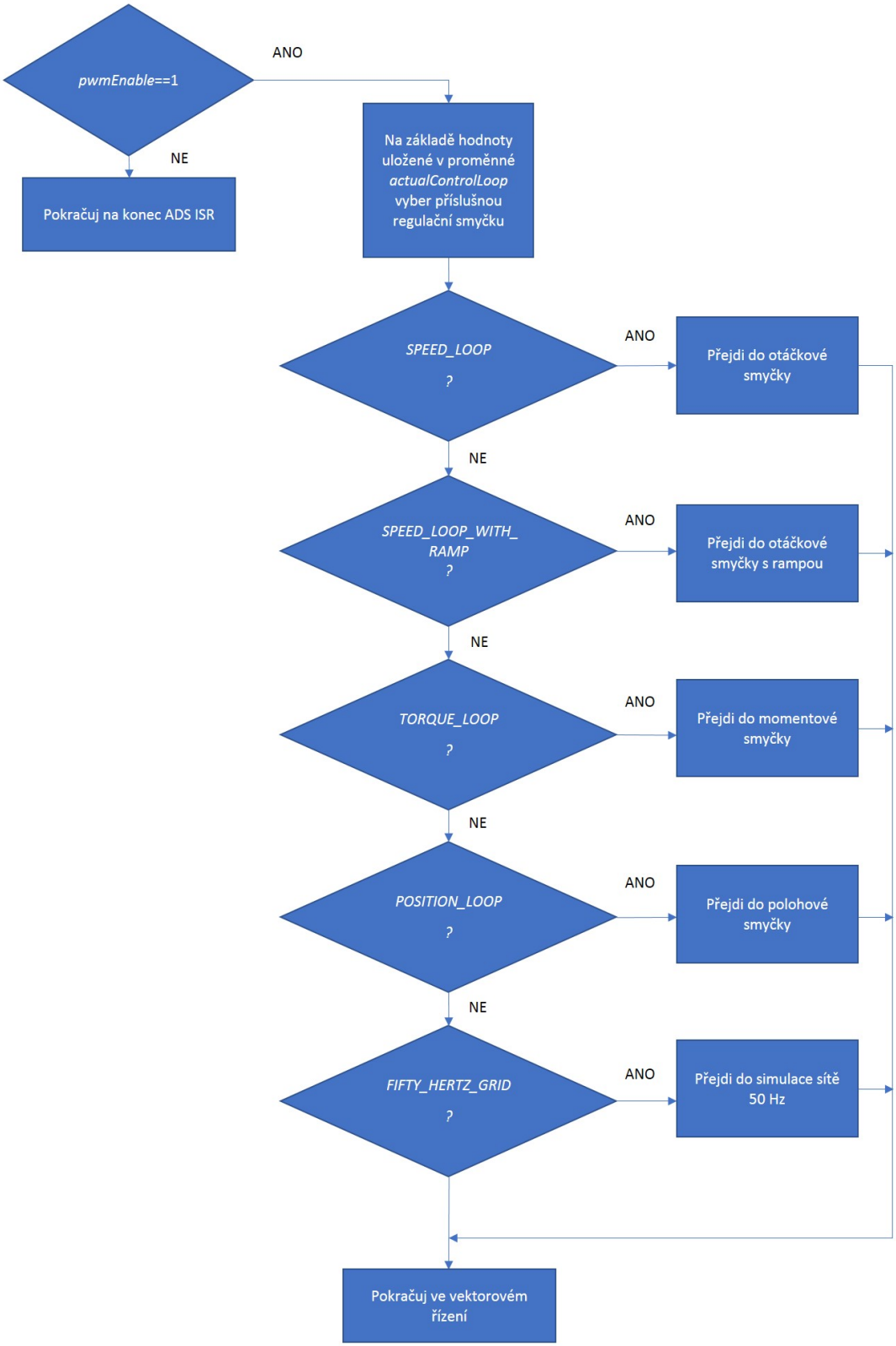
Níže je uveden výřez kódu zajišťující vytvoření a deklaraci zmíněných proměnných:

```
typedef enum
{
    SPEED_LOOP,SPEED_LOOP_WITH_RAMP,TORQUE_LOOP,
    POSITION_LOOP,FIFTY_HERTZ_GRID
}control_loop;

control_loop actualControlLoop;

control_loop newControlLoop;
```

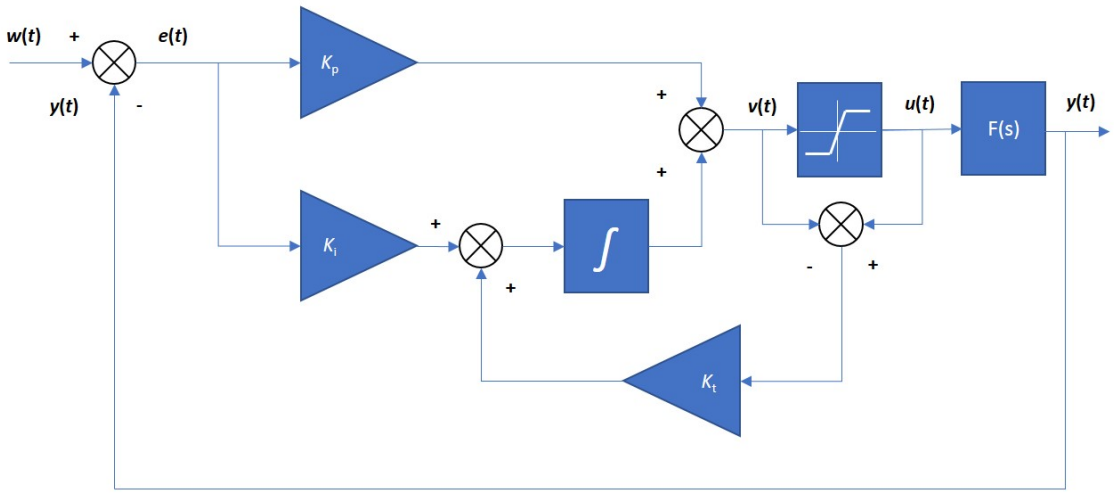
Na obrázku 7.2 je uveden vývojový diagram znázorňující výběr regulační smyčky uvnitř přerušení od AD převodníku. Syntakticky je toto zajištěno pomocí konstrukce `switch()`, do nějž vstupuje hodnota uložená v proměnné `actualControlLoop`.

7.6 PI regulátory

U všech regulačních smyček vyjma smyčky polohové je použit konvenční a v praxi hojně používaný proporciálně-integrační (PI) regulátor s ošetřením unášení integrační složky (anti-windup). Blokové schéma takového regulátoru (spojitého) je znázorněno na



Obrázek 7.2. Vývojový diagram výběru regulační smyčky



Obrázek 7.3. Blokově schéma PI regulátoru s ošetřením wind-up efektu

obr. 7.3.[27] Blok F(s) značí regulovaný proces, v našem případě se jedná o vektorově řízený asynchronní motor.

Význam časových veličin:

- $w(t)$ – žádaná hodnota,
- $e(t)$ – regulační odchylka,
- $y(t)$ – regulovaná veličina,
- $v(t)$ – akční veličina před saturací,
- $u(t)$ – akční veličina po saturaci.

Význam konstant v blocích zesílení:

- K_p – proporcionální konstanta,
- K_i – integrační konstanta,
- K_t – korekční konstanta anti-windup zapojení.

7.6.1 Wind-up efekt

Regulovaná veličina se nemění okamžitě. V důsledku toho může začít narůstat regulační odchylka a následně i integrační složka, která může značně přesáhnout interval hodnot realizovatelný akčním členem. Pokud po nějaké době regulační odchylka změní své znaménko, integrační složka začne klesat, ale její pokles na hodnoty realizovatelné akčním členem může být zdlouhavý. Dochází k překmitu opačným směrem a celkově je regulační pochod zdlouhavý a tlumeně kmitavý, někdy i nestabilní. Tento jev, který je v praxi nutné u každého regulátoru ošetřit, se nazývá **wind-up** integrační složky.[2]

Jedno z možných řešení je znázorněno na již zmíněném obr. 7.3. Za výstup PI regulátoru je zařazen blok omezení (saturace), který omezuje výstup regulátoru na fyzikálně realizovatelný interval hodnot. Akční veličina před omezením je následně odečtena od akční veličiny za omezením a tento rozdíl je vynásoben korekční konstantou K_t. Zesílený rozdíl se následně sečte s regulační odchylkou vynásobenou integrační konstantou a poté vstupuje do integrátoru.[27]

7.6.2 Diskrétní PI (PS) regulátor

Spojitý PI regulátor obsahuje proporcionální (P) a integrační (I) složku. Následující rovnice udává vztah mezi akční veličinou $u(t)$ a regulační odchylkou $e(t)$ spojitého PI regulátoru:[2, 28]

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int e(t)\mathrm{d}t, \tag{7.6}$$

Proporcionální člen přenášení na výstup regulátoru zesílenou regulační odchylku bez časového zpoždění. Integrační složka pak přenáší na výstup zesílený integrál regulační odchylky. Ve vyjádření Laplaceova obrazu se jedná o zpožďující člen prvního řádu. Proporcionální složka působí příznivě na stabilitu a integrační složka zajišťuje nulovou regulační odchylku v ustáleném stavu. V praxi se často používá ještě varianta regulátoru PID s derivační složkou (D). Ta predikuje budoucí chování regulační odchylky, což může zlepšit regulaci, ale na druhou stranu pokud zpracováváme zašuměné signály, může být její přítomnost kontraproduktivní, neboť může naopak působit negativně na stabilitu celé soustavy. Např. fázové proudy asynchronního motoru při vektorovém řízení jsou vlivem PWM poměrně zarušené, proto byla derivační složka vynechána.[2, 28]

Vztah (7.6) je vyjádřením pro spojitý PI regulátor. Číslicový počítač však pracuje diskrétním způsobem s určitou vzorkovací periodou T. Diskrétní tvar PI regulátoru, někdy nazývaný jako PS regulátor (proporcionálně-sumační), lze odvodit diskretizací rovnice (7.6).[2]

Nahrazení integrálu lze provést více způsoby. V tomto případě je použita náhrada dopřednou obdélníkovou metodou, pro kterou platí [2]

$$\int_0^{kT} x(t)\mathrm{d}t \approx \sum_{i=0}^{k-1} T x(i), \tag{7.7}$$

kde $x(t)$ je diskretizovaná proměnná, k označuje číslo vzorku, T je vzorkovací perioda v s a kT je diskrétní čas v s, který bude dále zjednodušeně značen pouze jako k.

Dosazením (7.7) do (7.6) a náhradou spojitého času t za diskrétní k dostaneme použitý tvar regulátoru:

$$u(k) = K_\mathrm{p} e(k) + K_\mathrm{i} \sum_{i=0}^{k-1} e(i). \tag{7.8}$$

Výsledný zápis diskrétního PI (PS) regulátoru s potlačením windup efektu v souladu s obr. 7.3 v jazyce C:

```
void PI_controller(void)
{
    error=setpoint-input;
    output=errorSum+Kp*error;
    outputLimited=output;
    if(output>MAX) outputLimited=MAX;
    if(output<MIN) outputLimited=MIN;
    errorSum+=Ki*error+Kt(outputLimited-output);
}
```

Nejdříve je vypočítána regulační odchylka `error` jako rozdíl žádané hodnoty `setpoint` a navzorkované regulované veličiny `input`. Dále je vypočítána akční veličina regulátoru `output` jako součet integrační složky `errorSum` a regulační odchylky `error` vynásobené proporcionálním zesílením `Kp`. Abychom neztratili informaci o výstupu regulátoru (akční veličině) před blokem omezení `output`, je tato uložena do proměnné reprezentující akční

veličinu za omezením `outputLimited`. Na následujících dvou řádcích je oříznutí proměnné `outputLimited` na hodnotu `MAX` v případě, že přesahuje horní maximální `MAX` mez a na `MIN` pokud přesahuje dolní minimální mez `MIN`. Následně je vypočítána suma regulačních odchylek `errorSum` (integrační člen) jako součet předcházející hodnoty sumy `errorSum`, regulační odchylky `e` vynásobené integračním zesílením `Ki` (konstantu předpokládáme včetně vzorkovací periody) a rozdílu saturovaného `outputLimited` a nesaturovaného `output` výstupu regulátoru vynásobeného korekční konstantou anti-windup zapojení `Kt`. Funkce `PI_controller` je pak volána každou vzorkovací periodu.

Při návrhu regulátoru byl tedy zvolen způsob, kdy je předem známa struktura a úkolem je nalezení vhodných parametrů regulátoru, které zajistí požadovanou kvalitu regulačního pochodu. Tento způsob nemusí vyžadovat znalost matematického modelu řízeného procesu. Diskrétní regulátor vhodný pro programovou realizaci na řídicím počítači byl pak získán diskretizací rovnice popisující regulátor spojitý. Při vhodně zvolené vzorkovací periodě pak lze u diskretizovaného regulátoru dosáhnout podobného chování jako u jeho spojité verze.[2]

7.7 Praktická implementace diskrétních PI (PS) regulátorů

V následujících podkapitolách bude detailněji popsána softwarová implementace použitých PI (PS) regulátorů a to s ohledem na různá omezení a blokování, která jsou vhodná či dokonce nutná pro správnou funkci v reálném nasazení. Všechny konstanty regulátorů byly určovány experimentálně.

7.7.1 Regulátor rotorového toku $|\Psi_2|$

Posloupnost činností po vstoupení do funkce zajišťující regulaci toku:

- omezení žádané hodnoty toku na kladný interval hodnot od minimální hodnoty toku, která je stanovena na 0,25 Wb do jmenovité hodnoty,
- zajištění nemožnosti odbudit stroj, pokud jsou skutečné otáčky stroje nenulové,
- nulování integračního členu regulátoru v případě, že žádaná hodnota toku je nulová a skutečná hodnota je menší než minimální,
- výpočet výstupní veličiny regulátoru dle algoritmu uvedeného v kap. 7.6.2.

Interval možných požadovaných výstupních hodnot tokotvorného proudu i_d tokového regulátoru je omezen na hodnoty od nuly do jmenovité hodnoty, která je vypočtena ze jmenovité hodnoty toku dle rovnice (3.11). Regulátor je volán každou vzorkovací periodu.

proporciální konst. K_p	integrační konst. K_i	korekční anti-windup konst. K_t
100	0,1	0,1

Tabulka 7.1. Nastavené konstanty regulátoru rotorového toku

7.7.2 Regulátor tokotvorné složky proudu i_d

Posloupnost činností po vstoupení do funkce zajišťující regulaci tokotvorné složky proudu:

- výpočet odvazbovacího členu dle rovnice (3.26) za předpokladu, že žádaná hodnota toku je větší než minimální (potlačení dělení nulou a malými čísly), jinak je odvazbovací člen položen nule,

- výpočet výstupní veličiny regulátoru dle algoritmu uvedeného v kap. 7.6.2.

