VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

OPTIMALIZACE STAVOVÉHO REGULÁTORU PRO ŘÍZENÍ DC MOTORU NA FPGA
OPTIMIZATION OF THE DC MOTOR STATE SPACE CONTROLLER FOR FPGA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Michal Maliszewski**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ondřej Andrš, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Michal Maliszewski**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Ondřej Andrš, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimalizace stavového regulátoru pro řízení DC motoru na FPGA

Stručná charakteristika problematiky úkolu:

Cílem práce je navrhnout, simulačně ověřit a optimalizovat algoritmus výpočtu stavového regulátoru pro implementaci na FPGA poli pro řízení DC motorů. Implementace bude provedena na platformě NI CompactRIO ve vývojovém prostředí LabVIEW a Simulink/MATLAB.

Cíle diplomové práce:

1. Seznámit se s vývojovým prostředím LabVIEW a platformou cRIO.
2. Provést rešeršní studii daného tématu.
3. Navrhnout a simulačně ověřit stavový regulátor pro DC motor.
4. Implementovat stavový regulátor na platformě cRIO.
5. Implementovaný regulátor otestovat a vyzkoušet.

Seznam doporučené literatury:

VLACH Jaroslav, HAVLÍČEK Josef a VLACH Martin: Začínáme s LabVIEW. 1. vyd. Praha: BEN - technická literatura, 2008. ISBN 9788073002459.

Firemní dokumentace National Instruments

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato práce se zabývá optimalizací stavové regulace DC motoru na FPGA s využitím programu LabVIEW a platformy NI cRIO. V první části je v prostředí Matlab/Simulink odvozen stavový model daného DC motoru, pro který je proveden návrh zpětnovazební regulace polohy s integrátorem na vstupu a stavovým pozorovatelem s kompenzací poruchy metodou LQR. Práce pokračuje převedením regulátoru do prostředí LabVIEW, kde je kód upraven pro použití na FPGA. Dále je aplikace optimalizována s důrazem na využití hardwarových prostředků FPGA, kdy je nezbytná zejména práce s datovým typem fixed-point. Po úspěšné kompilaci a spuštění na cílovém hardwaru je připojen reálný motor a je provedena série testů. Výstupem práce je funkční stavový regulátor na FPGA a uživatelské rozhraní na real-time kontroléru cRIO, které uživateli umožňuje daný DC motor řídit a ukládat důležitá data na disk.

Abstract

This thesis deals with the optimization of state space controller of DC motor on FPGA in LabVIEW environment on NI cRIO platform. In the first part, the state space model of the given DC motor is presented in Matlab/Simulink and then the position feedback controller with steady-state error elimination and with state observer with error compensation using LQR method is designed. The thesis continues with transforming the controller to LabVIEW environment where the code is edited for FPGA use. Next, the focus on FPGA hardware resources consumption optimization leads to careful work with fixed-point data type. After successful code compilation on target hardware, the real given DC motor is connected and the series of tests are performed. The output of the thesis is working state space controller running on FPGA and the graphical user interface on real-time host cRIO, which enables the user to control the plant and save the data on the disk.

Klíčová slova

Stavový regulátor, stavový pozorovatel, LQR, LabVIEW FPGA, stejnosměrný motor.

Key words

State space controller, state observer, LQR, LabVIEW FPGA, DC motor.

Bibliografická citace

MALISZEWSKI, M. Optimalizace stavového regulátoru pro řízení DC motoru na FPGA. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 79 s. Vedoucí diplomové práce Ing. Ondřej Andrš, Ph.D.

Čestné prohlášení

Čestně prohlašuji, že jsem tuto práci vypracoval samostatně s použitím uvedené literatury a pod vedením vedoucího DP.

……...

V Brně, 2017 Bc. Michal Maliszewski

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu DP panu Ing. Ondřeji Andršovi, Ph.D za cenné rady a za vytvoření příjemného pracovního prostředí. Dále bych chtěl ocenit podporu, která se mi dostala od mých rodičů a přítelkyně.

Obsah

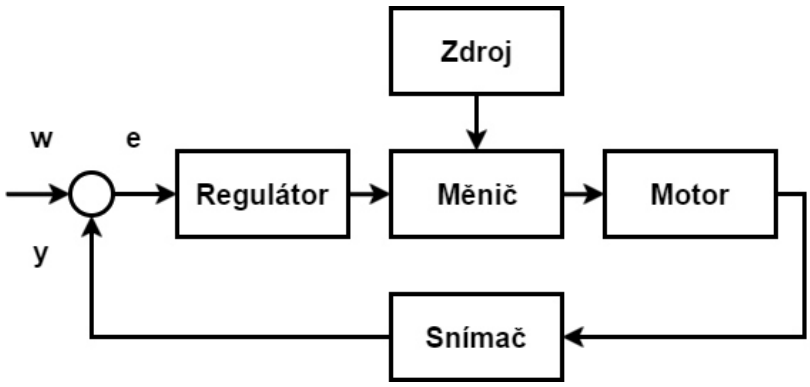
1. Úvod 15
1.1. Stanovení cílů 16
2. Teoretický rozbor 17
2.1. Shrnutí článků souvisejících s tématem práce 17
2.2. Vývoj na základě modelu 18
2.3. Řízení a simulace v reálném čase 21
2.4. FPGA 21
2.4.1. Části FPGA 22
2.4.2. Nástroje pro programování FPGA 24
2.5. Stejnosměrný motor 25
2.5.1. Popis 25
2.5.2. Rovnice pro popis přechodového děje 27
2.5.3. Odvození přenosu 28
2.6. Stavový popis a základy zpětnovazební regulace 29
2.6.1. Modelování ve stavovém prostoru 29
2.6.2. Řiditelnost a pozorovatelnost soustavy 30
2.6.3. Metoda pole placement 31
2.6.4. Integrátor na vstupu 32
2.6.5. Pozorovatel 33
2.6.6. Pozorovatel poruchy 34
2.6.7. LQR 35
2.7. Převod do diskrétní podoby 36
2.7.1. Vzorkování a kvantování signálu 37
2.7.2. Převod spojitého stavového modelu na diskrétní 38
2.7.3. Aritmetika s pevnou desetinnou čárkou 41
3. Vytvoření řídicího systému 43
3.1. Návrh a vytvoření regulátoru v prostředí Matlab/Simulink 43
3.1.1. Vytvoření modelu soustavy 43
3.1.2. Návrh zpětnovazebního stavového regulátoru s integrací na vstupu 44
3.1.3. Vytvoření pozorovatele s korekcí poruchy 46
3.1.4. Převedení do diskrétní podoby a testování v uzavřené smyčce 49
3.2. Implementace regulátoru pro platformu cRIO 51
3.2.1. Převedení modelů do prostředí LabVIEW 51
3.2.2. PWM modulace a zpracování dat z enkodéru 55

3.2.3. Implementace kódu na FPGA 56
3.2.4. Vytvoření host aplikace 59
4. Experimentování s reálným motorem 61
4.1. Popis použitého hardwaru 61
4.2. Konfigurace systému 63
4.3. Testování řídicího systému 64
4.4. Vyhodnocení výsledků měření 66
5. Závěr 69
6. Zdroje 71
7. Seznam použitých symbolů a veličin 75
8. Seznam obrázků 77
9. Seznam přiložených souborů 79

1. Úvod

Díky svému širokému využití tvoří v současnosti elektrické pohony významnou část produkce v průmyslu, a proto zdokonalování metod simulace při návrhu těchto celků vede k lepší efektivitě výroby a úspěchu na trhu. Na vývoj jsou dnes kladeny vysoké nároky, a to zejména minimální náklady, dosažení rychlých a spolehlivých simulačních výsledků, které vedou k optimalizaci parametrů produktu a k minimalizaci (ideálně eliminaci) rizik při testování prototypů. Splnění těchto kritérií je dosaženo při simulacích v reálném čase (real-time aplikace), které nahrazují experimentální měření na prototypech daného výrobku, čímž dochází k významné úspoře doby a nákladů na vývoj.

Numerické simulace elektrických strojů s malou časovou konstantou mohou vyžadovat krátký časový krok (menší než mikrosekunda). V takových případech se jako výhodný tah ukazuje přesunutí výpočtů z real-time procesoru na FPGA (programovatelná hradlová pole, anglicky Field Programmable Gate Arrays), které zajišťuje dostatečnou frekvenci výpočtů i pro velmi rychlé systémy.



Obrázek 1.1 - Schéma pohonu

Tato práce si dává za cíl implementovat metody stavového zpětnovazebního řízení v regulátoru na FPGA. Aby byla zajištěna správná a bezpečná funkce motoru, musí být měnič vhodně řízen. Hlavním úkolem regulátoru je tedy generovat signály pro polovodičové spínače (většinou tranzistory) v měniči, čímž je docíleno požadované úrovně napětí na vstupu do motoru. Mezi další úkoly řídicího obvodu patří obsluha brzdy motoru, monitorování provozních podmínek a ochrana výkonových tranzistorů měniče, kdy při detekci poruchy regulátor vypne napájení, aby bylo zabráněno zničení zařízení.

Během posledních desetiletí bylo rozvinuto mnoho metod návrhu řidicího členu. Z hlediska hardwaru lze regulátory rozdělit na tyto tři skupiny:

- Analogové obvody – spojité řízení je dnes na ústupu před diskrétním
- Mikroprocesory – zejména DSP (digitální signálové procesory).
- FPGA čipy

Analogové obvody jsou dnes nahrazovány diskrétním řízením zejména z těchto důvodů: offset a drift zesilovačů, nižší dlouhodobá přesnost (stárnutí součástek), algoritmus řízení je dán hardwarem (obtížná modifikace funkce oproti diskrétnímu řízení, kde je algoritmus dán snadno upravitelným softwarem), adaptivní řízení vyžaduje velmi složité zapojení obvodů, u složitěj-

ších soustav je regulátor obvykle dražší. Na druhou stranu dynamické vlastnosti pohonu nezávisí na diskretizaci (velikosti kroku výpočtu), a proto analogové obvody mají dobrou odezvu u lineárních soustav. Stále se proto používají u řízení nízkonapěťových stejnosměrných měničů zejména při velkých spínacích frekvencích.