Odvazbovací člen je počítán ze žádaných hodnot, neboť ty kolísají o dost méně než hodnoty skutečné. Omezení výstupní žádané hodnoty napětí je provedeno postupem uvedeným v kap. 7.4. Regulátor tokotovorné složky proudu je volán každou vzorkovací periodu.

proporciální konst. K_p	integrační konst. K_i	korekční anti-windup konst. K_t
5	0,15	0,15

Tabulka 7.2. Nastavené konstanty regulátoru tokotvorné složky proudu

7.7.3 Regulátor otáček Ω

Posloupnost činností po vstoupení do funkce zajišťující regulaci otáček:

- omezení maximálních žádaných otáček shora na jmenovitou a z důvodu rozlišení metody pro výpočet otáček zdola na minimální (zvoleno 5 s^{-1}),
- nastavení žádané hodnoty otáček na nulu v případě, že žádaná hodnota toku je menší než minimální,
- nulování integračního členu, pokud jsou žádané a skutečné hodnoty otáček rovny nule,
- výpočet výstupní veličiny regulátoru dle algoritmu uvedeného v kap. 7.6.2.

Z důvodu implementované otáčkové rampy pracuje regulátor otáček se stínovou proměnnou skutečných otáček, která je ale v případě čistě otáčkové smyčky rovna skutečným otáčkám. Omezení žádané hodnoty momentotvorné složky proudu je nastaveno v kladném i záporném smyslu na jmenovitou hodnotu. Regulátor otáček je volán jednou za deset vzorkovacích period, neboť se stejnou periodou probíhá i výpočet změřených otáček.

proporciální konst. K_p	integrační konst. K_i	korekční anti-windup konst. K_t
6	0,1	0,1

Tabulka 7.3. Nastavené konstanty regulátoru otáček

7.7.4 Regulátor momentotvorné složky proudu i_q

Posloupnost činností po vstoupení do funkce zajišťující regulaci momentotvorné složky proudu:

- omezení výstupní žádané hodnoty napětí postupem uvedeným v kap. 7.4,
- výpočet odvazbovacího členu dle rovnice (3.27) za předpokladu, že žádaná hodnota toku je větší než minimální (potlačení dělení nulou a malými čísly), jinak je odvazbovací člen položen nule,
- výpočet výstupní veličiny regulátoru dle algoritmu uvedeného v kap. 7.6.2.

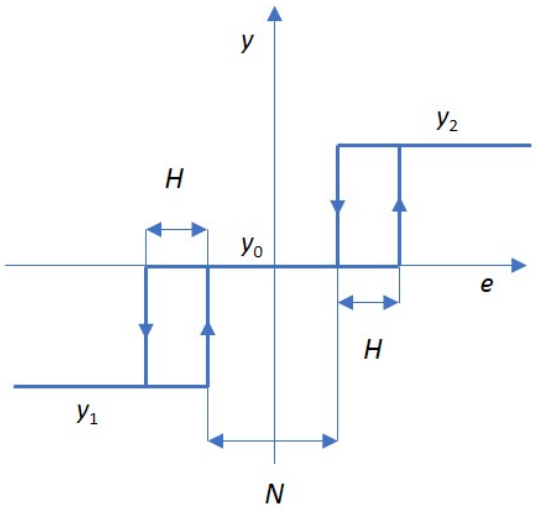
Odvazbovací člen je, stejně jako u regulátoru tokotvorné složky proudu, počítán ze žádaných hodnot. Regulátor tokotovorné složky proudu je volán každou vzorkovací periodu.

proporciální konst. K_p	integrační konst. K_i	korekční anti-windup konst. K_t
5	0,15	0,15

Tabulka 7.4. Nastavené konstanty regulátoru momentotvorné složky proudu

7.8 Třípolohový hysterezní regulátor

Jako regulátor polohy v polohové smyčce je použit třípolohový hysterezní regulátor, jenž patří do skupiny tzv. nespojitých regulátorů. Ty jsou charakteristické tím, že jejich akční veličina může nabývat pouze dvou (v případě dvoupolohového) nebo tří (v případě třípolohového) hodnot. Charakteristika třípolohového regulátoru s hysterezí je znázorněna na obr. 7.4.[29]



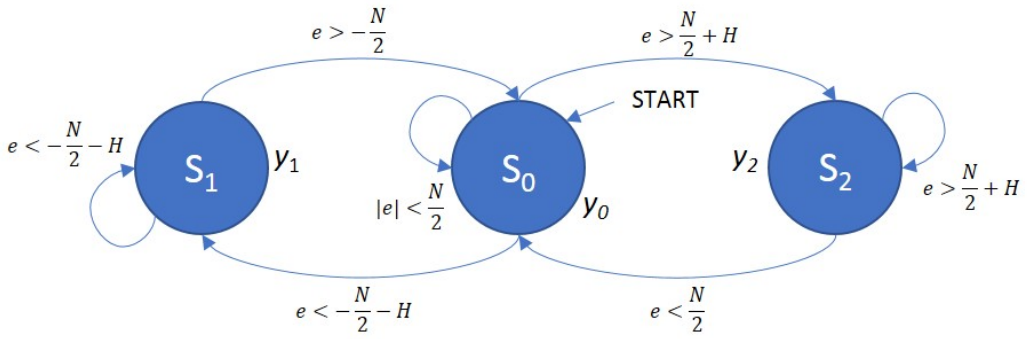
Obrázek 7.4. Charakteristika třípolohového regulátoru s hysterezí

Na uvedeném obrázku značí e regulační odchylku, y_1 první výstupní hodnotu, y_2 druhou výstupní hodnotu, y_0 nulovou výstupní hodnotu, N pásmo necitlivosti a H hysterezní pásmo. Pokud je $|e| < N/2$, je regulátor v nulovém stavu, to znamená, že akční veličina je rovna nulové hodnotě. Jestliže regulační odchylka narůstá směrem do kladných hodnot, je po překročení hysterezního pásma výstup regulátoru přepnut na hodnotu y_2. V případě, že regulační odchylka naopak klesá směrem do záporných hodnot, tak je po překročení hysterezního pásma v levé části charakteristiky výstup nastaven na hodnotu y_1. Intervaly H jsou zde vloženy z toho důvodu, aby se zabránilo častému přepínání a tedy i kmitání okolo hranic pásma necitlivosti s pásmy nenulových výstupních akčních veličin.[29]

Třípolohový hysterezní regulátor je vlastně rozšířením dvoupolohového hysterezního regulátoru, kde akční veličina nabývá dvou hodnot a regulovaná veličina se pohybuje uvnitř hysterezního pásma. Takovýto regulátor se používá například pro regulaci teploty topných těles. Pro použití v systému se servopohonem je ale nevhodný, neboť v takovém případě by pohon neustále kmital v pásmu kolem žádané polohy. Pro použití se servopohony je naopak vhodný třípolohový regulátor, který zajistí, že pohon bude v intervalu kolem žádané hodnoty v klidu.[28–29]

Programově lze třípolohový regulátor realizovat v podobě třístavového konečného automatu, jehož stavový diagram je znázorněn na obr. 7.5. Značení ve stavovém diagramu odpovídá charakteristice 7.4. V diagramu jsou uvedeny jednotlivé stavy společně s příslušnými výstupy a podmínkami přechodu.

Na základě obr. 7.5 lze psát kód v jazyce C, který bude programově tento typ regulátoru realizovat. Nejdříve je vhodné si pomocí klíčového slova `typedef` a výčtového typu `enum` vytvořit nový datový typ se symbolickým pojmenováním jednotlivých stavů



Obrázek 7.5. Stavový diagram třípolohového hysterezního regulátoru

automatu. Dále pomocí nově vytvořeného typu `controller_state` založíme dvě nové proměnné. První s názvem `currentControllerState` v sobě nese informaci o aktuálním stavu, v němž se automat nachází, a druhá s názvem `nextControllerState` reprezentuje nový stav, do kterého má automat po splnění příslušných podmínek přejít:

```
typedef enum
{
    S0,S1,S2
} controller_state;

controller_state currentControllerState=S0;
controller_state nextControllerState=S0;
```

Nyní je již možné psát funkci, která pro regulační odchylku `error` danou rozdílem žádané hodnoty `setpoint` a skutečné hodnoty vypočtené na základě vstupní informace z vnějšího čidla `input` vygeneruje příslušnou výstupní akční veličinu `y0`, `y1` nebo `y2`:

```
void threeLevelHysteresisController(void)
{
    error=setpoint-input;

    switch(currentControllerState)
    {
     case S0:
         if(error>N/2+H) nextControllerState=S2;
         if(error<-N/2-H) nextControllerState=S1;
         output=y0;
         break;
     case S1:
         if(e>-N/2) nextControllerState=S0;
         output=y1;
         break;
    case S2:
         if(e<N/2) nextControllerState=S0;
         output=y2;
         break;
    }
    currentControllerState=nextControllerState;
}
```

Pro realizaci konečného automatu v jazyce C lze s výhodou použít konstrukci `switch`. Vstupním parametrem je proměnná `currentControllerState`. Na základě hodnoty v této proměnné se přejde do příslušného stavu, kde se testuje splnění podmínek znázorněných ve stavovém diagramu 7.5. Pokud je nějaká z podmínek přechodu splněna, uloží se tento nový stav do proměnné `nextControllerState`. Před ukončením funkce je nutné předat do aktuálního stavu hodnotu stavu následujícího.

7.8.1 Regulátor polohy

Jestliže je jako nová regulační smyčka vybrána smyčka polohová a jsou-li splněny podmínky pro přechod do nové smyčky (viz kap. 7.10.4), je v případě nulových skutečných otáček provedena inicializace eQEP jednotky nutná pro zajištění správné regulace polohy. Jedná se o posloupnost následujících činností:

- zakázání přerušení UTO, ve kterém probíhá výpočet otáček,
- inicializace registru QPOSINIT pomocí bitu SWI (registr QEPCTL) na hodnotu odpovídající polovině maximálního rozsahu,
- inicializace proměnných obsahujících informaci o minulé a aktuální poloze na hodnotu QPOSCNT,
- inicializace stínové proměnné žádaných otáček rovněž na hodnotu QPOSCNT,
- opětovné povolení přerušení UTO.

Výše uvedenými kroky inicializujeme registr QPOSCNT a proměnné, se kterými přímo pracuje polohový regulátor do definovaného počátečního stavu. Výchozí polohou rotoru pro regulátor je pak ta, v níž se rotor nachází po zapnutí polohové regulační smyčky. Zadávání žádaného natočení probíhá v úhlových stupních.

Po zavolání regulátoru polohy dojde k omezení žádaného natočení na hodnotu 3600° v kladném i záporném smyslu. Následně je proveden přepočet žádané hodnoty z úhlových stupňů na pulzy enkodéru. Tato hodnota je následně přičtena, resp. odečtena od QPOSINIT registru QPOSCNT. Výsledek je uložen do stínové proměnné, která je odečtena od hodnoty v registru QPOSCNT (odpovídá načítaným pulzům). Nakonec je provedena samotná regulace pomocí algoritmu uvedeného v kap. 7.8. Maximální žádaná hodnota otáček je nastavena na 5 s^{-1} v kladném i záporném smyslu. V neutrálním pásmu jsou žádané otáčky rovny nule. Regulátor polohy je pak, stejně jako regulátor otáček, volán jednou za deset vzorkovacích period.

7.9 Blok modulace prostorového vektoru

Žádaný napěťový vektor transformovaný do systému $\alpha\beta$ inverzní Parkovo transformací vstupuje do modulátoru, který má za úkol vygenerovat odpovídající pulzy pro budiče tranzistorů. Kód pro modulaci prostorového vektoru se nachází v samostatném hlavičkovém souboru `space_vector_modulation.h`.

Na začátku jsou pomocí klíčového slova `typedef` a výčtového typu `enum` vytvořeny nové datové typy `quadrants` a `sectors`, které jsou použity pro uchování informace o kvadrantu, resp. sektoru, v němž se nachází požadovaný napěťový vektor:

```
typedef enum
{
  Q1,Q2,Q3,Q4
}quadrants;

typedef enum
```

```
{
  S1,S2,S3,S4,S5,S6
}sectors;

quadrants quadrant;
sectors sector;
```

Dále je nutné založit proměnné, které budou nést informaci nutnou pro finální výpočet spínacích časů (vysvětlení a značení viz kap. 3.7) a proměnné uchovávající hodnoty nahrávané do příslušných komparačních registrů (význam je patrný z jejich mnemotechnického pojmenování):

```
float32 ur,ul,Tr,Tl,T0;
Uint16 compValEPWM1A,compValEPWM2A,compValEPWM3A;
```

Samotná modulace probíhá ve funkci `spaceVectorModulation()`, jejímiž vstupními parametry jsou složky žádaného napěťového vektoru v systému $\alpha\beta$. Na začátku této funkce jsou založeny pomocné proměnné `a`, `b`, `c`, které jsou periodicky inicializovány dle rovnic (3.33). Následně je soustavou podmíněných příkazů rozlišeno, v jakém kvadrantu, resp. sektoru, se požadovaný vektor nachází. Jedná se vlastně o přímé přepsání vývojového diagramu 3.11 do jazyka C.

Po tomto kroku jsou v souladu s rovnicemi (3.30) vypočítány doby trvání „pravého“ a „levého“ napěťového vektoru v daném sektoru a vektorů „nulových“. Podmínkou je zde potlačeno dělení nulou, ke kterému dochází, jestliže je napětí ve stejnosměrném meziobvodu rovno nule.

Vypočítané doby trvání jednotlivých vektorů je následně nutné převést na celočíselné hodnoty, které budou nahrány do komparačních registrů jednotlivých ePWM submodulů. V této části je výhodné a nápomocné zároveň sledovat jednotlivé spínací sekvence znázorněné na obr. 3.9:

```
    switch(sector)
    {
        case S1:
            compValEPWM1A=0.5f*((T0/T)*PWM_TBPRD);
            compValEPWM3A=PWM_TBPRD-compValEPWM1A;
            compValEPWM2A=compValEPWM3A-((Tl/T)*PWM_TBPRD);
        break;
        case S2:
            compValEPWM2A=0.5f*((T0/T)*PWM_TBPRD);
            compValEPWM3A=PWM_TBPRD-compValEPWM2A;
            compValEPWM1A=compValEPWM3A-((Tr/T)*PWM_TBPRD);
        break;
        case S3:
            compValEPWM2A=0.5f*((T0/T)*PWM_TBPRD);
            compValEPWM1A=PWM_TBPRD-compValEPWM2A;
            compValEPWM3A=compValEPWM1A-((Tl/T)*PWM_TBPRD);
        break;
        case S4:
            compValEPWM3A=0.5f*((T0/T)*PWM_TBPRD);
            compValEPWM1A=PWM_TBPRD-compValEPWM3A;
            compValEPWM2A=compValEPWM1A-((Tr/T)*PWM_TBPRD);
        break;
        case S5:
            compValEPWM3A=0.5f*((T0/T)*PWM_TBPRD);
```

```
            compValEPWM2A=PWM_TBPRD-compValEPWM3A;
            compValEPWM1A=compValEPWM2A-((Tl/T)*PWM_TBPRD);
        break;
        case S6:
            compValEPWM1A=0.5f*((T0/T)*PWM_TBPRD);
            compValEPWM2A=PWM_TBPRD-compValEPWM1A;
            compValEPWM3A=compValEPWM2A-((Tr/T)*PWM_TBPRD);
        break;
    }
```

Vzhledem k omezení žádaného napěťového vektoru dle rovnic (7.5) by nemělo docházet ke stavům, kdy je šířka řídicího pulzu pro tranzistory příliš krátká. Nicméně vypočítané hodnoty jsou pro jistotu i tak omezeny a zároveň je zajištěno, aby nedošlo k překročení rozsahu komparačních registrů:

```
    if(compValEPWM1A>MAX_COMP_VAL && compValEPWM1A<PWM_TBPRD)
    {
        compValEPWM1A=MAX_COMP_VAL;
    } else if(compValEPWM1A<MIN_COMP_VAL && compValEPWM1A>0)
    {
        compValEPWM1A=MIN_COMP_VAL;
    } else if (compValEPWM1A>PWM_TBPRD)
    {
        compValEPWM1A=PWM_TBPRD;
    }

    if(compValEPWM2A>MAX_COMP_VAL && compValEPWM2A<PWM_TBPRD)
    {
        compValEPWM2A=MAX_COMP_VAL;
    } else if(compValEPWM2A<MIN_COMP_VAL && compValEPWM2A>0)
    {
        compValEPWM2A=MIN_COMP_VAL;
    } else if (compValEPWM2A>PWM_TBPRD)
    {
        compValEPWM2A=PWM_TBPRD;
    }

    if(compValEPWM3A>MAX_COMP_VAL && compValEPWM3A<PWM_TBPRD)
    {
        compValEPWM3A=MAX_COMP_VAL;
    } else if(compValEPWM3A<MIN_COMP_VAL && compValEPWM3A>0)
    {
        compValEPWM3A=MIN_COMP_VAL;
    } else if (compValEPWM3A>PWM_TBPRD)
    {
        compValEPWM3A=PWM_TBPRD;
    }
```

Minimální šířka řídicího pulzu, resp. doba trvání vektoru $\underline{v}_7$ byla stanovena pro použité tranzistory a budiče experimentálně a to na 2 μs. Z toho pak plyne $T_{0\min}$ rovno 4 μs. V kontextu komparačního registru tomu odpovídá minimální hodnota `MIN_COMP_VAL` rovna 75. Maximální hodnota `MAX_COMP_VAL` je pak rovna `PWM_TBPRD` minus `MIN_COMP_VAL`. Hodnoty rovny vrcholu komparačního registru a hodnoty nulové jsou však povoleny, neboť tomu odpovídá trvalé sepnutí, resp. rozepnutí tranzistorů.

V posledním kroku jsou vypočtené a omezené hodnoty nahrány do komparačních registrů EPWM jednotek:

```
EPwm1Regs.CMPA.half.CMPA=compValEPWM1A;
EPwm2Regs.CMPA.half.CMPA=compValEPWM2A;
EPwm3Regs.CMPA.half.CMPA=compValEPWM3A;
```

Funkce `spaceVectorModulation` je pak volána při nastaveném bitu `pwmEnable` po provedené všech předchozích kroků vektorového řízení, tj. téměr na konci přerušení AD převodníku. Jestliže je vybrána simulace sítě 50 Hz, jsou hodnoty do komparačních registrů nahrávány přímo a k jejímu volání tedy nedochází.

7.10 Ochrany, nulování, blokování

7.10.1 Blokování PWM během kalibrace

Pro správnou kalibraci offsetu obvodů pro měření proudu je nutné, aby motorem netekl žádný proud, tj. aby byly vypnuty PWM výstupy. To je zajištěno na pozadí programu a v přerušení AD převodníku. Pokud probíhá kalibrace, není možné zapnout PWM (bit je trvale resetován).

7.10.2 Otáčková a nadproudová ochrana

Na pozadí programu je sledováno, zdali skutečná hodnota otáček nepřekročí maximální možnou dovolenou hodnotu (stanovena na 165 s^{-1}) v kladném i záporném smyslu. V případě, že jsou maximální otáčky překročeny, je do bitu `maxSpeed` struktury `FAULTbits` zapsána hodnota 1.

Nadproudová ochrana se nachází na začátku přerušení AD převodníku. Pokud jakýkoliv kanál ze tří kanálů, na který jsou připojeny výstupy z upravovacích obvodů proudových čidel dosáhne kladné nebo záporné saturace (odpovídá zhruba proudu ±40 A), která bude trvat nepřetržitě 10 vzorkovacích period, dojde k zablokování PWM a do bitu `maxCurrent` struktury `FAULTbits` je zapsána hodnota 1.

Na pozadí je pak v případě, že dojde k jakékoliv poruše, blokována PWM. Aby bylo možné PWM znovu povolit, je nutné poruchu kvitovat zapsáním log. 1 do bitu `CONTROLbits.faultAcknowledge`. Jestliže je zároveň otáčivá rychlost motoru rovna nule, dojde k odblokování PWM a pohon lze dále provozovat. Při nepřítomnosti poruchy je bit `CONTROLbits.faultAcknowledge` trvale resetován.

7.10.3 Vypnutí PWM během chodu pohonu

Jestliže je vypnuta PWM v okamžiku, kdy je otáčivá rychlost pohonu nenulová, je do bitu `PROGRAMFLOWbits.pwmOFFDuringMotion` zapsána log. 1 a dochází trvale k blokování PWM, kterou lze znovu povolit, až když motor dosáhne nulových otáček.

7.10.4 Zvolení nové regulační smyčky a nulování paměťových proměnných

Informace o žádosti o přechodu do nové regulační smyčky je uchovávána v pomocné proměnné výčtového typu enum `newControlLoop`. Ta je na pozadí programu nahrána do skutečné rozhodovací proměnné `actualControlLoop` v přpadě, že je zablokována PWM, otáčky motoru jsou nulové a není přítomna žádaná porucha. Zároveň při tom dochází k nulování všech proměnných obsahujících žádané hodnoty, integračních členů a pomocných proměnných pro numerické řešení matematických modelů. Dále je zde inicializována eQEP jednotka a proměnné, které využívá polohový regulátor (viz kap. 7.8.1).

7.10.5 Spínání brzdného odporu

Při brzdění motoru nebo jeho reverzaci dochází k tomu, že pohon přechází z motorického do generátorického chodu, kdy se naakumulovaná mechanická energie mění na energii elektrickou. V důsledku toho začne stoupat napětí na kondenzátoru ve stejnosměrném meziobvodu, neboť tuto energii není možné z důvodu napájení střídače diodovým můstkem vrátit zpět do sítě. Aby nedošlo ke zničení použitých kondenzátorů, tak je paralelně k hornímu spínači čtvrté větve napájecího měniče připojen brzdný rezistor. V případě, že napětí v meziobvodu překročí určitou hodnotu, která je nastavena na 600 V, je sepnut dolní spínač, který tak zajistí, že je tato generátorická energie mařena na teplo a dále již nezvyšuje stejnosměrné napětí v meziobvodu nad ještě přípustnou hodnotu.

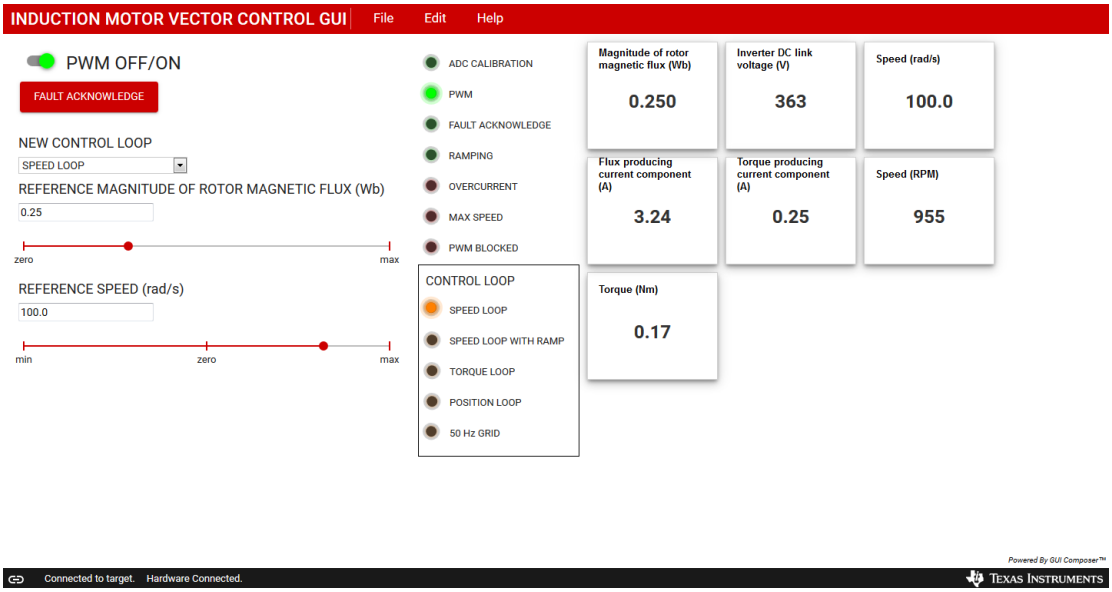
7.11 GUI Composer a prostředí pro ovládání pohonu

GUI Composer je cloudový nástroj od Texas Instruments, který umožňuje pomocí (zejména) grafického programování tvorbu nejrůznějších uživatelských aplikací, které ve výsledku zprostředkovávají komunikaci se zařízeními od Texas instruments. V tomto prostředí bylo vytvořeno uživatelské rozhraní pro ovládání asynchronního motoru přes PC. Pro používání GUI Composeru je nutné mít založený účet na webových stránkách společnosti Texas Instruments.[30]

Při tvorbě nového projektu v GUI Composeru je nutné definovat způsob komunikace s cílovým zařízením. V tomto případě je pro komunikaci použit XDS100v3 USB JTAG emulátor, tudíž je v nastavení nutné vybrat z nabídky možností komunikace „XDS“ a z nabídky zařízení vybrat „TMSF28335“ a „Texas Instruments XDS100v3 Debug Probe“. Pro úspěšnou komunikaci je nutné mít na PC nainstalován TICloudAgent. K jeho instalaci je uživatel automaticky vyzván a je mu zároveň poskytnut instalační soubor. Výsledná GUI Composer aplikace může běžet jednak online v cloudu, dále lze projekt exportovat a spustit uvnitř CCS nebo jako standalone aplikaci na PC. Pro chod z cloudu je nutné mít ve webovém prohlížeči nainstalován doplněk TICloudAgent Bridge a pro běh jako standalone aplikace je nezbytné mít na PC nainstalován GUI Composer Runtime. V obou případech je uživatel stejně jako v případě TICloudAgent vyzván k instalaci těchto rozhraní a jsou mu ke stažení nabídnuty příslušné instalační soubory. Z vytvořeného rozhraní je možné cílové zařízení naprogramovat. V projektovém nastavení je potom nutné vybrat a uploadovat spustitelný soubor s kódem.[30–31]

Nevýhodou ovládání uživatelské aplikace pomocí GUI Composeru s připojeným XDS100vs3 USB JTAG emulátorem je to, že změna proměnných z vytvořené aplikace probíhá jejich přepsáním (podobně jako v debug režimu) a to v (pravděpodobně) nedefinovaném časovém okamžiku. Proto musí být modifikovatelné proměnné v programu průběžně blokovány, pokud není žádané, aby docházelo k jejich změně.

Vytvořené rozhraní (obr. 7.6) umožňuje vizualizaci vybraných proměnných, zadávání žádané hodnoty a přepínání mezi regulačními smyčkami. Zároveň jsou zde kontrolky, které signalizují poruchové i běžné stavy.



Obrázek 7.6. Vytvořené ovládací prostředí v GUI Composeru

Kapitola 8
Naměřené průběhy

8.1 Průběh regulovaných veličin v přechodných dějích

V této kapitole jsou ukázány experimentálně naměřené průběhy žádaných a skutečných motorových veličin v přechodných dějích. Grafy jsou sestrojeny z csv dat exportovaných z RTM. Pohon byl provozován v otáčkové smyčce.

8.1.1 Regulované veličiny při rozběhu motoru

Na několika následujících obrázcích jsou uvedeny průběhy regulovaných veličin při zadání skoku otáček. Nastavené podmínky během rozběhu:

- žádaná hodnota rotorového toku: 0,8 Wb,
- skok otáček: $0 \rightarrow 100$ s^{-1},
- napětí ve stejnosměrném meziobvodu střídače: 540 V,
- perioda odesílání vzorků proměnných na RTM: 1 ms,
- zátěžný moment: nulový (pohon naprázdno).

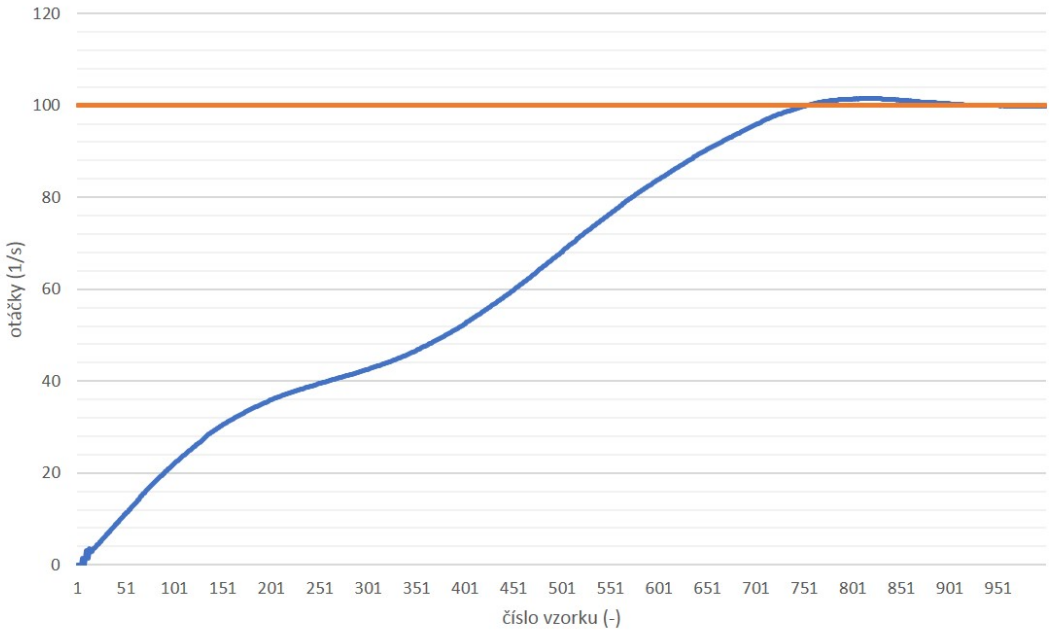
Na obr. 8.1 je znázorněn průběh žádaných a skutečných otáček. Je zde patrná vysoká dynamika, neboť pohon je rozbíhán hnacím momentem blízkým jmenovitému a tím pádem dosahuje žádaných otáček v relativně krátkém čase (do 1 s). Na přechodové charakteristice je patrný mírný překmit. Ten lze zmírnit zvýšením proporciální konstanty otáčkového regulátoru, což ale v některých případech vede k jeho nežádoucímu kmitání. Je pravděpodobné, že při nenulovém zátěžném momentu by tento překmit byl potlačen.

Na obr. 8.2 je zachycen průběh žádané a skutečné hodnoty momentotvorné složky proudu. Na tomto grafu je patrná velice dobrá schopnost regulátoru sledovat žádanou hodnotu v širokém rozsahu proudů a to při přijatelném překmitu zhruba 15 %. Je zde také dobře vidět průběh poklesu žádané hodnoty proudu při dosažení nastavených otáček.