Mikrokontroléry, DSP a FPGA jsou v současnosti jasnou hardwarovou volbou pro vytvoření moderního regulátoru, který řídí výkonový měnič. S neustále se zvyšujícím výpočetním výkonem je dnes umožněno jednomu mikroprocesoru řídit celý pohon, který může zahrnovat několik měničů. Navíc ceny mikroprocesorů se neustále snižují, což umožňuje jejich uplatnění u nízkonákladových aplikací. Naproti tomu hlavními argumenty pro použití FPGA jsou zejména velmi vysoká rychlost odezvy a z principu paralelní charakter výpočtů (srovnání mikroprocesoru a FPGA je detailněji rozebráno v kapitole 2.4). Moderní pohony jsou dnes komplexními inženýrskými systémy, které v sobě kombinují vysokou účinnost energetické přeměny s pokročilým zpracováváním dat. Kvůli složitosti návrhu jsou dnes komerčně úspěšné pohony obvykle výsledkem týmové práce inženýrů s různým odborným zaměřením.

1.1. Stanovení cílů

Dle názvu diplomové práce je hlavním cílem aplikovat vhodný typ stavového regulátoru na FPGA a otestovat řídicí člen na reálném stejnosměrném motoru. Pro lepší orientaci byly vytvořeny následující dílčí cíle, které přináší náhled do postupu práce.

- vytvoření stavového modelu motoru
- návrh regulátoru
- simulační ověření v prostředí Matlab/Simulink
- převod do prostředí LabVIEW
- off-line testování v prostředí LabVIEW
- vytvoření regulátoru na platformě cRIO
- testování na reálném motoru

Prvním krokem je vytvoření stavového modelu motoru, kdy dosazením parametrů použitého reálného stejnosměrného motoru jej můžeme chápat jako jeho náhradu, na které budou prováděny experimenty s různými typy řídicích algoritmů. Poté bude provedena optimalizace nastavení parametrů regulátoru pro vybraný řídicí algoritmus (tzv. MIL simulace, viz kapitola 2.2). Výstupem je tedy regulátor naladěný na virtuální model reálného DC motoru.

Dalším krokem bude převod virtuálního modelu stejnosměrného motoru a připravovaného regulátoru z prostředí Matlab/Simulink do prostředí LabVIEW. Model regulátoru musí být převeden takovým způsobem, aby jeho podoba v LabVIEW umožňovala kompilaci softwaru do FPGA na cRIO. Musí být zaručena správnost výpočtu a zároveň nesmí být program příliš rozsáhlý, aby nepřekročil kapacitu FPGA. Správnost výpočtu tedy bude testována opět simulačně formou MIL, tentokrát ale v prostředí LabVIEW a to s důrazem na minimalizaci náročných matematických operací a na velikost použitých datových typů.

Po úspěšné kompilaci přijde poslední krok, kterým bude připojení regulátoru na FPGA k reálné soustavě se stejnosměrným motorem, doladění parametrů regulátoru a porovnání výsledků se simulacemi.

2. Teoretický rozbor

2.1. Shrnutí článků souvisejících s tématem práce

Problematikou teorie řízení se zabývá kniha Modern Control Engineering [1], která se zaměřuje zejména na analýzu a návrh řídicích systémů. Autor Katsuhiko Ogata knihu koncipoval jako učebnici, tudíž jsou předkládané myšlenky napsané srozumitelně a často jsou demonstrovány na řešených příkladech. Autor u čtenáře předpokládá základní znalosti z oblastí diferenciálních rovnic, Laplaceovy transformace, vektorového a maticového počtu, analýzy elektrických obvodů, mechaniky a základů termodynamiky. K řešení příkladů je využíváno prostředí Matlab. Z hlediska tématu práce jsou zajímavé především kapitoly 2, 9 a 10 zabývající se modelováním ve stavovém prostoru, analýzou stavových modelů a návrhem stavových regulátorů.

Problémem polohové regulace stejnosměrného motoru se zabývá práce [2], která si klade za cíl srovnat různé řídicí algoritmy. Simulace jsou provedeny v prostředí Matlab/Simulink a praktické experimenty realizovány s pomocí softwarových a hardwarových nástrojů firmy National Instruments. PID, kaskádový a stavový zpětnovazební regulátor byly vytvořeny pro daný DC motor s převodovkou, u kterého byla snímána poloha rotoru pomocí kvadraturního enkodéru. Parametry PID regulátoru byly získány metodou Ziegler-Nichols, kaskádové regulace experimentálně na modelu motoru v Matlabu a parametry stavového regulátoru pomocí metody pole placement.

Srovnání typů regulace	PID regulátor	Kaskádový regulátor	Stavový regulátor
Náběh [ms]	425	1120	390
Překmit [%]	0	6.35	0.5
Ustálení [ms]	800	3500	550
Ustálená odchylka [%]	0	0.1	0.04

Tabulka 1 – Srovnání různých typů regulátoru pro daný DC motor

U stavové regulace byl použit zpětnovazební regulátor bez integrace na vstupu a bez pozorovatele (viz kapitola 2.6.3), statická odchylka byla kompenzována násobením regulační odchylky kompenzační konstantou. Nejpřesnějších výsledků bylo dosaženo pomocí PID regulátoru a nejrychlejší odezvu vykazoval stavový regulátor.

Článek [3] se zabývá simulacemi stejnosměrných i střídavých motorů na základě LabVIEW FPGA. Náplní práce je vytvořit stavový model motoru a implementovat jej na FPGA, kdy při malém časovém kroku dosahuje model motoru odezvy blízké tomu reálnému. Z prostředí LabVIEW real-time je poté model řízen a regulován. Jako aproximace integrace byl na FPGA naprogramován algoritmus Runge-Kutta 4. řádu. Tato práce je dobrým základem pro testování regulátorů metodou HIL, kdy stačí pozměnit parametry modelu podle daného motoru a navrhnutý regulátor k modelu připojit. Implementace matic na FPGA není standartním způsobem možná, protože LabVIEW FPGA umožňuje práci pouze s 1-D poli. Proto v této práci byla použita kombinace clusterů a 1-D polí. Aby mohly být matice zrekonstruovány, je vyžadována dodatečná funkce pro vyčítání hodnot z těchto struktur.

V práci [4] se autoři zabývají návrhem algoritmu řízení tak, aby bylo dosaženo co nejlepší odezvy na skok u servopohonu. Vychází přitom ze stavového popisu stejnosměrného motoru. Výsledná „dynamická synergická metoda“ v sobě zahrnuje řízení polohy, rychlosti i proudu. Simulačně je poté testována na parametrech daného motoru a dosažená odezva na skok má následující vlastnosti. Při časovém kroku 10 μs a skoku polohy o 90° dosahuje překmitu 0,0167% a

ustálené odchylky 0,00025° v čase 6 sekund. Snižováním časového kroku simulace se kvalita regulace zvyšuje.

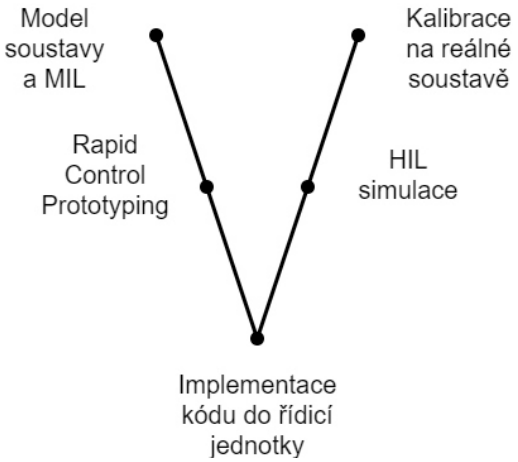
V práci [5] je s využitím LabVIEW a platformy cRIO analyzována a implementována technika návrhu regulace otáček pro čtyři stejnosměrné motory, které řídí kola vozítka Mars rover. Vozítko nemá žádné mechanické řízení, proto je jeho pohyb řízen dálkově čistě otáčkami poháněných kol, která jsou na sobě nezávislá. Rozhraní mezi motory a dálkovým ovládáním je implementováno na FPGA a regulátor v real-time systému. PID regulátor byl zvolen jako řídicí algoritmus a zpětná vazba je zajištěna měřením polohy pomocí kvadraturních enkodérů. Regulátor byl naladěn pomocí metody Ziegler-Nichols.

Článek [6] se zabývá problémem přesné polohové regulace při využití LVDT (linear variable differential transformer) jako senzoru lineárního posuvu v prostředí LabVIEW FPGA. Výhodou tohoto senzoru je chod bez tření, ovšem za cenu nelinearity při vyšších změnách požadované polohy. Tato nelinearita je kompenzována pomocí experimentálně získané korekční funkce v host VI. Jako řídicí algoritmus byl použit PID a tzv. ramp control regulátor (řízené napětí do motoru se proporcionálně snižuje s tím, jak se aktuální poloha blíží té žádané). Při skoku o 10 mm dosahovala chyba regulace 30 – 50 µm u obou typů regulátorů, u kroku 100 µm chyba nepřekročila 7 µm a obecně platilo, že u ramp control regulátoru byla zhruba poloviční oproti PID. Po linearizaci se chyba zmenšila na maximálně 4 µm pro oba typy regulace při 100 µm kroku.

2.2. Vývoj na základě modelu

Vývoj na základě modelu (neboli Model Based Design - MBD) je podle [7] matematická metoda vhodná pro řešení komplexních problému z oblasti řízení, zpracování signálu a komunikačních systémů. Tato metoda se obecně uplatňuje při vývoji embedded („zabudovaného“) zařízení.

MBD poskytuje efektivní přístup k vytvoření schématu vývoje produktu a tím usnadňuje vývojový cyklus, který je pak určen tzv. V – diagramem.



Obrázek 2.1 - V – diagram

V-diagram tedy popisuje jednotlivé fáze návrhu kontroléru.

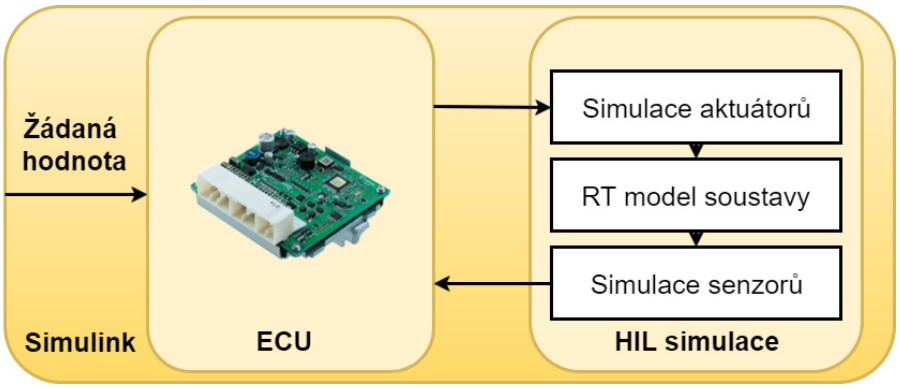
K modelování soustavy se dá s ohledem na znalosti daného systému přistupovat dvěma způsoby. U jednodušších soustav je v Simulinku obvykle prvním krokem sestavení blokového

schématu představujícího soustavu diferenciálních rovnic, které charakterizují dynamické chování soustavy nebo využití tzv. fyzikálního modelování, kde jednotlivé bloky schématu představují fyzické prvky soustavy a na pozadí je řešena soustava diferenciálních rovnic (knihovna Simscape). U složitých soustav, kde je nemožné nebo nevýhodné sestavit rovnice je model získáván z dat naměřených na reálné soustavě. Modelování dynamických systémů tedy můžeme podle [8] rozdělit následovně:

- White box – rovnice jsou známy a parametry jsou dopočítány, změřeny, získány z externích zdrojů (např. technická dokumentace produktu) nebo expertně odhadnuty.
- Grey box - rovnice jsou známy a parametry jsou získány experimentálně ze vstupních a výstupních dat.
- Black box – rovnice nejsou známy, hledá se proto závislost mezi vstupními a výstupními daty získanými experimentálně a model je de facto funkce závislosti výstupu na vstupu.

Pro odhad parametrů u grey boxu lze s výhodou použít nástroje Simulink Parameter Estimation, kdy vstupem je model v Simulinku a naměřená vstupní a výstupní data a výstupem je hledaný parametr. Klíčová je volba počátečního odhadu hledaného parametru a optimalizační metody (nelineární nejmenší čtverce, simplex search, pattern search, gradient descent apod.). U black boxu je možné využít nástroj System Identification Toolbox, kde vstupem jsou naměřená vstupní a výstupní data a výstupem je funkce závislosti výstupu na vstupu. Zde je opět klíčová volba aproximačního modelu, často jsou tyto modelu ve tvaru polynomu (např. ARX).

Dalším krokem je MIL (Model in the Loop) návrh řídicího algoritmu, který vychází z dynamických vlastností vytvořeného virtuálního modelu soustavy. Regulátor je pak na něm testován a jeho parametry jsou v simulaci iteračně laděny tak, aby došlo k požadovanému chování soustavy. Jedná se o idealizovaný návrh, kdy neuvažujeme vliv propojení řídicí jednotky (ECU – elektronická řídicí jednotka) a reálné soustavy (zejména rychlost a komunikace), vliv překladače, odchylku chování modelu od reálné soustavy, výpočtový výkon ECU atd.



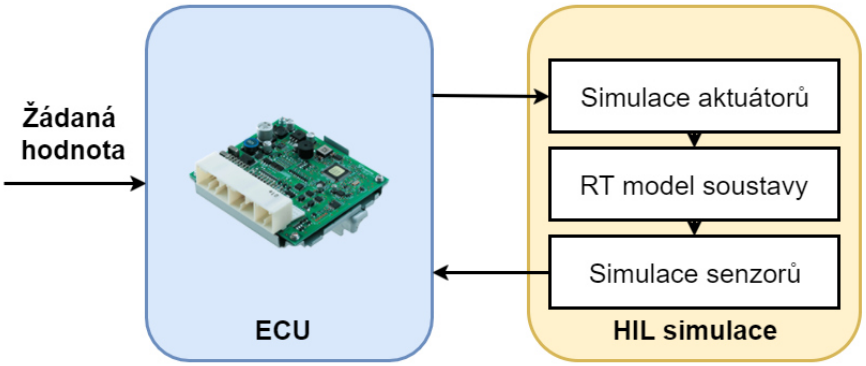
Obrázek 2.2 - Schéma MIL simulace

Rapid Control Prototyping (RCP) představuje podle **Chyba! Nenalezen zdroj odkazů.** testování připraveného řídicího algoritmu na reálné soustavě, kdy řídicí algoritmus běží na PC připojeném na reálnou soustavu. Simulace regulátoru probíhá v režimu Hard Real Time (viz kapitola 2.3), což klade vysoké nároky na hardware, na kterém je simulován kontrolér. Klasický počítač díky tomu není možné bez úprav použít ať už z důvodu operačního systému (nejčastější Windows není deterministický) nebo periferií nutných pro propojení s okolními zařízeními. Po-

čítač lze ovšem rozšířit o speciální vstupně/výstupní (I/O) karty, které umožňují deterministickou komunikaci procesoru s okolím. Poté po kompilaci programu a spuštění na procesoru lze do určité míry dosáhnout deterministické odezvy. Další podstatně nákladnější možností je využít např. produkty značky dSPACE nebo National Instruments, které poskytují kompletní hardwarové a softwarové řešení simulací v reálném čase.

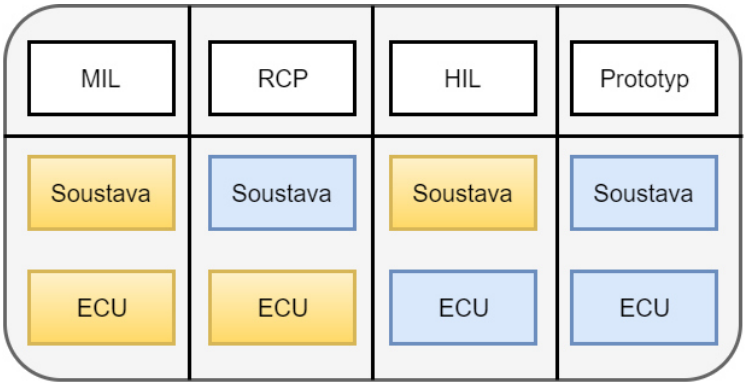
Implementace kódu do řídicí jednotky neboli „target code generation“ spočívá v kompilaci a nahrání softwaru regulátoru do konečného hardwaru (obvykle mikrokontrolér). Tento proces probíhá automaticky s využitím nástrojů pro generování kódu. V tomto kroku tedy vzniká elektronická řídicí jednotka.

Oproti MIL simulacím HIL (Hardware in the Loop) simulace představuje testování řídicí jednotky na modelu soustavy, který je simulován na real-time hardwaru. ECU tedy „vnímá“ model jako reálnou soustavou. Zde je proveden kompletní test řídicí jednotky jako prototypu, testuje se řídicí algoritmus a uvažuje se vliv připojení ECU na soustavu a výpočtový výkon ECU. Velkou výhodou HIL simulací je možnost otestovat chování řídicí jednotky pro nebezpečné soustavy (např. velké točivé stroje), kdy by selhání regulace při testování na reálné soustavě mohlo způsobit závažnou havárii. Další uplatnění HIL simulace nachází jako náhrada obtížně reprodukovatelných experimentů (např. posouzení kvality jízdních asistentů u osobních automobilů za nepříznivých podmínek).



Obrázek 2.3 - Schéma HIL simulace

Jelikož byl regulátor odladěn na modelu soustavy, který aproximuje reálný systém pouze s určitou přesností je nezbytná kalibrace na vlastní soustavě, kdy se provádí upravení parametrů regulátoru na základě měření na reálné soustavě.



Obrázek 2.4 - Srovnání metod návrhu a testování řídicí jednotky

Na obrázku Obrázek 2.4 - Srovnání metod návrhu a testování řídicí jednotky jsou oranžovou barvou vyznačeny simulované částí a modrou barvou reálná zařízení.

Mezi výhody vývoje na základě modelu patří:

- Schopnost identifikovat chyby v raném stádiu vývoje a minimalizovat tak finanční náklady a čas vynaložený na jejich korekci.
- Grafická prostředí, která jsou oproti tradičnímu textovému programování pro inženýry uživatelsky přívětivější (méně náchylné na chyby a rychlejší vývoj), vedou k větší přehlednosti a k zjednodušení modelů jejich hierarchizací. Jednotlivé funkční bloky pak mohou být rozděleny mezi pracovní skupiny.
- Snadná editace a rozšíření stávajících modelů.

2.3. Řízení a simulace v reálném čase

Základním prvkem simulačních nástrojů jsou řešiče tzv. ODE (obyčejná diferenciální rovnice), jež jsou postaveny na numerických integračních metodách (např. Runge-Kutta apod.), a které lze rozdělit podle délky kroku na tyto dvě hlavní skupiny:

- Řešiče s variabilním (proměnným) krokem výpočtu.
- Řešiče s konstantním krokem výpočtu.

Hlavní výhodou řešičů s variabilním krokem výpočtu je možnost měnit délku kroku v závislosti na změně přírůstku integrované funkce. Pokud jsou přírůstky velké, řešič zjemňuje krok a zpomaluje výpočet, naopak pokud nedochází k výrazným změnám, algoritmus prodlužuje krok a tím zkracuje dobu simulace. Z principu tedy tyto řešiče nejsou vhodné pro simulace v reálném čase, ale jejich úkolem je co nejvíce urychlovat simulace s dlouhým simulačním časem.