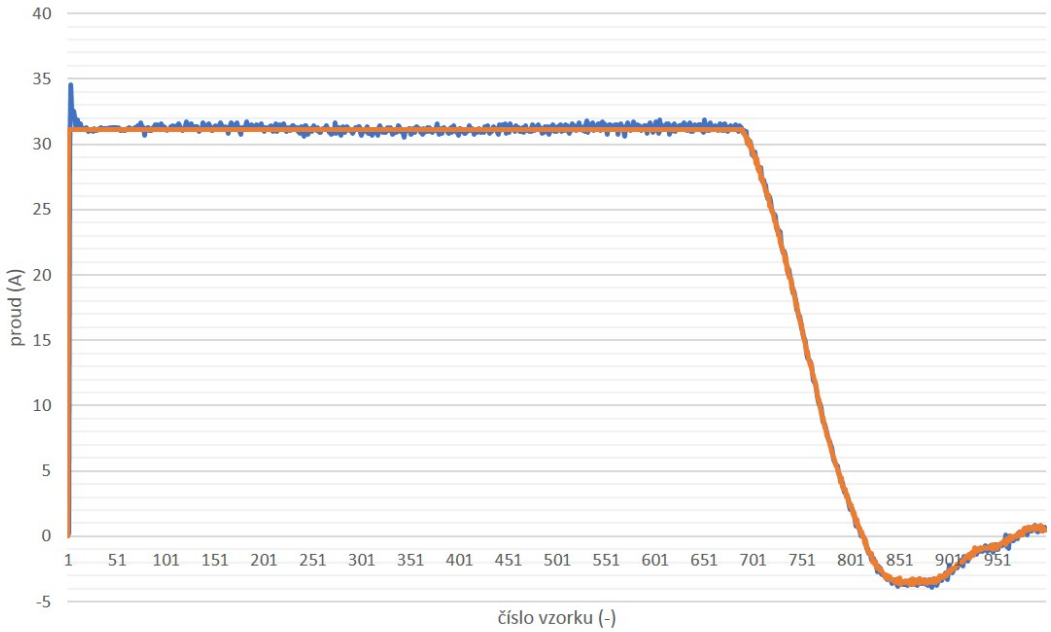
Obr. 8.3 zobrazuje průběh žádané a skutečné hodnoty tokotvorné složky proudu. Úmyslně je zvoleno přiblížení veličin na svislé ose, aby bylo patrné vyšší kmitání požadované i skutečné hodnoty během mechanického přechodného děje.

Konečně obr. 8.4 zachycuje průběh žádané a skutečné hodnoty rotorového toku. Pro větší názornost jsou opět děje přiblíženy. Je vidět, že během rozběhu dochází k oscilacím skutečné hodnoty toku okolo hodnoty požadované. Při bližším pohledu na měřítko však zjistíme, že absolutní hodnota těchto oscilací se řádově pohybuje v tisících Wb, což dokazuje dobrou stabilitu a přesnost bloku matematického modelu.

Na základě dokumentovaných průběhů lze závěrem konstatovat, že navržené schéma vektorového řízení spolu se zvolenou strukturou regulátorů a jejich zvolenými parametry poskytuje relativně kvalitní průběhy veličin během mechanického přechodného děje.



Obrázek 8.1. Odezva na skok otáček, Ω^* (oranžová), Ω (modrá)

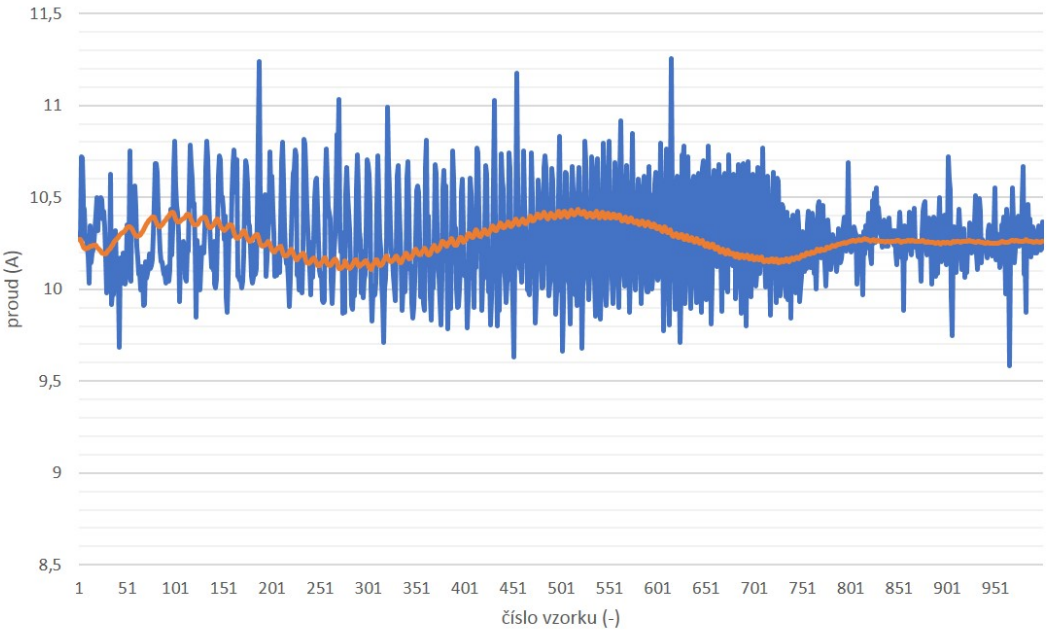


Obrázek 8.2. Průběh momentotvorného proudu při rozběhu, i_{1q}^* (oranžová), i_{1q} (modrá)

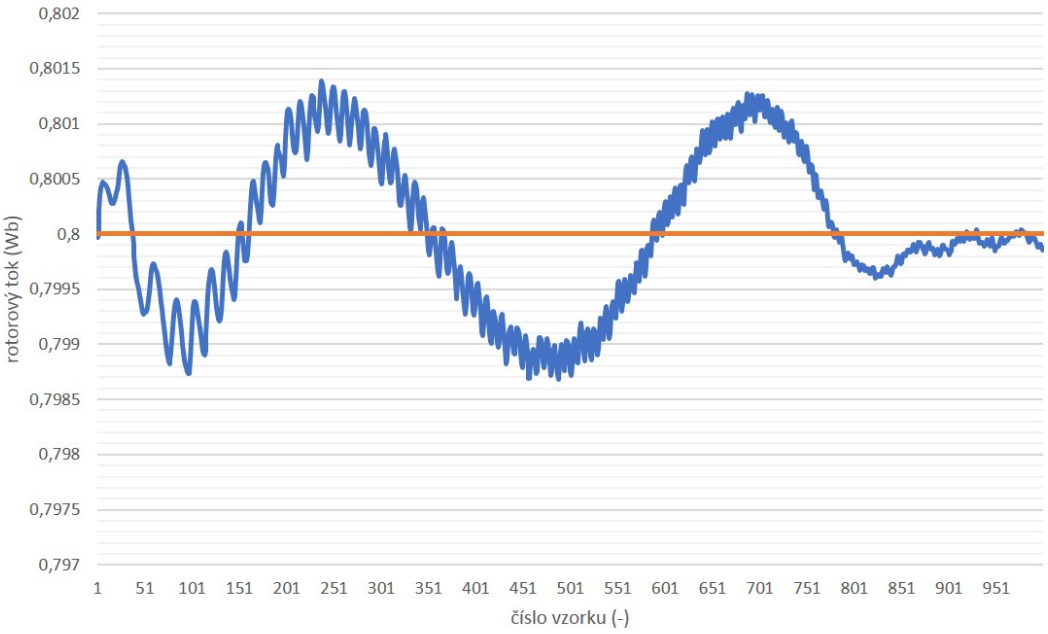
8.1.2 Regulované veličiny během skokové žádosti toku

V této kapitole jsou uvedeny záznamy dokumentující průběhy žádaných a skutečných hodnot rotorového toku a tokotvorné složky proudu během procesu nabuzení, tj. žádosti na skok toku. Nastavené podmínky:

- skok rotorového toku: $0 \rightarrow 0,8$ Wb,
- napětí ve stejnosměrném meziobvodu střídače: 540 V,
- perioda odesílání vzorků proměnných na RTM: 1 ms.



Obrázek 8.3. Průběh momentotvorného proudu při rozběhu, i_{1d}^{*} (oranžová), i_{1d} (modrá)

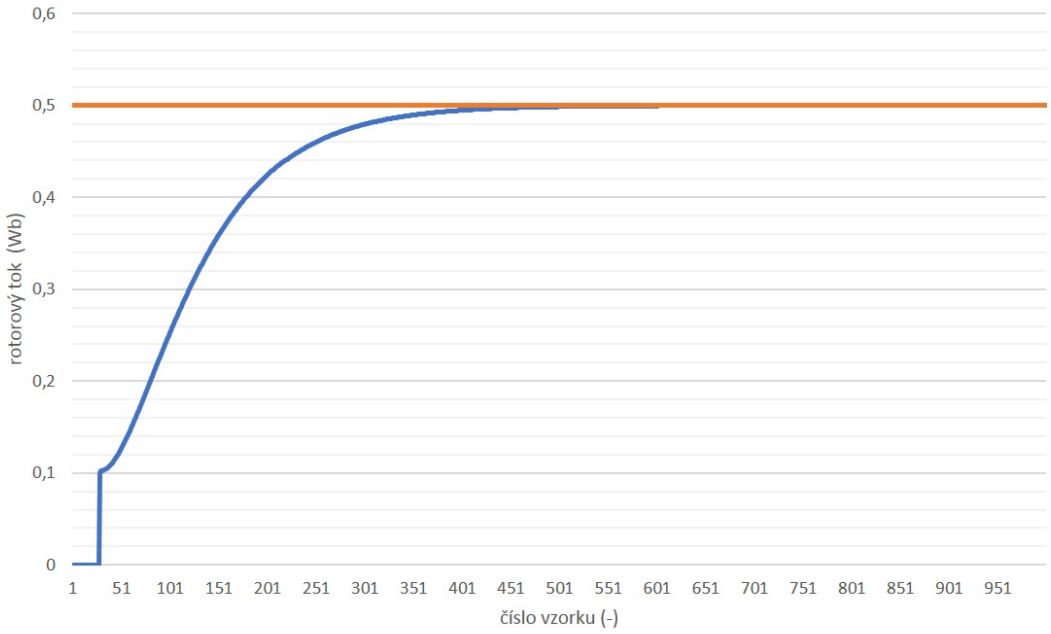


Obrázek 8.4. Průběh momentotvorného proudu při rozběhu, $|\Psi_2^{*}|$ (oranžová), $|\Psi_2|$ (modrá)

Na obr. 8.5 je vyobrazen průběh žádané a skutečné hodnoty rotorového toku. Okolo 25. vzorku je zde patrný skok skutečné hodnoty toku. Ten je způsoben implementovaným opatřením, které zahazuje nově vypočtené složky rotorového toku ve stavu, kdy motorem protéká nulový proud a naměřené hodnoty proudu tedy odpovídají pouze šumu. V těchto případech je vyhodnocován rotorový tok a transformační úhel nesprávně, na což regulátory reagují nenulovými akčními veličinami. Po tomto skoku dochází k plynulému náběhu rotorového toku směrem k žádané hodnotě, která je do-

sažena bez překmitu. Jelikož je rotorový tok relativně pomalu se měnící veličina, není zde nastavení parametrů regulátoru tolik kritické, jako např. u regulátorů proudu.

Obr. 8.6 pak znázorňuje průběh požadované a reálné hodnoty tokotvorné složky proudu. Opatření pro nulování toku způsobí, že na začátku je žádána maximální hodnota proudu, která ale po dosažení zlomové hodnoty rychle klesá a dále pak pozvolna narůstá v souladu s vyhodnocovanou regulační odchylkou a následně dosahuje hodnoty, která odpovídá ustálenému stavu. Rovněž je vidět dobrá schopnost regulátoru tokotvorného proudu sledovat hodnotu požadovanou. Přestože jsou konstanty proudových regulátorů nastaveny stejně, neprojevuje se u tokotvorného regulátoru narozdíl od momentotvorného po skokové žádosti překmit regulované veličiny.



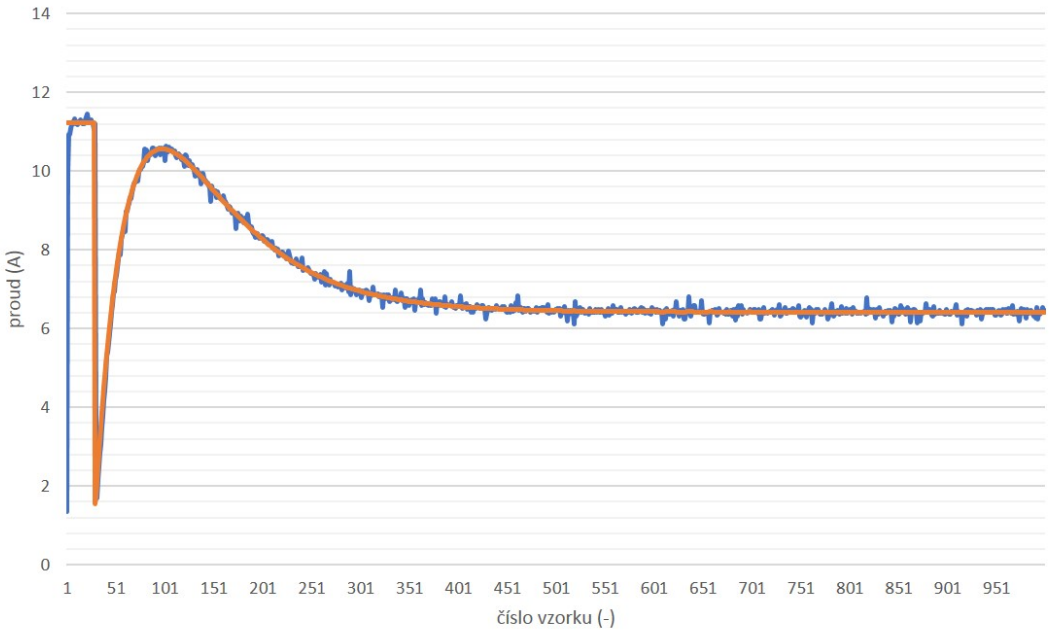
Obrázek 8.5. Odezva na skok rotorového toku, $|\Psi_2^*|$ (oranžová), $|\Psi_2|$ (modrá)

8.2 Porovnání numerických metod pro řešení proudového a napěťového modelu

Chování asynchronního motoru je možné popsat spojitými nebo diskrétními[1] modely. V této práci byl zvolen spojitý popis v časové oblasti, který je realizován obyčejnými diferenciálními rovnicemi. V kapitole 3.1 byly představeny dva nejpoužívanější modely asynchronního motoru – proudový tzv. I-n model a napěťový tzv. U-I model. Pro praktickou realizaci vektorového řízení do řídicího počítače je při zvoleném spojitém popisu nutné řešit tyto diferenciální rovnice numericky. Existuje mnoho numerických metod, které se liší přesností aproximovaného řešení ve smyslu lokální a globální diskretizační chyby, výpočetní náročností ve smyslu provedených kroků pro získání aproximace apod.

Tato kapitola si klade za cíl porovnat vliv dvou numerických metod představených v kap. 7.3.1 na chování pohonu během rozběhu – jednoduché explicitní Eulerovy metody a složitější, zato ale o mnoho robustnější metody RK4. Předmětem řešení jsou rovnice

[1] Více o diskrétním popisu např. v [7].



Obrázek 8.6. Průběh tokotvorné složky proudu při skokové žádosti toku, i_{1d}^{*} (oranžová), i_{1d} (modrá)

proudového modelu dané soustavou (3.5) a rovnice napěťového modelu ve tvaru (3.10). Aplikace metody RK4 na rovnice proudového modelu byla ukázána v kap. 7.3.2 (aplikace pro použití při řešení rovnic napěťového modelu by byla analogická). Důležité je použití „triku“, kdy jsou neznámé hodnoty uprostřed vzorkovací periody, nutné pro správný chod metody RK4, odhadovány jako aritmetický průměr z hodnot krajních. Implementace Eulerovy metody je přímočará a nebude zde diskutována.