Naopak řešiče s konstantním krokem výpočtu jsou vhodné pro real-time simulace, kdy časové nároky na dobu výpočtu definuje tzv. dead-time. Komunikace a správný výpočet musí proběhnout v čase kratším, než stanovuje dead-time. Takové požadavky vedou na použití zařízení s deterministickým operačním systémem a periferiemi pro komunikaci s reálnými soustavami. Deterministický operační systém tedy provádí operace v předem stanovených časových intervalech, jež definují délku kroku. Tyto operační systémy můžeme rozdělit na dvě skupiny:

- Hard real-time – ryzí deterministický systém, kdy nedodržení dead-time je považováno za havárii.
- Soft real-time – deterministický systém, kdy občasné nedodržení dead-time znamená ztrátu kvality systému.

Hard real-time systémy se proto uplatňují u důležitých funkcí, kdy nedodržení dead-time může ohrozit kvalitu regulace a způsobit velké škody (např. řízení stability moderního stíhacího letounu apod.). FPGA se svou povahou řadí mezi hard real-time systémy.

2.4. FPGA

Programovatelné hradlové pole, ve zkratce FPGA, je křemíkový čip s piny, hradly a dalšími hardwarovými prostředky, které nejsou v základním stavu propojeny. Naprogramováním čipu se rozumí vytvoření takových fyzických spojení mezi hradly a bloky hardwaru, aby byla naplněna požadovaná funkce.

FPGA je z hlediska programování nejobecnější typ čipu, který umožňuje programování na té nejnižší možné hardwarové úrovni. V počátečním stavu (bez nahraného softwaru) neposkytuje

žádnou funkčnost. Na FPGA může být implementována jakákoliv digitální funkce, čímž čip poskytuje neomezenou flexibilitu omezenou pouze schopnostmi a představivostí uživatele (na FPGA může být například naprogramován DSP nebo jakýkoliv jiný typ mikroprocesoru).

Každý čip je vyroben s určitým omezeným množstvím hardwarových prostředků, které je možné mezi sebou propojovat pomocí programovatelných spojů. Rekonfigurací spojů vznikají nové digitální obvody, které díky vstupně/výstupním (I/O) pinům komunikují s okolními zařízeními.

Oproti mikroprocesorům jsou FPGA čipy rychlejší, flexibilnější, umožňují snadnější dodatečné úpravy (nevyužité hardwarové prostředky mohou vykonávat přidanou funkci paralelně k hlavnímu programu), poskytují možnost paralelního chodu výpočtů a mají vysoký počet vstupně/výstupních pinů. Na druhou stranu se FPGA potýká s vyšší pořizovací cenou, vyšší spotřebou energie, volatilitou (obvykle absence flash paměti), složitostí čipů (technická dokumentace v řádu tisíců stran), složitými nástroji a málo intuitivním programování pomocí HDL (Hardware Desription Language) a s obtížným porovnáním výkonu jednotlivých zařízení z důvodu různé architektury čipů a terminologie u jednotlivých výrobců.

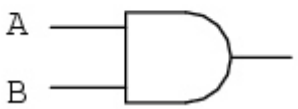
2.4.1. Části FPGA

Hardwarové prostředky FPGA od firmy Xilinx, které ve svých produktech používá firma National Instruments, se dají podle [9] rozdělit na tři skupiny: konfigurovatelné logické bloky, nekonfigurovatelné logické funkční bloky a paměťové bloky.

Konfigurovatelné logické bloky (CLB) jsou základní jednotka FPGA. V literatuře často nazývány slices nebo logic cells. Nejčastěji se jedná o klopné obvody (flip-flops) a vyhledávací tabulky (look-up tables, zkráceně LUT).

- Flip-flops – bistabilní klopné obvody, které slouží k synchronizaci a uložení binárního stavu do dalšího časového kroku. Na začátku periody obvod podle vstupu nastaví na výstup hodnotu pravda nebo nepravda a drží ji do konce iterace. Detailnější rozbor v [11].
- LUTs – mnoho logiky je v CLB implementováno pomocí malých pamětí RAM ve formě vyhledávacích tabulek. Elementární logické funkce (NAND, NOR, AND, atd.) nejsou na čipu implementovány hardwarově, ale jako pravdivostní tabulky uložené v LUT. V pravdivostní tabulce jsou předdefinovány výstupní hodnoty pro různé kombinace vstupů.

A	0	0	1	1
B	0	1	0	1
Výstup	0	0	0	1



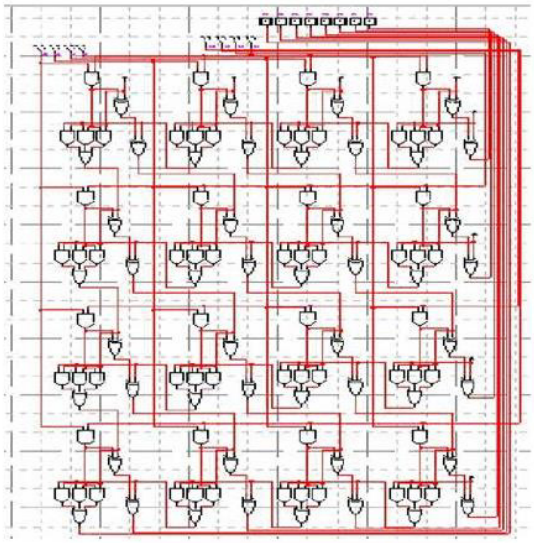
Obrázek 2.5 - Vyhledávací tabulka (LUT) pro funkci AND

Nekonfigurovatelné logické funkční bloky mají zejména za úkol optimalizovat výpočetně náročnou operaci násobení. Tuto elementární matematickou operaci je poměrně obtížné převést

do prostředí digitálních obvodů. Důvodem je velký počet operací a záběr značného množství hardwarových prostředků.

Pro násobení dvou 32-bitových čísel je podle [9]_potřeba více než 2000 operací, proto mají FPGA předbudované násobičky, které snižují využití konfigurovatelných logických bloků a celkově tak šetří místo na čipu.

- DSP48 slices – předdefinovaný obvod funkce multiply-accumulate (vynásob a přičti do akumulátoru), která je implementována do FPGA vyšších tříd (např. Xilinx Virtex-5).

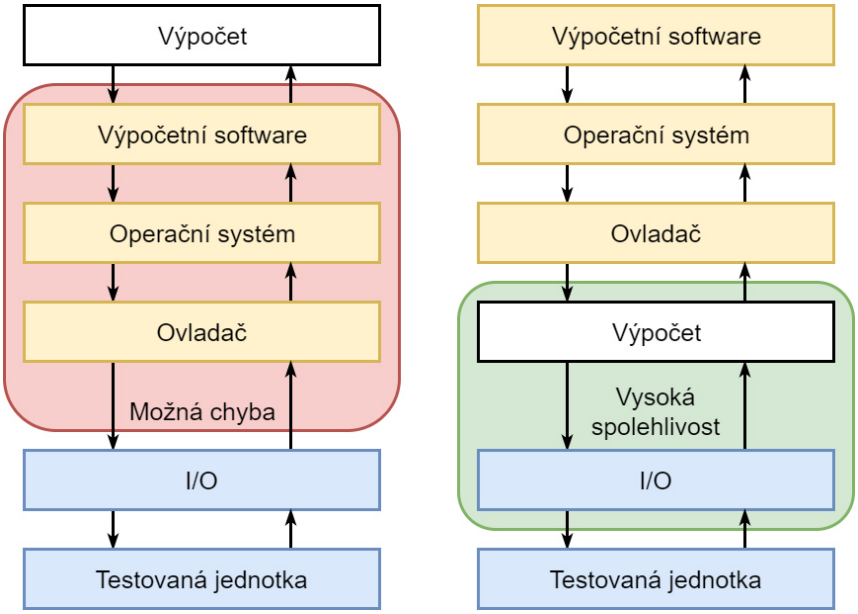


Obrázek 2.6 - Násobení dvou 4-bitových čísel pomocí kombinačního obvodu [9]

Paměťové bloky slouží k ukládání dat nebo jejich předávání mezi paralelními procesy mimo konfigurovatelné logické bloky do paměti RAM. Tím dochází ke značné úspoře hardwarových prostředků. V závislosti na konkrétním FPGA má uživatel k dispozici bloky od 16 do 36kb paměti. Data mohou být alternativně uchovávána v klopných obvodech, což ovšem způsobuje výrazný úbytek hardwarových prostředků (např. u čipu Virtex-II 1000 pole o sto 32-bitových číslech zabírá 30% kapacity klopných obvodů nebo 1% zabudované paměti RAM).

Implementace funkce na hardwarové úrovni s sebou přináší zlepšení rychlosti běhu programu, spolehlivosti a flexibility. Na druhou stranu, FPGA umožňuje přístup pouze ke vstupně/výstupním pinům. Neexistují zde žádné ovladače nebo operační systémy jako u mikroprocesoru. Mikroprocesory s operačními systémy umožňují snadnou práci se souborovým systémem a komunikaci s periferiemi, což je velmi důležité například pro funkce vyžadující zápis dat.

Proto je dnes často využívána tzv. hybridní (heterogenní) architektura, která spočívá v připojení mikroprocesoru na vstupně/výstupní porty skrz FPGA. Tímto mohou být využity výhody obou zařízení. Touto architekturou se zabývají např. firmy Xilinx (rodina Zynq) nebo National Instruments, které spojení mikroprocesoru s FPGA využívá ve svých rekonfigurovatelných vstupně/výstupních (RIO) zařízeních, mezi které patří například použité cRIO.



Obrázek 2.7 - Srovnání běhu výpočtu na mikroprocesoru a na FPGA

2.4.2. Nástroje pro programování FPGA

FPGA sestává řádově ze statisíců až miliónů hardwarových součástí, které je potřeba vhodně propojit pomocí programovatelných spojů tak, aby byla zajištěna zamýšlená funkce. V minulosti byly pro programování FPGA k dispozici pouze nízkoúrovňové nástroje, což kladlo vysoké nároky na znalosti programátora zejména z oblasti návrhu digitálních obvodů. V současnosti jsou na trhu vysokoúrovňové nástroje (HLS – high-level synthesis), které umožňují definovat funkci programu v uživatelsky přívětivém prostředí. Poté je vyvíjená aplikace použitým softwarem s pomocí kompilátoru přeložena do konfiguračního bitového souboru, který obsahuje informace o nastavení programovatelných spojů na FPGA tak, aby došlo ke správnému chodu programu.

Mezi nízkoúrovňové nástroje se podle [9] řadí HDL, jejichž hlavními zástupci jsou VHDL a Verilog. Tyto softwary spojují textové programování s vývojem digitálních obvodů. Programování v těchto jazycích je velice náročné a inženýrů a vědců s hlubokými znalostmi v tomto odvětví není mnoho, proto doposud nebyla FPGA technologie pro většinu badatelů dostupná, což se změnilo s příchodem HLS. Mezi obtíže s vývojem za pomocí HDL například patří:

- Paralelní povaha běhu výpočtů je skryta v sekvenci příkazů definovaných textovým programováním.
- Testování správného chodu FPGA obvykle zabere více času než vlastní vývoj aplikace.

Mezi HLS se řadí například použitý software LabVIEW od firmy National Instruments. LabVIEW patří mezi graficky programovatelná prostředí, kde je zřetelně zobrazen tok dat a paralelnost výpočtů, což programátorovi usnadňuje zorientovat se v chodu aplikace bez ohledu na hloubku znalostí FPGA technologií. Pro snadný přechod z HDL umožňuje LabVIEW importovat funkce napsané ve VHDL. Správný chod programu lze ověřit přímo ve vývojovém prostředí, což znatelně urychluje celkový vývoj produktu. Proces kompilace je zautomatizován a rozdělen na několik kroků, z nichž z každého je generován report pro snadnější odhalení chyb v programu. Mezi nejzajímavější informace, které lze z reportů vyčíst, patří procentuální využití jednotlivých hardwarových prostředků a očekávaná doba jednoho časového kroku. Pokud

tyto hodnoty překročí možnosti zařízení nebo programátorem stanovenou periodu výpočtu, kompilace se v dané fázi ukončí a upozorní uživatele na chybu.

S rozvojem HLS se rozšiřuje i míra použití FPGA technologie a to především v tzv. hybridní architektuře s připojeným mikroprocesorem. Stále však zůstává důležité porozumět dějům probíhajícím na FPGA a to zejména z těchto důvodů:

- Porovnání a volba vhodného zařízení pro danou aplikaci (množství LUT, násobiček apod.).
- Optimalizace rychlosti a využití hardwarových prostředků může umožnit přechod na jednodušší a tudíž levnější zařízení při současném zajištění požadované funkce.

2.5. Stejnosměrný motor

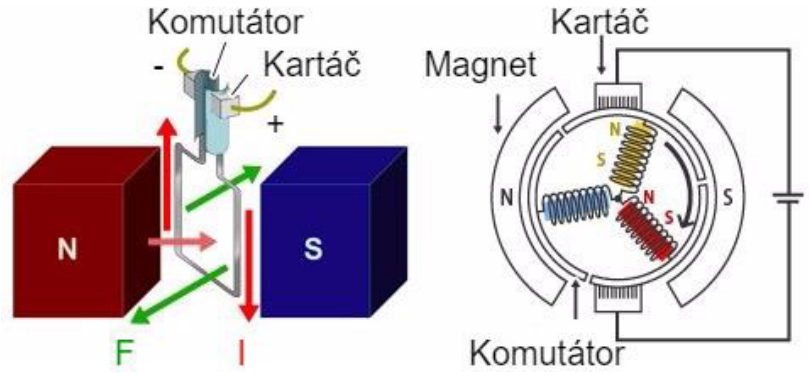
Následující kapitola je věnována popisu a odvození rovnic DC motoru, které budou dále využity pro sestavení jeho matematického modelu, který je nezbytný pro návrh pozorovatele a simulační ladění parametrů regulátoru.

2.5.1. Popis

Stejnosměrný motor neboli DC motor je jedním z historicky nejstarších elektrických strojů. Motor je napájen stejnosměrným napětím, čehož se s výhodou používá u regulace otáček, které jsou úměrné napájecímu napětí. DC motory lze podle typu zapojení rozdělit do následujících skupin:

- Cizí buzení
    - Budicí vinutí je uloženo ve statoru stroje a je napájeno ze samostatného zdroje napětí.
    - Magnetické pole je vytvářeno permanentními magnety.
- Sériové buzení – vinutí rotoru a statoru je zapojeno do série, tím pádem oběma vinutími protéká stejný proud.
- Paralelní buzení – vinutí rotoru a statoru je zapojeno paralelně, tím pádem se proudy statorem a rotorem liší.
- Kompaundní buzení – kombinace sériového a paralelního buzení, kdy je magnetizační statorové vinutí složeno minimálně ze dvou cívek, z nichž jedna je připojena k vinutí rotoru sériově a druhá paralelně.

V dalším textu se budeme zabývat výhradně DC motorem s cizím buzením, kde je magnetické pole buzeno permanentními magnety. Tyto stroje můžeme dále rozlišit podle typu komutace na kartáčové a bezkartáčové (BLDC motory). Důvodem je použití kartáčového DC motoru Maxon RE 35. Detailnější popis DC motorů je k dispozici v [12] a [13].



Obrázek 2.8 - Princip vytvoření momentu u kartáčového DC motoru

Vznik momentu u smyčky protékané proudem je dán Ampérovým zákonem, kdy na vodič v magnetickém poli při průchodu proudu působí síla podle vztahu:

$$d\boldsymbol{F} = \boldsymbol{I}d\boldsymbol{l}\mathrm{x}\boldsymbol{B} \qquad (1)$$

kde F … síla působící na vodič [N]

I … proud protékající vodičem [A]

l … aktivní délka vodiče [m]

B … magnetická indukce [T]

Nezbytnou součástí stroje je komutátor, který zajišťuje přepínání směru toku proudu rotorem tak, aby síla působila vždy stejným směrem a vznikal točivý moment. DC motor tohoto typu má konstantní magnetický tok, jehož velikost je dána typem magnetu a konstrukcí magnetického obvodu.

Stejnosměrný motor umožňuje provoz až ve 4 režimech (kvadrantech), kdy stroj může pracovat jako motor nebo jako generátor. Rozhodující je směr přeměny elektrické energie na mechanickou. Pokud je napájecí napětí větší než napětí indukované, teče proud směrem do motoru. Naopak pokud je indukované napětí větší než napětí zdroje, proud teče do zdroje. Toto platí pro oba směry otáčení (napětí) při použití vhodné výkonové elektroniky.

Pro motor v ustáleném stavu lze odvodit následující rovnice:

$$U = R_a I_a + c\phi\omega \qquad (2)$$

kde U … napájecí napětí [V]

R_a … odpor kotvy [Ω]

I_a … proud kotvy [A]

cϕ … konstanta motoru [V·s]

ω … úhlová rychlost [rad/s]

Z rovnice 2 plyne možnost řízení otáček pouze změnou napětí kotvy. A pro moment platí:

$$M = c\phi I_a \tag{3}$$

kde M … moment motoru [N·m]

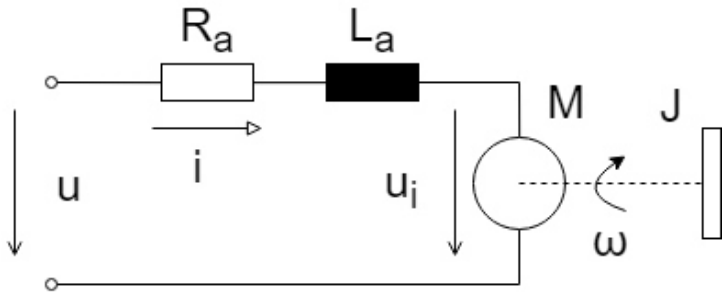
Z předchozích rovnic pak můžeme odvodit statickou zatěžovací charakteristiku:

$$\omega = \frac{U}{c\phi} - \frac{R_a}{(c\phi)^2} M = \omega_0 - kM \tag{4}$$

Při konstantním buzení mohou být otáčky řízeny změnou odporu kotvy R_a nebo změnou napájecího napětí U.

2.5.2. Rovnice pro popis přechodového děje

Pro popis dynamického chování soustavy je pro motor potřeba odvodit diferenciální rovnice, které vychází z obrázku Obrázek 2.9.



Obr. 2.9 - Schéma kartáčového DC motoru

Pro elektrickou část stroje platí rovnice:

$$u = R_a i_a + L_a i_a + c\phi\omega \tag{5}$$

Mechanická rovnice pak má tvar:

$$m_e = c\phi i_a = J\frac{d\omega}{dt} + b\omega + m_z \tag{6}$$

kde L_a … indukčnost kotvy [H]

J … moment setrvačnosti na hřídeli motoru [kg·m^2]

b … koeficient viskózního tření [N·m·s]

m_z … zátěžný moment [N·m]

Pomocí modelu sestaveného z mechanické a elektrické diferenciální rovnice můžeme simulovat dynamické chování DC motoru.