Podmínky během experimentálního porovnání:

- žádaná hodnota rotorového toku: 0,8 Wb,
- skok otáček: $0 \rightarrow 100$ s^{-1},
- napětí ve stejnosměrném meziobvodu střídače: 540 V,
- perioda odesílání vzorků proměnných na RTM: 0,5 ms,
- zátěžný moment: nulový (pohon naprázdno).

Pro názorné porovnání byl krok řešení modelů desetkrát zvýšen, tj. k výpočtům nových hodnot docházelo jednou za deset vzorkovacích period. Z důvodu náchylnosti ke kmitání napěťového modelu byly pak desetkrát sníženy konstanty tokového regulátoru.

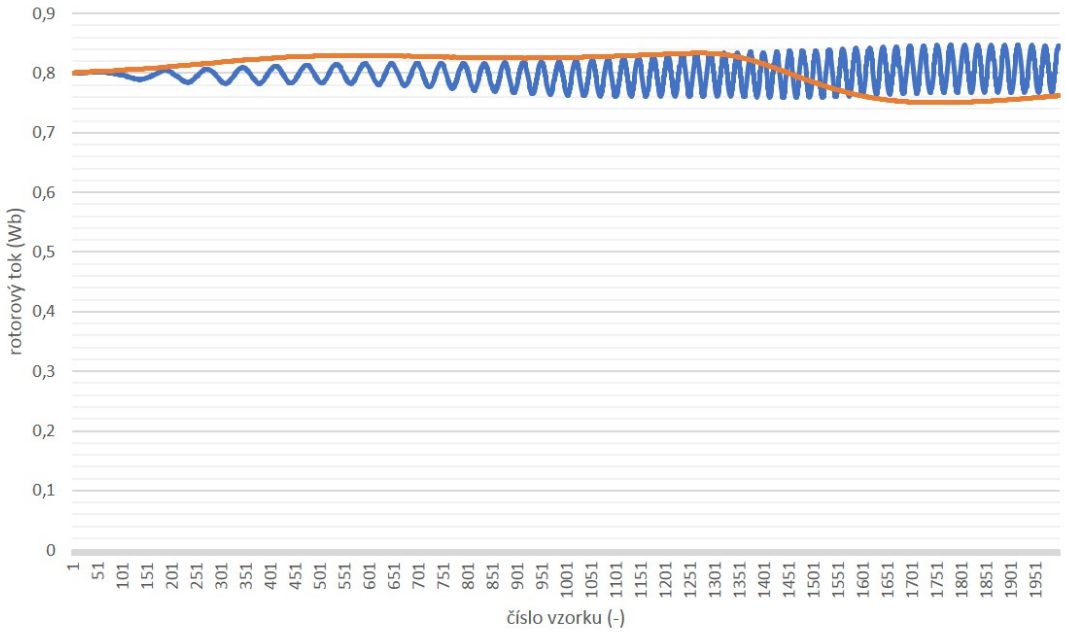
8.2.1 Řešení proudového modelu

Rovnice (3.5) představují z matematického hlediska poměrně komplikovanou soustavu obyčejných lineárních diferenciálních rovnic s nekonstantními koeficienty (člen násobený otáčkami). Lze tedy očekávat výraznější závislost přesnosti řešení na zvolené numerické metodě a velikosti kroku.

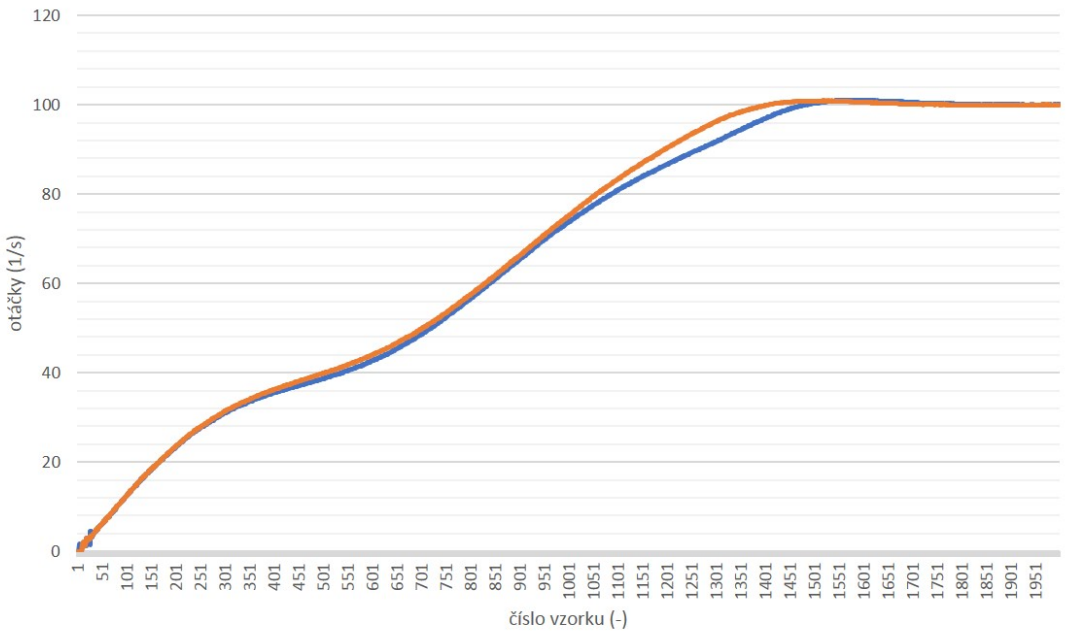
Naměřené průběhy vybraných veličin jsou na obr. 8.7, 8.9 a 8.8. Je vidět, že metoda RK4 poskytuje mnohem hladší výstupní průběhy a rychlejší rozběh. Velikost rotorového toku získaného Eulerovou metodou poněkud osciluje, což může značit nepřesnost aproximovaného řešení. Tyto oscilace se zároveň projevují na průběhu vypočteného momentu. Bohužel reálný moment stroje při rozběhu není znám, neboť ho v době tvorby práce a pořizování záznamů nebylo možné efektivně změřit.

Na obr. 8.10 je pro srovnání ukázán výpočet rotorového toku oběma metodami při původním kroku metody rovném vzorkovací periodě. Pro lepší rozlišení jsou průběhy přiblíženy. Je patrné, že metoda RK4 i při původním kroku dává hladší a stabilnější výsledky.

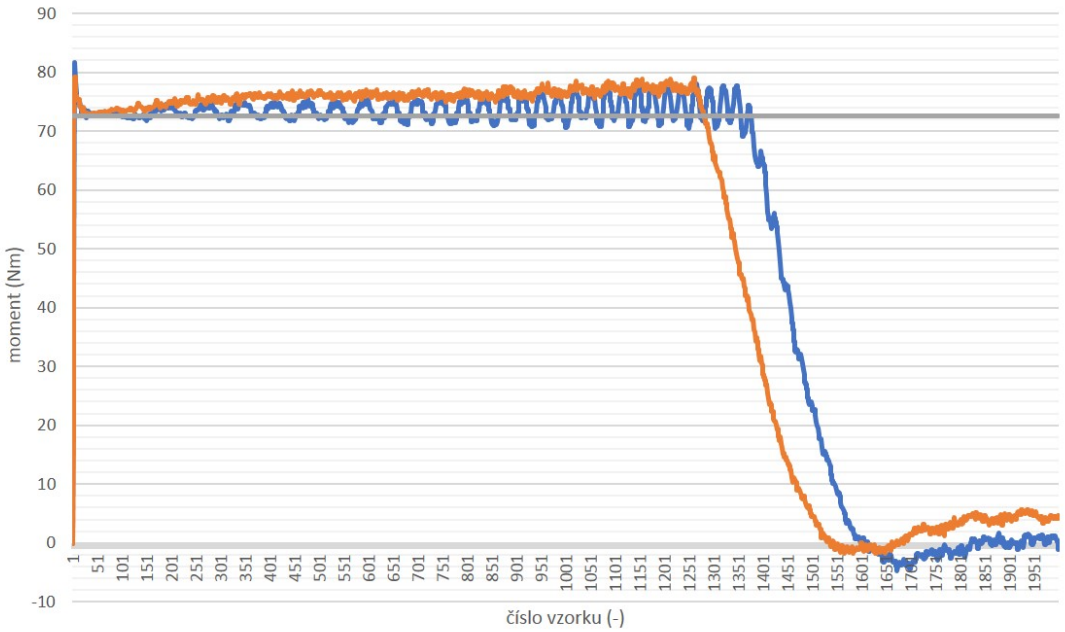
Na závěr ještě uveďme, že u Eulerovy metody byla nezávisle na nastaveném kroku pro výpočet nové hodnoty $\Psi_{2\beta}$ použita nově vypočtená hodnota $\Psi_{2\alpha}$, tj. výpočet složky α probíhal jako první.



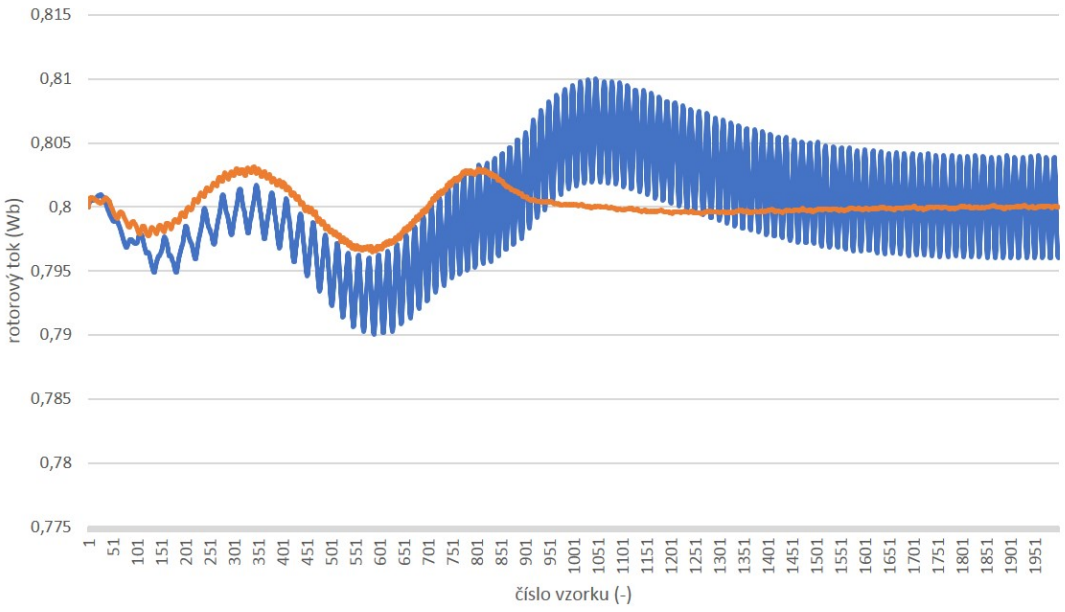
Obrázek 8.7. Řešení rovnic proudového modelu – toky, RK4 (oranžová), Euler (modrá)



Obrázek 8.8. Řešení rovnic proudového modelu – otáčky, RK4 (oranžová), Euler (modrá)



Obrázek 8.9. Řešení rovnic proudového modelu – momenty, RK4 (oranžová), Euler (modrá), teoreticky vypočtený rozběhový moment (šedá)



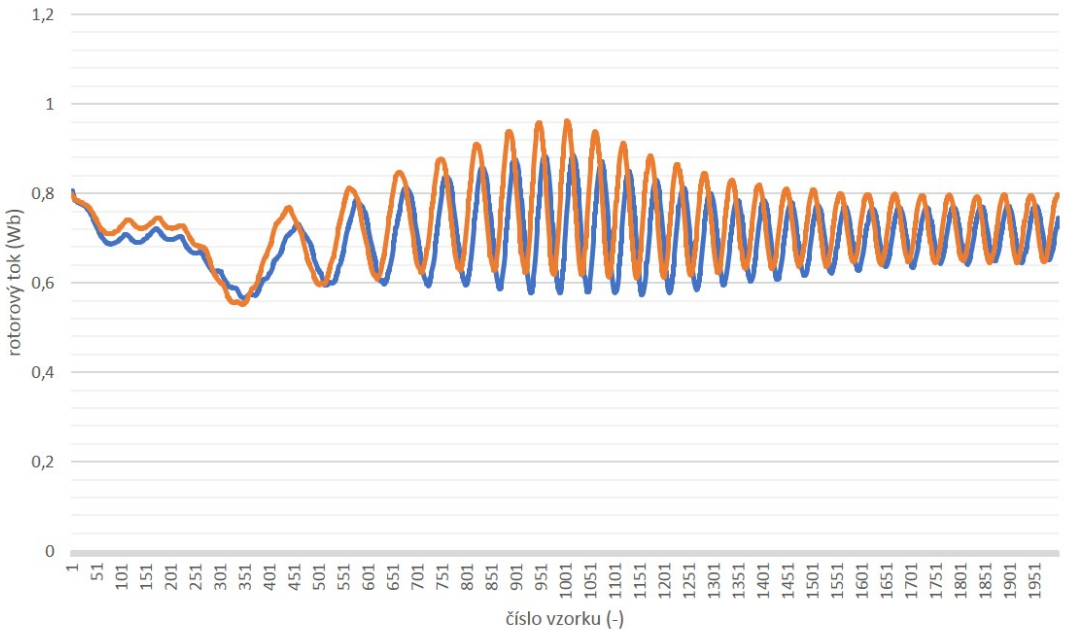
Obrázek 8.10. Řešení rovnic proudového modelu (původní krok výpočtu) – toky, RK4 (oranžová), Euler (modrá)

8.2.2 Řešení napěťového modelu

Na první pohled je patrné, že napěťový model daný soustavou (3.10) představuje oproti proudovému modelu mnohem jednodušší sadu rovnic, neboť se jedná o dvě na sobě nezávislé obyčejné lineární diferenciální rovnice s konstantními koeficienty doplněné dvěma rovnicemi algebraickými. Z čistě teoretického hlediska by tak měl napěťový model dávat mnohem stabilnější a přesnější řešení.

Reálná situace je ale poněkud odlišná. Předně je zde problém vzniku stejnosměrné složky, která se vyhodnocuje při nesprávných počátečních podmínkách, tedy prakticky vždy. Ta je tlumena koeficientem γ, což ale vnáší do výpočtu statorového toku určitou chybu fáze. Další problém je přesnost modelu při nízkých otáčkách, kdy se negativně projevuje vliv úbytku napětí na statorovém odporu. V neposlední řadě je obtížné vůbec přesně určit velikost statorového napětí. Jednou z možností je vyhodnocovat toto napětí ze změřeného stejnosměrného napětí v meziobvodu a známého stavu řídicích signálů pro budiče. V této práci byl zvolena strategie, kdy vstup do napěťového modelu tvoří složky žádaného vektoru napětí v systému $\alpha\beta$. Jenže vlivem vložených ochranných dob, zpoždění budičů, konečné rychlosti sepnutí a vypnutí součástek a jejich nenulového odporu v sepnutém stavu se bude skutečné napětí vždy o něco lišit od toho žádaného.

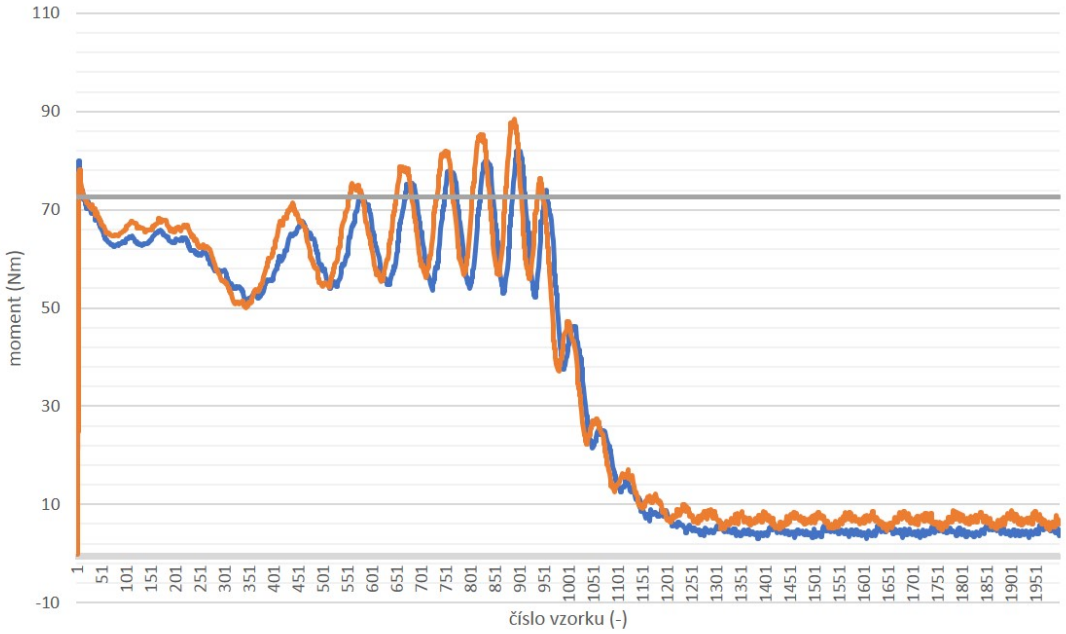
Na obr. 8.11, 8.12 a 8.13 jsou zaznamenány průběhy při řízení založeném na napěťovém modelu. Pohon se sice podařilo úspěšně rozeběhnout a dostat do ustáleného stavu, nicméně je vidět, že vyhodnocovaný tok, resp. moment poněkud osciluje, a to navíc se střední hodnotou, která se poměrně dost liší od požadované. Z pohledu numerických metod lze říci, že metoda RK4 se blíží žádané hodnotě o něco více. Zajímavý je pak náběh otáček, kde rovněž o trochu lépe vychází RK4, a který je hladší a pohon dosahuje žádané hodnoty rychleji než v případě řízení proudovým modelem, což ale nekoresponduje s vyhodnocenýmii veličinami na dvou předchozích obrázcích. To může značit rozdíl mezi vypočtenými a reálnými veličinami, případně rozdílný vliv regulátorů na proudový a napěťový model.



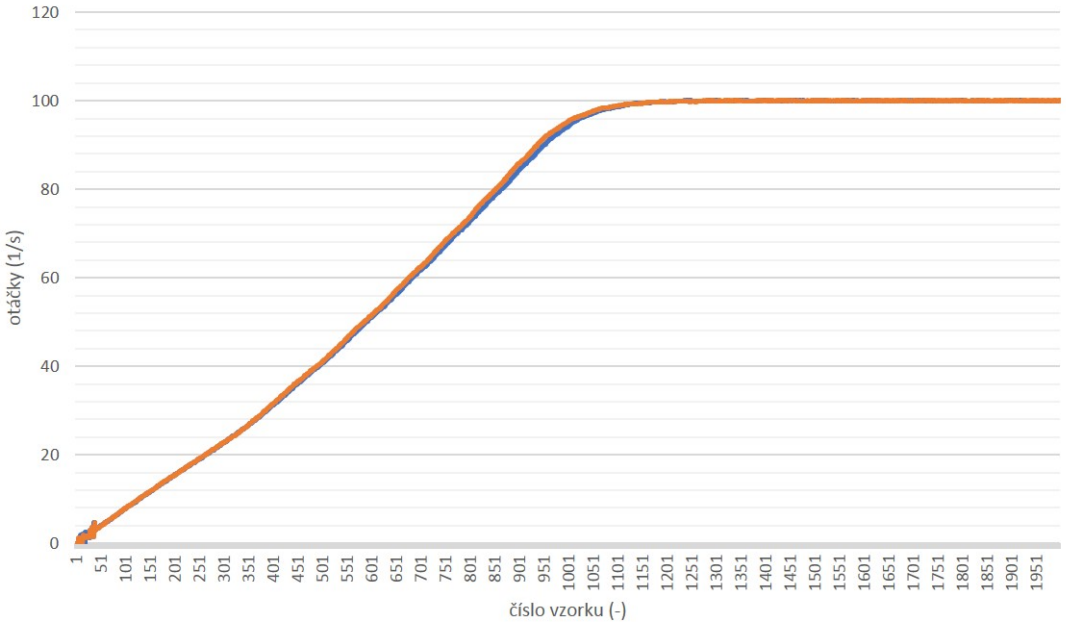
Obrázek 8.11. Řešení rovnic napěťového modelu – toky, RK4 (oranžová), Euler (modrá)

8.2.3 Výpočetní náročnost

Na závěr této kapitoly je v tab. 8.1 uvedena časová náročnost řešení matematických modelů jednotlivými numerickými metodami.



Obrázek 8.12. Řešení rovnic napěťového modelu – momenty, RK4 (oranžová), Euler (modrá), teoreticky vypočtený rozběhový moment (šedá)



Obrázek 8.13. Řešení rovnic napěťového modelu – otáčky, RK4 (oranžová), Euler (modrá)

Eulerova metoda		Metoda RK4	
I-n model	1,9 μs	I-n model	4,2 μs
U-I model	0,73 μs	U-I model	3,53 μs

Tabulka 8.1. Časová náročnost numerických metod při řešení matematických modelů

Kapitola 9
Závěr

9.1 Zhodnocení dosažených výsledků

V rámci této diplomové práce byla navržena struktura vektorového řízení, která byla následně naprogramována do digitálního signálového procesoru TMS320F28335 od společnosti Texas Instruments. Realizované vektorové řízení pracuje s oddělenou regulací tokotvorné a momentotvorné složky proudu. Jejich skutečné hodnoty jsou získávány transformací dvou měřených fázových proudů pomocí Clarkové a následné Parkovy transformace do souřadnicového systému dq rotujícího synchronní rychlostí. Transformační úhel je vypočítáván ze složek rotorového toku, které jsou obdrženy z řešení soustavy diferenciálních rovnic tzv. proudového I-n modelu metodou Runge-Kutta 4. řádu. Výstupem regulátorů proudu je pak omezený a odvazbený žádaný napěťový vektor v dq, který je transformován do souřadnicového systému $\alpha\beta$ spojeného se statorem a přiveden na vstup modulátoru, který pomocí modulace prostorového vektoru vygeneruje signály pro příslušně tranzistory dvouúrovňového napěťového střídače. Předmětem regulace je pak asynchronní motor o výkonu 12 kW, který je součástí Ward-Leonardova soustrojí.

Program byl psán v jazyce C ve vývojovém prostředí Code Composer Studio, jehož součástí je i rozhraní pro debugging, které bylo využíváno během ladění. Součástí této práce bylo i nezbytné seznámení se s dokumentací použitého signálového procesoru, která sloužila jako poklad pro jeho konfiguraci a inicializaci. Hlavními využívanými periferiemi byl analogově-číslicový převodník, ePWM modul a eQEP jednotka.

Dalším krokem po inicializaci bylo programové zpracování signálů z čidel. Jedná se o měření fázových proudů motoru a stejnosměrného napětí v meziobvodu střídače a měření otáček inkrementálním kvadraturním enkodérem. Pro propojení procesoru s měničem a čidly je použita deska, která byla poskytnuta vedoucím práce. Ta zajišťuje zejména přizpůsobení napěťových úrovní a dále pak vhodnou úpravu analogových signálů.

Pohon lze provozovat v několika vytvořených regulačních smyčkách. Jedná se o smyčku otáčkovou, otáčkovou s rampou otáček ve tvaru „S“ křivky, momentovou se zadáváním žádaného proudu i_q a polohovou. Pro regulaci příslušných veličin jsou použity konvenční PI regulátory s anti-windup zapojením, jejichž parametry byly stanovovány experimentálně. Výjimku tvoří regulátor polohy, který je realizován jako nespojitý třípolohový regulátor s hysterezí. Pro demonstrační účely a porovnání přímého připnutí na síť je dále možné pohon připojit na 50Hz síť, která je simulována pomocí měniče.

Velice cenným nástrojem, který byl používán během celého vývoje a zejména pak při ladění regulátorů je Real-Time Monitor, což je software vyvíjený na katedře elektrotechnologie, který umožňuje vizualizaci deklarovaných proměnných a jejich modifikaci. Real-Time Monitor zahrnuje desktop Java aplikaci a knihovny pro F28335.

Součástí algoritmu vektorového řízení je rovněž softwarová nadproudá a otáčková ochrana a dále pak blokování z pohledu motoru nepřípustných stavů.

Pro ovládání pohonu bylo v cloudovém prostředí GUI Composer od společnosti Texas Instruments vytvořeno grafické uživatelské rozhraní. Z něho je přímo možné zadávat žádané hodnoty, přepínat mezi regulačními smyčkami a taktéž sledovat vybrané veličiny.

Na konec byly provedeny záznamy přechodových dějů a dále pak porovnání jednoduché explicitní Eulerovy metody a metody Runge-Kutta 4. řádu při řešení rovnic napěťového a proudového modelu, a to i z hlediska výpočetní náročnosti. Pro názornost byl při výpočtech desetkrát zvýšen krok, tj. modely byly počítány jednou za deset vzorkovacích period. U proudového modelu dává RK4 mnohem hladší výstupy než Eulerova metoda, která poněkud osciluje, navíc bylo zjištěno, že model je celkově stabilnější a není tolik citlivý na okolní parametry. Pohon rovněž v případě použití RK4 dosáhne dříve požadovaných otáček.

U napěťového modelu je situace složitější, neboť ten dává oscilující výstupy při řešení oběma metodami. Je ale vidět, že při použití RK4 se střední hodnota těchto průběhů více blíží žádané hodnotě. Paradoxně se pohon při řízení napěťovým modelem rozeběhne rychleji, což by mohlo opět značit nepřesnost obdržených výsledků, kdy je možné, že vypočtené hodnoty se mohou celkem zásadně lišit od skutečnosti. Tyto odchylky pravděpodobně vznikají nepřesným vyhodnocováním statorového napětí.

Ná závěr lze shrnout, že se podařilo vyvinout a implementovat stabilní algoritmus vektorového řízení asynchronního motoru vyššího výkonu, pomocí něhož lze dosáhnout dobré dynamiky regulovaného pohonu.

9.2 Prostor pro další vylepšení

Celý systém samozřejmě skýtá další možnosti vylepšení. Předně je to vytvoření desktopové ovládací aplikace a protokolu pro komunikaci počítače s procesorem, neboť řešení v podobě vytvořeného GUI Composer rozhraní nelze považovat za ideální.

Dalším vhodným rozšířením by byla analýza nelinearit střídače a možností jejich kompenzace – zejména pak vlivu ochranných dob či zpoždění budičů na výsledné napětí. Je možné, že právě tyto nelinearity vedou ke zkresleným výsledkům obdrženým z napěťového modelu. Správná činnost tohoto modelu je pak dále klíčem k bezsenzorovému vyhodnocování otáček.

Jako poslední lze uvést vhodnost kompenzace změn hodnot rotorového a statorového odporu vlivem teploty, resp. zatížení, což je další faktor, který negativně ovlivňuje výslednou přesnost matematických modelů.

Literatura

[1] VOŽENÍLEK, Petr, Vladimír NOVOTNÝ a Pavel MINDL. *Elektromechanické měniče*. 2. vyd. Praha: České vysoké učení technické v Praze, 2011. ISBN 978-80-01-04875-7.

[2] PAVELKA, Jiří a Jiří ZDĚNEK.*Elektrické pohony a jejich řízení*. V Praze: České vysoké učení technické, 2010. ISBN 9788001046425.

[3] JAVŮREK, Jiří. *Regulace moderních elektrických pohonů*. Praha: Grada, 2003. ISBN 80-247-0507-9.

[4] KOBRLE, Pavel. *Odvození matematického modelu asynchronního motoru a vektorově orientované metody regulace*. Podkladový materiál ke cvičením z předmětu „Elektrické pohony a trakce“. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2017.

[5] ZEMAN, Karel, Zdeněk PEROUTKA a Martin JANDA. „Automatická regulace pohonů s asynchronními motory“. Plzeň: Západočeská univerzita, 2004. ISBN 9788070433508.

[6] BAUER, Jan. *Compact Matrix Convertor Power and Control System Design and Verification*. Praha, 2014. Disertační práce. České vysoké učení technické v Praze, Fakulta elektrotechnická.

[7] QUANG, Nguyen P. a Jörg-Andreas DITTRICH. *Vector Control of Three-Phase AC Machines: System Development in the Practice*. 1. Aufl. Berlin, Heidelberg: Springer-Verlag, 2008. ISBN 9783540790280;3540790284;.

[8] HAVELKA, David. *Programové vybavení diagnostického systému pro trakční aplikace*. Praha, 2005. Disertační práce. České vysoké učení technické v Praze, Fakulta elektrotechnická.

[9] LEPKA, Jaroslav a Petr STEKL. *3-Phase AC Induction Motor Vector Control Using a 56F80x, 56F8100 or 56F8300 Device* [online]. Design of Motor Control Application. Application Note No. AN1930. Rev. 2, 2/2005. Freescale Semiconductor, 2004. [cit. 2018-03-6] Dostupné z:
`http://cache.freescale.com/files/product/doc/AN1930.pdf`

[10] QIAN, Cheng a YUAN Lei. *Vector Control of and Induction Motor based on a DSP*. Göteborg, Sweden, 2011. Master of Science Thesis. Chalmers University of Technology, Department of Energy and Environment, Division of Electric Power Engineering.

[11] YU, Zhenyu. *Space-Vector PWM With TMS320C24x/F24x Using Hardware and Software Determined Switching Patterns*[online]. Application Report No. SPRA524. Texas Instruments, 1999. [cit. 2018-02-21] Dostupné z:
`http://www.ti.com/lit/an/spra524/spra524.pdf`

[12] SPECTRUM DIGITAL, INC. *eZdsp F28335* [online]. Technical Reference. 510195-0001 Rev. C. November 2007. [cit. 2017-10-06]. Dostupné z:
`http://c2000.spectrumdigital.com/ezf28335/docs/ezdspf28335c_techref.pdf`.