2.5.3. Odvození přenosu

Přenosem je v technice obvykle myšlen vztah (podíl) výstupní veličiny vůči vstupní veličině, převrácená hodnota je pak nazývána převodem. Laplaceovou transformací je možno výše uvedené rovnice převést do operátorového tvaru. Elektrická a mechanická (při zanedbání viskózního tření) rovnice má pak po Laplaceově transformaci tvar:

$$U(p) = R_a i_a(p) + pL_a i_a(p) + U_i(p) \tag{7}$$

$$M_e(p) = pJ\omega(p) + M_z \tag{8}$$

kde p … Laplaceův operátor

U_i … indukované napětí

Celkový přenos rozdělíme do 4 přenosových funkcí:

Výstupem bude proud kotvou I_a a vstupem rozdíl napájecího a indukovaného napětí.

$$F_1(p) = \frac{I_a}{U - U_i} = \frac{1}{R_a + pL_a} = \frac{\frac{1}{R_a}}{1 + p\tau_a} \tag{9}$$

$$\tau_a = \frac{L_a}{R_a} \tag{10}$$

kde τ_a … elektrická časová konstanta [s]

Výstupem je moment motoru, který je přímo úměrný proudu kotvou.

$$F_2(p) = \frac{M_e}{I_a} = c\phi \tag{11}$$

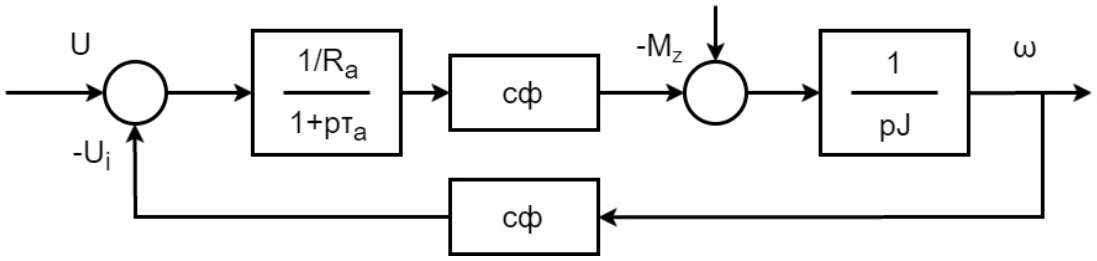
Přenos z momentu na úhlovou rychlost má integračního charakter.

$$F_3(p) = \frac{\omega}{M_e - M_z} = \frac{1}{pJ} \tag{12}$$

A nakonec indukované napětí je úměrné úhlové rychlosti.

$$F_4(p) = \frac{U_i}{\omega} = c\phi \tag{13}$$

Složením přenosových funkcí vzniká matematický model, jehož schéma je patrné z obrázku 4.



Obrázek 2.9 - Matematický model DC motoru v operátorovém tvaru

2.6. Stavový popis a základy zpětnovazební regulace

Alternativní možností vytvoření modelu soustavy je použití stavového prostoru pro popis dynamické soustavy. Tento postup je podrobně rozebrán v následující kapitole. Tato kapitola čerpá zejména ze zdrojů [14] a [15].

2.6.1. Modelování ve stavovém prostoru

Zpětnovazební stavové řízení se zabývá návrhem regulátoru pro dynamický systém popsaný ve stavovém prostoru. Soustava je tedy charakterizována diferenciální rovnicí n-tého řádu:

$$a_n y^{(n)} + a_{n-1} y^{(n-1)} + \cdots + a_1 y' + a_0 y = u \tag{14}$$

Zavedením substituce $x_1 \equiv y$, $x_2 \equiv y'$, …, $x_n \equiv y^{(n)}$ lze tuto rovnici převést na n diferenciálních rovnic prvního řádu, které budeme nazývat stavovými rovnicemi.

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= x_3 \\ &\vdots \\ x_{n-1}' &= x_n \end{aligned} \tag{15}$$

$$x_n' = -\frac{a_0}{a_n} x_1 - \frac{a_1}{a_n} x_2 - \cdots - \frac{a_{n-1}}{a_n} x_n + \frac{1}{a_n} u$$

Jako výstup soustavy pak můžeme volit libovolnou stavovou proměnou, např.: $y = x_1$.

Stav soustavy je popsán stavovými proměnnými x_1, x_2, …, x_n, které společně tvoří stavový vektor: $\mathbf{x} = [x_1, x_2 \ldots x_n]^T$.

Ze znalosti stavového vektoru v čase $t = t_0$ a hodnoty vstupu pro $t \geq t_0$ lze určit chování pro každé $t \geq t_0$. Souřadnice stavových proměnných v n-rozměrném prostoru pak tvoří stavový prostor, v němž body představují možné stavy soustavy.

Soustava stavových rovnic a rovnice výstupu je obvykle vyjádřena maticově ve tvaru:

$$\boldsymbol{x' = Ax + Bu} \tag{16}$$

$$\boldsymbol{y = Cx + Du} \tag{17}$$

kde **A** … matice soustavy

B … matice vstupů

C … matice výstupů

D … matice vazby vstupů na výstup

x … stavový vektor

y … výstupní vektor

u … vstupní vektor

Dynamické chování soustavy je dáno vlastními čísly matice **A**. Vlastní čísla λ_i jsou kořeny charakteristické rovnice:

$$|\lambda \boldsymbol{I} - \boldsymbol{A}| = 0 \tag{18}$$

Kořeny jsou obecně komplexní čísla. Mohou nastat tyto případy:

- Kořeny jsou reálná záporná čísla – stabilní soustava
- Kořeny jsou reálná čísla, kdy alespoň jedno je kladné – nestabilní nekmitavá soustava
- Kořeny obsahují imaginární čísla, kdy všechna reálná čísla a všechny reálné části komplexních čísel jsou záporná čísla – stabilní kmitavá soustava
- Kořeny obsahují imaginární čísla, kdy alespoň jedno reálné číslo nebo alespoň jedna reálná část komplexního čísla je kladná – nestabilní kmitavá soustava
- Kořeny jsou imaginární čísla, kdy reálné složky jsou rovné nule – netlumené kmitání

Cílem stavové zpětnovazební regulace je tedy docílit takové matice soustavy, aby soustava byla stabilní a regulovaná veličina se ustálila na požadované hodnotě za co nejkratší čas, bez ustálené odchylky a ideálně bez překmitu.

2.6.2. Řiditelnost a pozorovatelnost soustavy

Než přistoupíme k vlastnímu návrhu regulátoru soustavy je potřeba ověřit, zda je soustava řiditelná a pozorovatelná.

Soustava je řiditelná tehdy, pokud pomocí vstupů **u** je možno soustavu převést ze stavu $\mathbf{x_0}$ do libovolného stavu $\mathbf{x_k}$. Existuje tedy vazba mezi vstupy **u** a stavovými proměnnými **x**, která umožňuje převod z jednoho bodu stavového prostoru do jeho libovolného dalšího bodu.

Nutná a postačující podmínka je, aby matice řiditelnosti $\mathbf{M_c}$ měla hodnost n, kde n představuje řád soustavy. To znamená, že determinanty matice $\mathbf{M_c}$ řádu n a menšího jsou nenulové a zároveň determinanty vyššího řádu jsou rovné nule.

$$\boldsymbol{M_c} = [\boldsymbol{B}\ \boldsymbol{AB}\ \boldsymbol{A^2B} \cdots \boldsymbol{A^{n-1}B}] \tag{19}$$

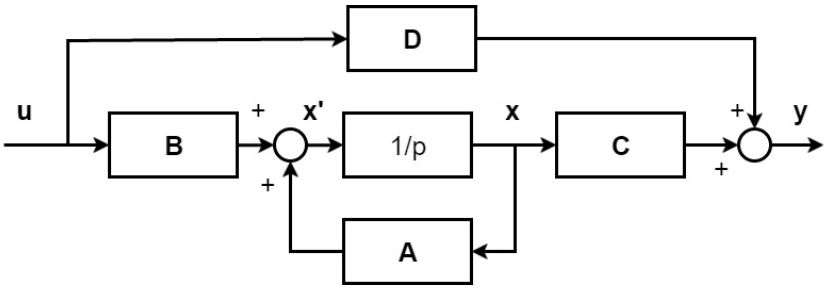
Soustava je pozorovatelná, pokud z pozorování výstupu **y**(t) v čase $t_0 \leq t \leq t_1$ lze určit každý stav $\mathbf{x}(t_0)$. Výstup soustavy **y** je tedy propojen se stavovými proměnnými **x** a je schopný o nich podat úplné informace.

Nutná a postačující podmínka pozorovatelnosti je, aby matice pozorovatelnosti $\mathbf{M_o}$ měla hodnost n.

$$\mathbf{Mo} = [\mathbf{C}\ \mathbf{CA}\ \mathbf{CA^2} \cdots \mathbf{CA^{n-1}}]^T \tag{20}$$

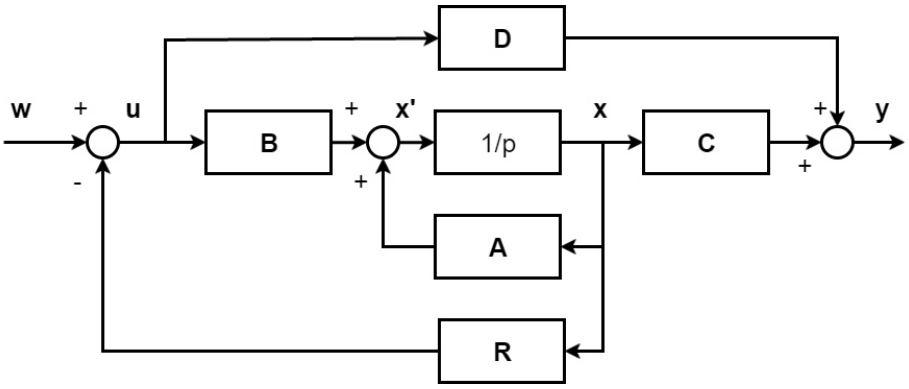
2.6.3. Metoda pole placement

Nyní máme k dispozici dynamickou soustavu popsanou stavovými rovnicemi (viz rovnicemi (16 a (17), u které předpokládáme, že je pozorovatelná a řiditelná.



Obrázek 2.10 - Grafické zobrazení stavových rovnic

Chování této soustavy je určeno maticí soustavy **A**, kdy její vlastní čísla tvoří póly v komplexní rovině, podle jejichž polohy usuzujeme stabilitu, vlastní frekvenci a tlumení. Stěžejní myšlenkou je proto rozšířit schéma na obrázku Obrázek 2.10 tak, aby došlo ke korekci pólů matice soustavy bylo tím docíleno změny dynamického chování soustavy požadovaným směrem.