[13] Texas Instruments. *TMS320F2833x, TMS320F2823x Digital Signal Controllers (DSCs) Data Manual* [online]. SPRS439N – JUNE 2007 – REVISED OCTOBER 2016. [cit. 2017-10-06]. Dostupné z:
http://www.ti.com/lit/ds/symlink/tms320f28335.pdf

[14] Texas Instruments. *TMS320x2833x, 2823x System Control and Interrupts* [online]. Reference Guide No. SPRUFB0D. September 2007 – Revised March 2010. [cit. 2017-10-07]. Dostupné z:
http://www.ti.com/lit/ug/sprufb0d/sprufb0d.pdf

[15] Texas Instruments. *TMS320x2833x, 2823x Enhanced Pulse Width Modulator (ePWM) Module* [online]. Reference Guide No. SPRUG04A. October 200 – Revised July 2009. [cit. 2017-10-07]. Dostupné z:
http://www.ti.com/lit/ug/sprug04a/sprug04a.pdf

[16] Texas Instruments. *TMS320x2833x, 2823x Enhanced Quadrature Encoder Pulse (eQEP) Module* [online]. Reference Guide No. SPRUG05A. August 2008 – Revised December 2008. [cit. 2017-10-11]. Dostupné z:
http://www.ti.com/lit/ug/sprug05a/sprug05a.pdf

[17] Texas Instruments. *TMS320x2833x Analog-to-Digital Converter (ADC) Module* [online]. Reference Guide No. SPRU812A. September 2007 – Revised October 2007. [cit. 2017-10-08]. Dostupné z:
http://www.ti.com/lit/ug/spru812a/spru812a.pdf

[18] XDS100 - Texas Instruments Wiki. *Texas Instruments Wiki* [online]. [cit. 2017-10-06]. Dostupné z:
http://processors.wiki.ti.com/index.php/XDS100

[19] CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE) — TI.com. *Analog, Embedded Processing, Semiconductor Company, Texas Instruments - TI.com* [online]. © 1995-2017. [cit. 2017-10-06]. Dostupné z:
http://www.ti.com/tool/CCSTUDIO

[20] Texas Instruments. *C2000Ware Quick Start Guide* [online]. SPRUI46 – January 2017. [cit. 2017-10-21]. Dostupné z:
http://www.ti.com/lit/ml/sprui46/sprui46.pdf

[21] Texas Instruments. *C28x Floating Point Unit fastRTS Library*. Module User's Guide No. SPRCA75. C28x Foundation Software. V1.00. June 16, 2010.

[22] Dou, M & Wu, G & Shi, C & Liu, Xiaoqing. (2014).*High accuracy speed measurement for rotary motor based on DSP*. 6. 205-209. [cit. 2017-3-12]. Dostupné z:
http://www.jocpr.com/articles/high-accuracy-speed-measurement-for-rotary-motor-based-on-dsp.pdf

[23] OHMAE, Tsutomu et al. A Microprocessor-Controlled High-Accuracy Wide-Range Speed Regulator for Motor Drives. *IEEE Transactions on Industrial Electronics*. 1982, vol. IE-29, no. 3, s. 207-211. ISSN 0278-0046.

[24] Pu, Jian-tao & Wang, Hui. (2012). *A novel variable M/T method for speed measurement with high precision in a wide speed range.* 10.2991/emeit.2012.411. [cit. 2017-3-12]. Dostupné z:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1000.1510&rep=rep1&type=pdf

[25] VONDRÁK, Vít a Lukáš POSPÍŠIL. *NUMERICKÉ METODY I* [online]. Vysoká škola báňská – Technická univerzita Ostrava a Západočeská univerzita v Plzni,

2011. [cit. 2018-02-07]. Dostupné z:
http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/numericke_metody.pdf

[26] HABALA, Petr. *Ordinary Differential Equations and Numerical Analysis* [online]. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2016. [cit. 2018-02-07]. Dostupné z:
https://math.feld.cvut.cz/habala/teaching/veci-ODE/dnbook.pdf

[27] ZDĚNEK, Jiří. *Digital Controller* [online]. Podkladový materiál k přednáškám z předmětu „Digitální řízení elektrických pohonů". České vysoké učení technické v Praze, Fakulta elektrotechnická, 2017. [cit. 2018-03-6]. Dostupné z:
http://motor.feld.cvut.cz/sites/default/files/predmety/BE1M14DEP-2017-10z-digital-controller_0.pdf

[28] BALÁTĚ, Jaroslav. *Automatické řízení*. 2., přeprac. vyd. Praha: BEN, 2004. ISBN 9788073001483.

[29] HLAVA, Jaroslav. *Nespojité (dvou- a třípolohové) regulátory* [online]. Technická univerzita v Liberci; Fakulta mechatroniky, informatiky a mezioborových studií. [cit. 2018-03-06]. Dostupné z:
www.fm.tul.cz/esf0247/index.php?download=1030

[30] GUI Composer 2 User's Guide *Analog, Embedded Processing, Semiconductor Company, Texas Instruments - TI.com* [online]. © 1995-2017. [cit. 2017-10-15]. Dostupné z:
https://dev.ti.com/gc/designer/help/UsersGuide/index.html#Introduction

[31] GUI Composer 2: Getting Started*Analog, Embedded Processing, Semiconductor Company, Texas Instruments - TI.com* [online]. © 1995-2017. [cit. 2017-10-15]. Dostupné z:
https://dev.ti.com/gc/designer/help/Tutorials/GettingStarted/index.html

Příloha A
Značení a symbolika

A.1 Značení obecných fyzikálních veličin

$x, x(t)$	okamžitá hodnota
x^*	žádaná hodnota
$\underline{x}$	prostorový vektor
$x(k)$	diskrétní proměnná v časovém okamžiku kT

A.2 Použité symboly

f	(s^{-1})...frekvence napájecího napětí, resp. proudu
i, I	(A)...proud
L_1	(H)...celková statorová indukčnost
L_2	(H)...celková rotorová indukčnost
$L_{1\sigma}$	(H)...statorová rozptylová indukčnost
$L_{2\sigma}$	(H)...rotorová rozptylová indukčnost
L_m	(H)...magnetizační indukčnost
L_r	(H)...vlastní indukčnost jedné fáze rotorového vinutí
L_s	(H)...vlastní indukčnost jedné fáze statorového vinutí
m_i	(Nm)...vnitřní elektromechanický moment
M	(H)...vzájemná indukčnost odpovídajících si fází statoru a rotoru
M_r	(H)...vzájemná indukčnost dvou fází rotorového vinutí
M_s	(H)...vzájemná indukčnost dvou fází statorového vinutí
n	(min^{-1})...otáčivá rychlost rotoru
p_p	(-)...počet pólpárů
P	(W)...výkon
P_δ	(W)...výkon ve vzduchové mezeře stroje
R	(Ω)...elektrický odpor
R_1	(Ω)...statorový elektrický odpor
R_2	(Ω)...rotorový elektrický odpor
s	(-)...skluz asynchronního motoru
t	(s)...čas
t_D	(s)...ochranná doba
T	(s)...vzorkovací perioda
T_p	(s)...spínací perioda
u, U	(V)...napětí
U_DC	(V)...napětí ve stejnosměrném meziobvodu střídače
X_1	(Ω)...statorová reaktance
X_2	(Ω)...rotorová reaktance

$X_{1\mathrm{m}}$	(Ω)...hlavní reaktance statoru
$X_{2\mathrm{m}}$	(Ω)...hlavní reaktance rotoru
$X_{1\sigma}$	(Ω)...rozptylová reaktance statoru
$X_{2\sigma}$	(Ω)...rozptylová reaktance rotoru
X_{m}	(Ω)...magnetizační reaktance
θ	(rad)...okamžitý úhel mezi souřadnicovými systémy $\alpha\beta$ a dq
ϑ	(rad)...okamžitý úhel natočení rotoru oproti statoru
ϑ_{k}	(rad)...okamžitý úhel mezi stojícím a obecným rotujícím sourařdnicovým systémem
σ	(-)...činitel rozptylu
τ_{r}	(s)...rotorová časová konstanta
Ψ	(Wb)...spřažený magnetický tok
Ψ_1	(Wb)...statorový spřažený magnetický tok
Ψ_2	(Wb)...rotorový spřažený magnetický tok
ω	(s^{-1})...elektrická úhlová rychlost rotoru
ω_{k}	(s^{-1})...elektrická úhlová rychlost obecného rotujícího souřadnicového systému
ω_{s}	(s^{-1})...synchronní elektrická úhlová rychlost
ω_{slip}	(s^{-1})...skluzová elektrická úhlová rychlost
Ω	(s^{-1})...mechanická úhlová rychlost rotoru

abc	třífázový systém statoru
a, b, c	označení jednotlivých fází statoru
ABC	třífázový systém rotoru
A, B, C	označení jednotlivých fází rotoru
$\alpha\beta$	ortogonální souřadnicový systém svázaný se statorem
dq	ortogonální souřadnicový systém rotující synchronní rychlostí
xy	obecný ortogonální souřadnicový systém
e	Eulerovo číslo e=2,718...
j	imaginární jednotka $\mathrm{j}^2 = -1$
$\Re$	reálná část komplexního čísla
$\Im$	imaginární část komplexního čísla
$\mathbf{a}$, $\mathbf{a^2}$	operátory natočení v komplexní rovině
p	operátor Laplaceovy transformace

A.3 Použité zkratky

AD	Analogově-Digitální
ADC	Analog-to-Digital Converter – analogově-číslicový převodník
AM	Asynchronní Motor
CCS	Code Composer Studio – integrované vývojové prostředí
CPU	Central Processing Unit – centrální procesorová jednotka
DC	Direct Current – stejnosměrná hodnota
DSP	Digital Signal Processor – digitální signálový procesor
GUI	Graphical User Interface – grafické uživatelské rozhraní
H/W	Hardware
IDE	Integrated Development Enviroment – integrované vývojové prostředí
IGBT	Isolated Gate Bipolar Transistor – bipolární tranzistor s izolovaným hradlem
JTAG	Join Test Action Group – systém pro komunikaci se zařízením a jeho vzdálené testování

LSB	Least Significant Bit – nejméně významný bit
MSB	Most Significant Bit – nejvýznamnější bit
PC	Personal Computer – osobní počítač
PI	Proporciálně-Integrační
PS	Proporciálně-Sumační
PWM	Pulse-Width Modulation – pulzně šířková modulace
QEP	Quadrature Encoder Pulse
RTM	Real-Time Monitor – software pro monitorování proměnných v reálném čase
S&H	Sample and Hold – vzorkovací obvod
S/W	Software
TI	Texas Instruments

Příloha B
Laboratorní pracoviště

Obrázek B.1. Laboratorní pracoviště

B.1 Ward-Leonardovo soustrojí

Jmenovitý výkon	12 kW
Jmenovitý proud	22 A
Jmenovité napětí hvězda/trojúhelník	380/220 V
Jmenovitá frekvence	50 Hz
Jmenovitý účiník	0,8
Jmenovité otáčky	1 400 min^{-1}

Tabulka B.1. Štítkové parametry asynchronního motoru

R_1	0,358 Ω
R'_2	0,354 Ω
$L_{1\sigma}$	0,002 27 H
$L'_{1\sigma}$	0,002 27 H
R_{Fe}	202,1 Ω
L_{m}	0,077 9 H

Tabulka B.2. Změřené parametry náhradního schématu asynchronního motoru

R_1	0,358 Ω
R_2	0,354 Ω
L_1	0,080 19 H
L_2	0,080 19 H
L_{m}	0,077 9 H

Tabulka B.3. Vypočtené parametry asynchronního motoru používané matematickým modelem

Jmenovitý rotorový tok	0,877 Wb
Jmenovitá mechanická úhlová rychlost rotoru vypočtená ze jmenovitých otáček	146,6 s^{-1}
Jmenovitá mechanická úhlová rychlost rotoru vypočtená ze jmenovité frekvence	157,07 s^{-1}
Jmenovitá složka momentotvorného proudu	31,11 A
Jmenovitá složka tokotvorného proudu	11,28 A
Jmenovitý moment vypočtený ze štítkových hodnot	81,85 Nm

Tabulka B.4. Vypočtené jmenovité hodnoty asynchronního motoru

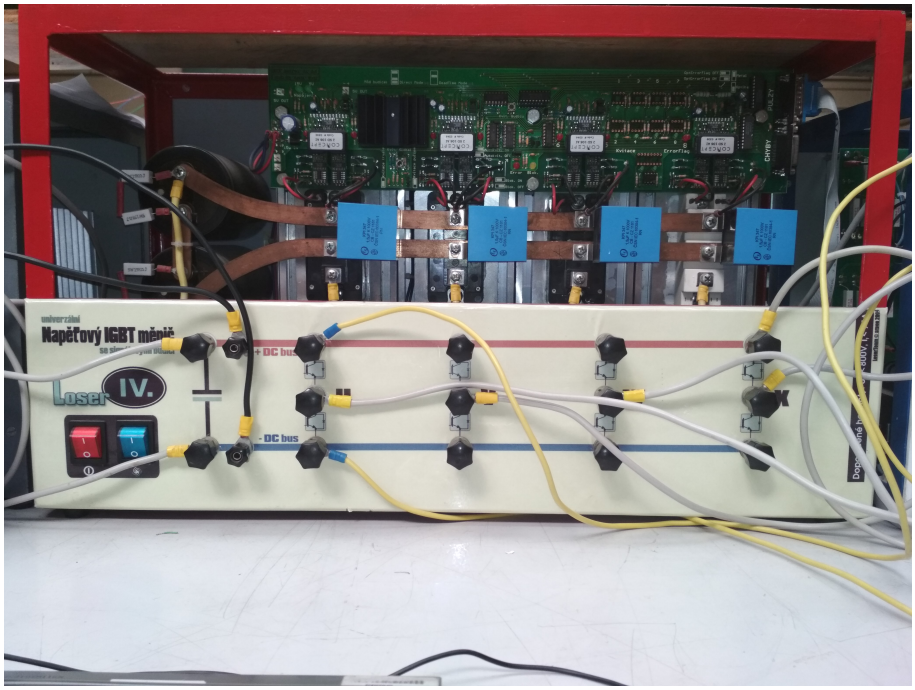
Jmenovitý výkon	8,8 kW
Jmenovitý proud	38,3 A
Jmenovité napětí	320 V
Jmenovitý budicí proud	2,8 A
Jmenovité budicí napětí	110 V
Jmenovité otáčky	1 400 min^{-1}

Tabulka B.5. Štítkové parametry stejnosměrného stroje

B.2 Napájecí část motoru

Řízený asynchronní motor je napájen z laboratorního univerzálního napěťového IGBT měniče Loser se signálovými budiči, jehož fotografie je na obr. B.2. První tři výstupy zleva, ke kterým jsou připojeny fáze asynchronního motoru, jsou osazeny IGBT moduly CM100DY-24NF od společnosti Mitsubishi Electric. Jedná se o integrovaný polomůstek se zabudovanými antiparalelními diodami. Maximální kolektorový proud IGBT tranzistorů je 100 A, maximální napětí kolektor-emitor 1,2 kV. Čtvrtý výstup, k němuž je připojen brzdný odpor, je osazen IGBT modulem SKM50GB12T4 od společnosti Semikron, který dovoluje maximální kolektorový proud 81 A a maximální napětí kolektor-emitor pak rovněž 1,2 kV.

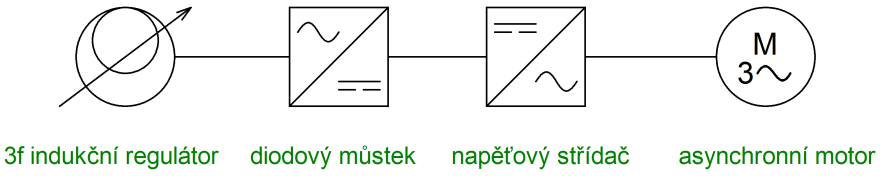
Jako budič pro tranzistory je použit integrovaný obvod 2SD106AI-17 od společnosti Concept, který pracuje s napětím 15 V. Ovládací signály pro budiče generované uvnitř DSP jsou napěťově upraveny propojovací deskou a pomocí 15pinového standardního konektoru CANNON připojeny k měniči. Vnitřní obvody měniče mimo jiné umožňují



Obrázek B.2. Napájecí měnič

nastavení budiče do direct nebo dead-time módu (vytváření ochranných dob pomocí RC sítě), generaci poruchových signálů a rovněž zajišťují zkratovou saturační ochranu

Stejnosměrný meziobvod střídače je tvořen dvěma do série zapojenými elektrolytickými kondenzátory K01450472 od společnosti Kendeil. Kondenzátory disponují kapacitou 4 700 μF a jsou určeny pro stejnosměrné napětí do 450 V. Aby bylo možné dle potřeb měnit hodnotu napětí v meziobvodu, je střídač napájen z třífázového indukčního regulátoru přes diodový můstek. Schéma napájecí části motoru je znázorněno na obr. B.3. Maximální doporučené stejnosměrné napětí měniče je 800 V, maximální doporučený fázový proud pak 75 A.



Obrázek B.3. Schéma napájecí části motoru

B.3 Měřicí hardware

B.3.1 Měření napětí a proudů

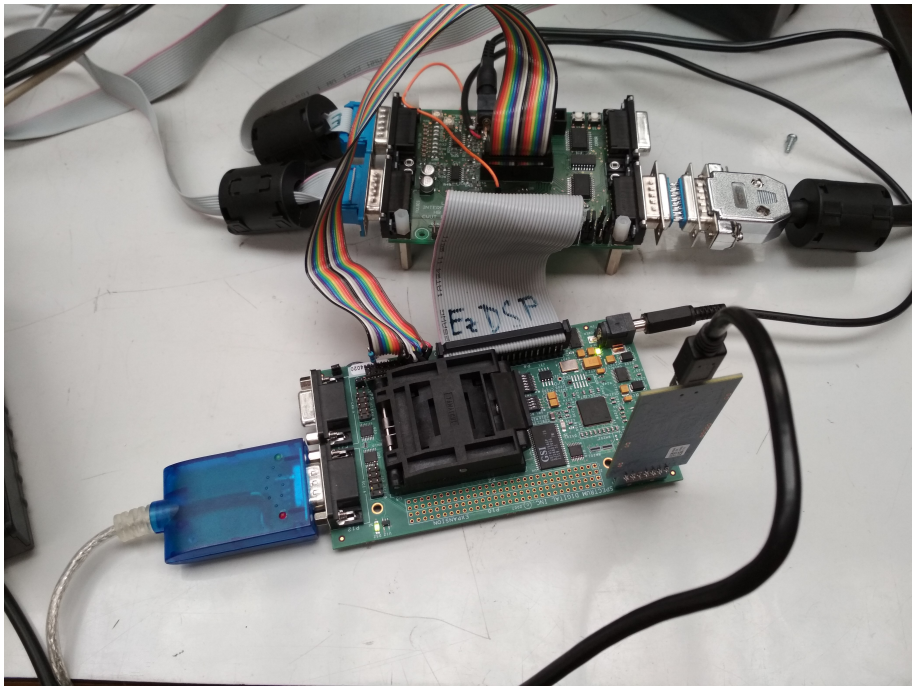
Pro měření fázových proudů a napětí ve stejnosměrném meziobvodu napěťového měniče je použito laboratorní izolované měřicí rozhraní Bummer s čidly LEM, jehož fotografie je na obr. B.4. Pro měření proudu je použit typ LEM LF 205-S, pro měření napětí pak LEM LV 25-P. Maximální vstupní napětí je 800 V, maximální vstupní proud pak 200 A na jeden závit. Na každém proudovém čidle jsou provedeny 4 závity, tudíž je maximální proud snížen na 50 A.

Proudový výstup napěťových a proudových čidel je pomocí rezistorů převeden na napětí, které je na propojovací desce (obr. B.5) dále upraveno na napěťovou úroveň F28335 a podloženo stejnosměrnou složkou pro možnost měření obou polarit. Takto upravené napětí je následně přivedeno na vstup AD převodníku.

Obrázek B.4. Měřicí rozhraní

B.3.2 Čidlo otáček

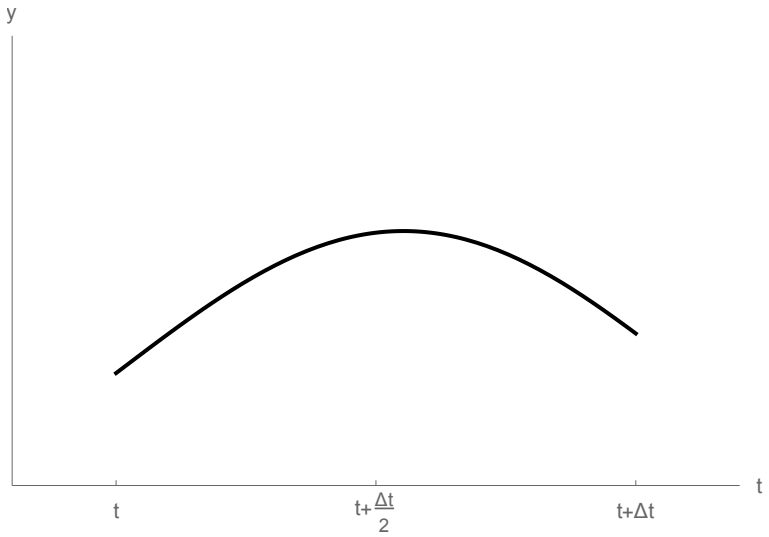
Pro měření otáček je použit inkrementální rotační snímač IRC305/1024 KB od firmy LARM, který disponuje 1024 pulzy na otáčku. Výstup z čidla je napěťově upraven na propojovací desce a následně přiveden na příslušné vstupy eQEP jednotky.



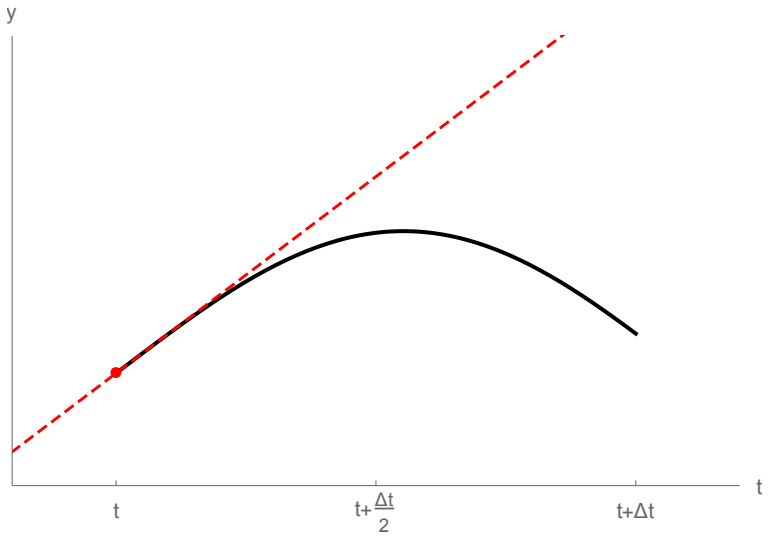
Obrázek B.5. Propojovací deska společně s připojeným procesorem

Příloha C
Grafická demonstrace metody Runge-Kutta 4. řádu

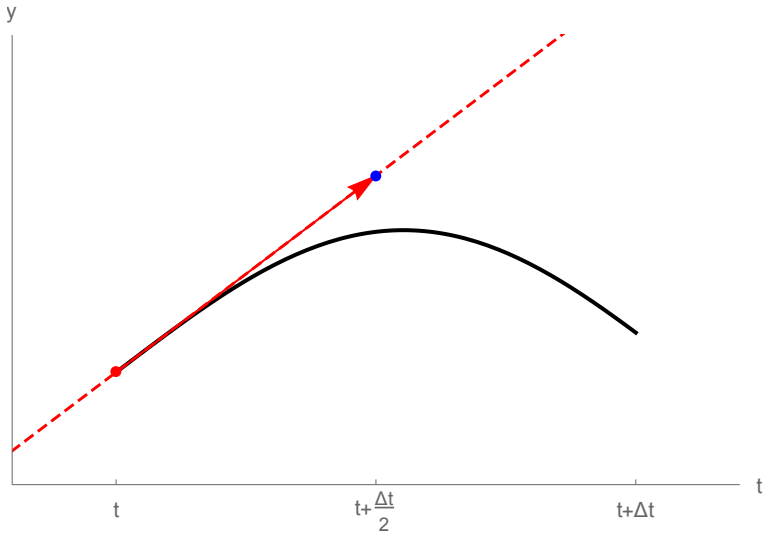
Předpokládejme, že máme k dispozici předpis pro derivaci hledané funkce a že známe průběh přesného řešení:



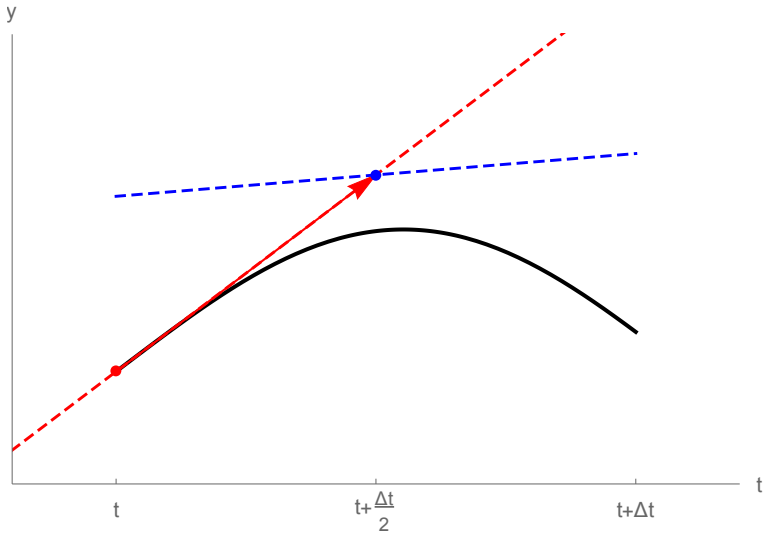
Nejdříve spočítáme derivaci k_1 v počátku. Ta má směr podle červené čárkované úsečky:



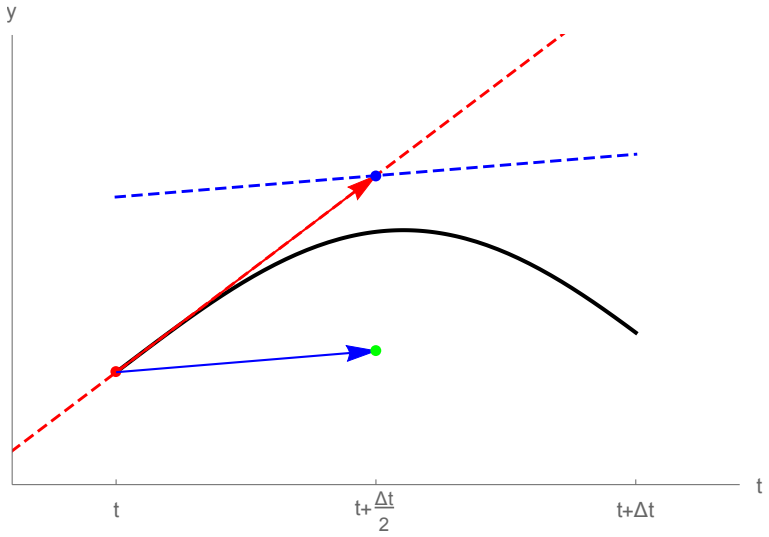
Nyní popojdeme z bodu t do bodu $t + \frac{\Delta t}{2}$ ve směru k_1 (modrý bod):



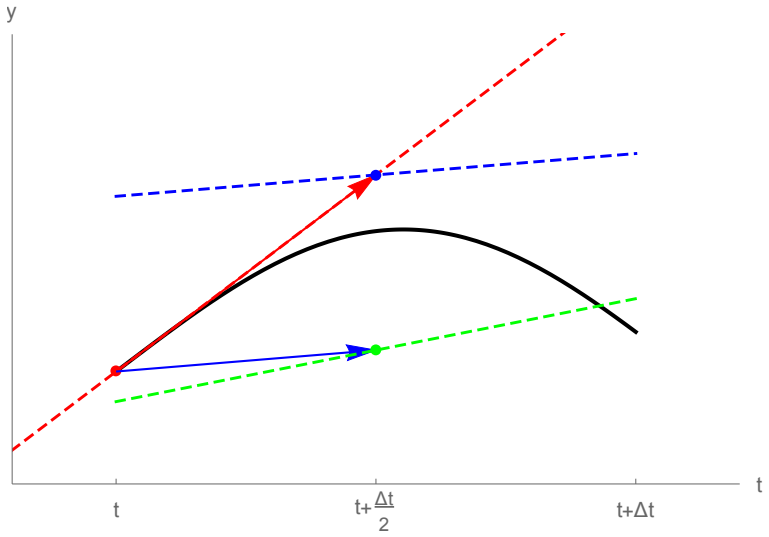
V modrém bodě spočítáme novou derivaci k_2, která má směr podle modré čárkované úsečky:



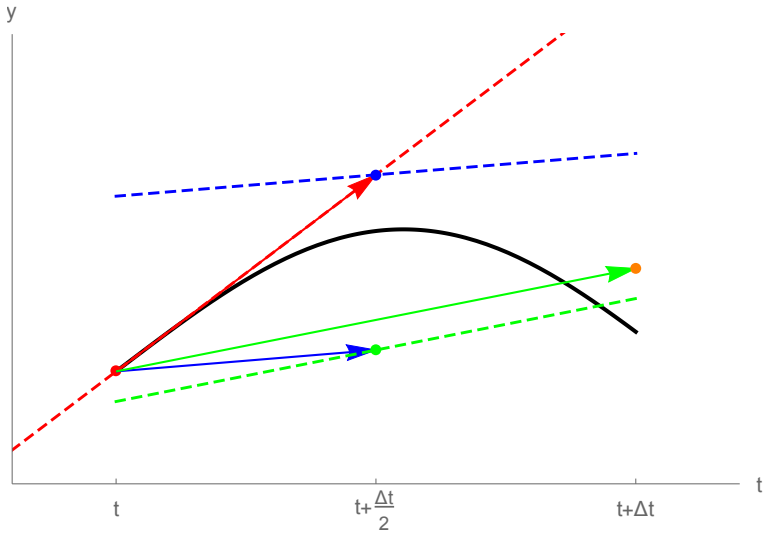
Znova vyrazíme z bodu t do bodu $t+\frac{\Delta t}{2}$, ale nyní ve směru derivace k_2 (zelený bod):



V zeleném bodě spočítáme novou derivaci k_3, která má směr podle zelené čárkované úsečky:



Znova vyrazíme z bodu t, ale tentokrát o celý krok do bodu $t + \Delta t$, a to ve směru derivace k_3 (oranžový bod):



V tomto bodě spočítáme derivaci k_4, která má směr oranžové čárkované úsečky. Výslednou derivaci k spočítáme jako $1/6(k_1 + 2k_2 + 2k_3 + k_4)$. Aproximaci hledané funkce získáme tak, že z bodu t vyrazíme do bodu $t + \Delta t$ ve směru derivace k (černý bod):