Obrázek 2.11 - Grafické zobrazení soustavy se stavovým regulátorem

Ze schématu na obrázku Obrázek 2.11 lze poté odvodit následující stavové rovnice pro uzavřenou smyčku:

$$\boldsymbol{x' = Ax + B(-Rx + w) = (A - BR)x + Bw} \tag{21}$$

kde w … žádaná hodnota

Novou maticí soustavy se stává výraz (**A** - **BR**). Kořeny charakteristické rovnice pak získáme z rovnice:

$$|p\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{BR}| = 0 \tag{22}$$

Metoda pole placement spočívá ve volbě pólů p_1, p_2, …, p_n uzavřené smyčky tak, aby soustava vykazovala požadované chování. Poté je ze zvolených pólů dopočítána matice **R**, která představuje stavový regulátor. Toho je docíleno srovnáním koeficientů u rovnice (22 s požadovaným polynomem:

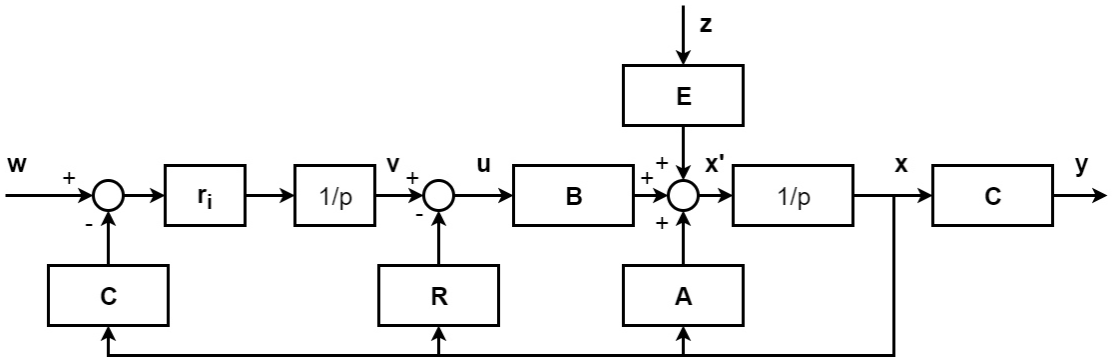
$$\prod_{i=1}^{n}(p - p_i) = p^n + a_{n-1}p^{n-1} + \cdots + a_1 p + a_0 \tag{23}$$

Prakticky se matice **R** získá pomocí funkce place v prostředí Matlab, u které jako vstupní argumenty vystupují matice **A**, **B** a požadované póly **p** a výstupem funkce je matice **R**. Nevýhodou tohoto typu regulace je odchylka v ustáleném stavu.

2.6.4. Integrátor na vstupu

Regulátor v klasické struktuře stavové zpětnovazební regulace (viz obrázek Obrázek 2.11 - Grafické zobrazení soustavy se stavovým regulátorem) pracuje obdobně jako P-regulátor, kdy je rozdíl mezi žádanou a skutečnou hodnotou regulované veličiny násoben proporcionálním členem. V takovém případě se ovšem při působení poruchy objevuje nenulová odchylka v ustáleném stavu. Proto je žádoucí regulátor rozšířit o integrační složku. Běžně se proto využívá těchto regulačních struktur:

- Regulátor s integrátorem na vstupu.
- Struktura s pozorovatelem poruchy (integrátor je součástí pozorovatele (viz kapitola 2.6.6).



Obrázek 2.12 - Struktura s integrátorem na vstupu

Stavové rovnice podle obrázku Obrázek 2.13 pak vypadají takto:

$$\boldsymbol{x'} = \boldsymbol{Ax} + \boldsymbol{Bu} + \boldsymbol{Ez} = (\boldsymbol{A} - \boldsymbol{BR})\boldsymbol{x} + \boldsymbol{Bv} + \boldsymbol{Ez} \tag{24}$$

$$\boldsymbol{v'} = \boldsymbol{r_i}(\boldsymbol{w} - \boldsymbol{Cx}) \tag{25}$$

kde z … porucha na vstupu

V maticové formě dostáváme rovnici:

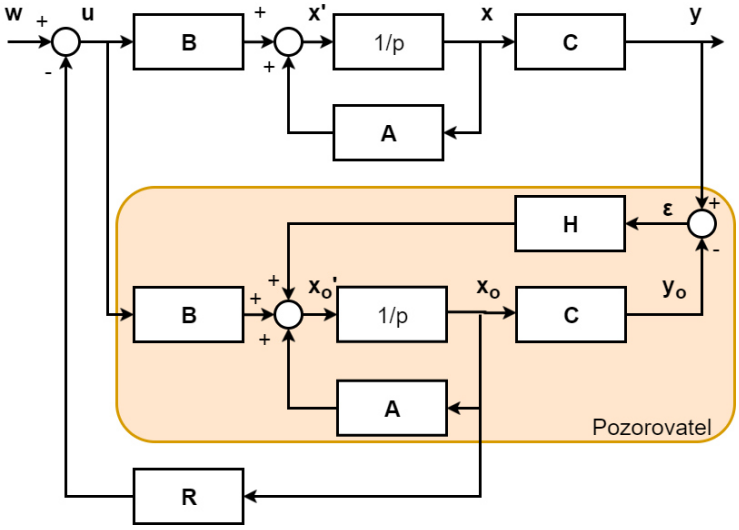
$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{v'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A - BR} & \boldsymbol{B} \\ \boldsymbol{-r_i C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{r_i} \end{bmatrix} \boldsymbol{w} + \begin{bmatrix} \boldsymbol{E} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{z} \tag{26}$$

Jako neznámé ve výše uvedené rovnici vystupují matice **R** a $\mathbf{r_i}$, které představují parametr regulátoru a lze je získat metodou pole placement, po dalších úpravách (viz [15]) dostáváme vztah:

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{v'} \end{bmatrix} = \left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{-C} & \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{-D} \end{bmatrix} \begin{pmatrix} \boldsymbol{-R} & \boldsymbol{r_i} \end{pmatrix} \right) \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{v} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{1} \end{bmatrix} \boldsymbol{w} \tag{27}$$

2.6.5. Pozorovatel

Výše uvedené struktury pracují s celým stavovým vektorem, což by znamenalo nutnost měřit v reálném čase všechny stavové proměnné. To ovšem v reálném světě ve většině případů není možné a to buď z technického, nebo z ekonomického hlediska. Z tohoto důvodu je výhodné použít strukturu s tzv. pozorovatelem, který ze známých vstupů a výstupů soustavy rekonstruuje stavové proměnné. Pozorovatel je v podstatě model běžící současně s regulovanou soustavou poskytující regulátoru informace o stavových proměnných bez nutnosti měřit každou z nich.



Obrázek 2.13 - Struktura s pozorovatele

Na obrázku Obrázek 2.13 si můžeme všimnout, že do pozorovatele vstupuje vektor vstupů **u**, výstup soustavy **y** (snímané veličiny) a výstupem pozorovatele je odhad stavového vektoru $\hat{\boldsymbol{x}}$ (v obrázku Obrázek 2.13 dolní indexy o představují veličiny se stříškou), který je použit pro výpočet akčního zásahu pomocí matice **R**. Místo s naměřenými stavovými veličinami řídicí člen pracuje s jejich odhady, které provádí pozorovatel. Model soustavy aproximuje reálné zařízení pouze s určitou přesností, proto je tato nepřesnost „dotahována“ maticí **H**, která pracuje s rozdílem skutečného výstupu soustavy a jeho rekonstrukcí. Matice **H** se nazývá zpětnovazební matice pozorovatele.

Stavové rovnice u struktury s pozorovatelem pak mají následující tvar:

$$\boldsymbol{x'} = \boldsymbol{Ax} + \boldsymbol{Bu} \tag{28}$$

$$\boldsymbol{y} = \boldsymbol{Cx} \tag{29}$$

$$\hat{\boldsymbol{x}}' = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{Bu} + \boldsymbol{H}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{30}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{C}\hat{\boldsymbol{x}} \tag{31}$$

Po úpravě na standartní tvar (viz [15]) dostáváme:

$$\hat{\boldsymbol{x}}' = (\boldsymbol{A} - \boldsymbol{HC})\hat{\boldsymbol{x}} + \boldsymbol{Bu} + \boldsymbol{Hy} \tag{32}$$

Matice **F** = (**A** - **HC**) se nazývá systémová matice pozorovatele. Tato matice určuje jeho odezvu, která musí být rychlejší než odezva regulované soustavy. Proto při návrhu musí být zajištěna větší zápornost pólů matice **F** než matice **A**.

2.6.6. Pozorovatel poruchy

Výše odvozeného pozorovatele dále rozšíříme tak, aby navíc detekoval poruchy vstupující do řízené soustavy (viz obrázek Obrázek 2.14) a korigoval tak odhadované stavy nejen na základě odchylky výstupů soustavy a pozorovatele, ale navíc i podle odhadu poruchy. Rozšířený model soustavy je pak ve tvaru:

$$\boldsymbol{x}' = \boldsymbol{Ax} + \boldsymbol{Bu} + \boldsymbol{Ez} \tag{33}$$

$$\boldsymbol{y} = \boldsymbol{Cx} + \boldsymbol{Du} \tag{34}$$

a model pozorovatele vypadá takto:

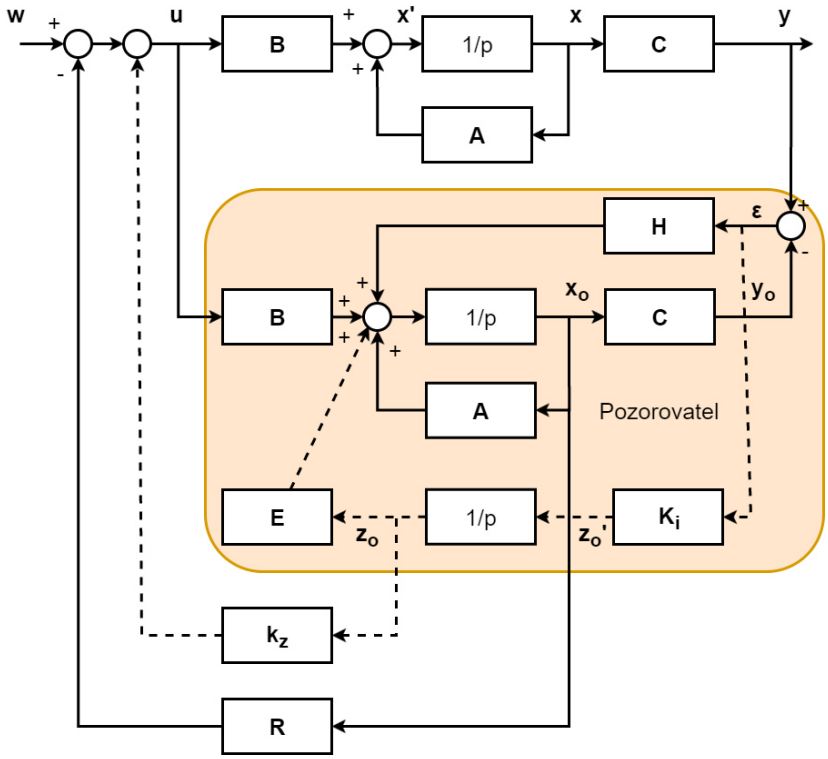
$$\hat{\boldsymbol{x}}' = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{Bu} + \Delta\boldsymbol{x} \tag{35}$$

kde Δx představuje rozdíl odhadovaných stavů od stavů soustavy včetně poruchy.

$$\Delta\boldsymbol{x} = \boldsymbol{H}(\boldsymbol{y} - \hat{\boldsymbol{y}}) + \boldsymbol{E}\hat{\boldsymbol{z}} \tag{36}$$

Odhad poruchy $\hat{\boldsymbol{z}}$ získáme ze stavové rovnice, jíž rozšíříme stavový model pozorovatele.

$$\hat{\boldsymbol{z}}' = \boldsymbol{K_i}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{37}$$



Obrázek 2.14 - Schéma pozorovatele se sledováním poruchy

Model pozorovatele pak vypadá následovně:

$$\begin{bmatrix}\hat{\boldsymbol{x}}'\\ \hat{\boldsymbol{z}}'\end{bmatrix} = \begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{x}}\\ \hat{\boldsymbol{z}}\end{bmatrix} + \begin{bmatrix}\boldsymbol{B}\\ \boldsymbol{0}\end{bmatrix}\boldsymbol{u} + \begin{bmatrix}\boldsymbol{H}\\ \boldsymbol{K_i}\end{bmatrix}(\boldsymbol{y}-\hat{\boldsymbol{y}}) + \begin{bmatrix}\boldsymbol{0} & \boldsymbol{E}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{x}}\\ \hat{\boldsymbol{z}}\end{bmatrix} \tag{38}$$

Po sérii algebraických úprav (detailně rozebráno v [15]) model upravíme do tvaru vhodného pro použití funkce place a získáváme matici pozorovatele **H** a matici $\mathbf{K_i}$.

$$\begin{bmatrix}\hat{\boldsymbol{x}}'\\ \hat{\boldsymbol{z}}'\end{bmatrix} = \left(\begin{bmatrix}\boldsymbol{A} & \boldsymbol{E}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}^{\boldsymbol{T}} - [\boldsymbol{C} \quad \boldsymbol{0}]^{\boldsymbol{T}}\begin{bmatrix}\boldsymbol{H}\\ \boldsymbol{K_i}\end{bmatrix}^{\boldsymbol{T}}\right)^{\boldsymbol{T}}\begin{bmatrix}\hat{\boldsymbol{x}}\\ \hat{\boldsymbol{z}}\end{bmatrix} + \left(\begin{bmatrix}\boldsymbol{B} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} - \begin{bmatrix}\boldsymbol{H}\\ \boldsymbol{K_i}\end{bmatrix}[\boldsymbol{D} \quad -\boldsymbol{I}]\right)\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{y}\end{bmatrix} \tag{39}$$

Pro póly matice pozorovatele $\begin{bmatrix}\boldsymbol{H}\\ \boldsymbol{K_i}\end{bmatrix}$ musí opět platit jejich větší zápornost oproti pólům soustavy, aby byl pozorovatel rychlejší než řízená soustava.

2.6.7. LQR

Metoda Linear Quadratic Regulator slouží k návrhu stavového regulátoru, který je u složitějších soustav intuitivnější než návrh pomocí metody pole placement. Nevýhoda metody pole placement spočívá ve způsobu volby pólů tak, abychom maximálně využili reálné vstupy (napájení) do soustavy a dosáhli nejlepší možné odezvy systému. U dobře známých soustav například systém druhého řádu (v mechanice těleso na pružině s tlumičem) se dá poměrně dobře odhadovat chování soustavy. U složitějších soustav se s výhodou používá metoda LQR, kdy „penalizujeme“ stavové proměnné a vstupy do soustavy, čímž při ladění regulátoru vidíme vliv změny jednotlivých parametrů na celkovou odezvu soustavy.

U spojité soustavy je zpětná vazba tvořena výrazem **u** = -**Kx**, kde matici **K** nazýváme maticí zpětných vazeb. Penalizace stavových proměnných a vstupů je nastavena v maticích **Q** (matice vah stavů) a **R** (matice vah vstupů).

Hlavní myšlenkou algoritmu (detailnější matematický popis v [16]) je minimalizovat váhovou funkci ve tvaru:

$$J = \int_0^\infty (\boldsymbol{x^T Q x} + \boldsymbol{u^T R u}) dt \tag{40}$$

Řešením této rovnice získáváme vztah pro matici **K**:

$$\boldsymbol{K = R^{-1} B^T P} \tag{41}$$

kde matice **P** je řešením rovnice:

$$\boldsymbol{A^T P + P A - P B R^{-1} B^T P + Q = 0} \tag{42}$$

Pro posouzení kvality regulace jsou obvykle definována následující kritéria:

- Rychlost náběhu na požadovanou hodnotu – implicitně nastavujeme výrazem $\mathbf{x^T Q x}$.
- Doba ustálení na požadované hodnotě - implicitně nastavujeme výrazem $\mathbf{x^T Q x}$.
- Překmit, oscilace, míra tlumení – záporná i kladná odchylka je regulována stejnou mírou a to díky použití kvadratického kritéria.
- Ustálená odchylka se v nekonečnu blíží nule díky minimalizaci váhové funkce.
- Špičky na vstupu do soustavy jsou tlumeny díky použití kvadratického kritéria, tím pádem velké hodnoty napájecí veličiny jsou více utlumeny.

Výsledný systém je za splnění podmínky řiditelnosti stabilní. Výše uvedené výpočty nejsou podle [8] obecně řešitelné s pomocí tužky a papíru, proto je vhodné pro získání matice **K** využít počítač. Nabízí se například funkce lqr implementovaná v Matlabu ve tvaru:

$$\boldsymbol{K = lqr(A, B, Q, R)}$$

kdy z matic soustavy, vstupů a matic vah je vypočítána matice zpětných vazeb.

2.7. Převod do diskrétní podoby

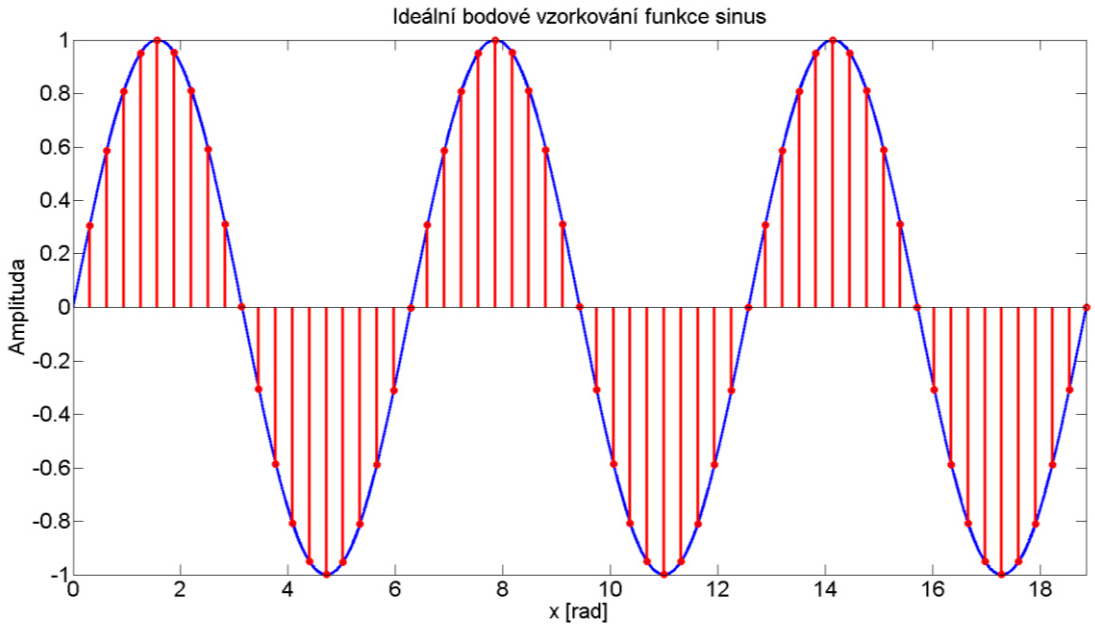
Stejně jako mikroprocesory se FPGA řadí mezi digitální techniku, proto na těchto čipech není možná práce ve spojitém čase. Vstupní a výstupní signály jsou převáděny pomocí analogově-digitálních (A/D) a digitálně-analogových (D/A) převodníků, tudíž výpočet na FPGA probíhá v diskrétních krocích při dané periodě, přičemž připojená soustava vykazuje spojitý charakter. Mezi důležité charakteristiky A/D a D/A převodníků patří:

- Rozlišení – udáváno v bitech, kdy rozsah převáděné veličiny je rozdělen na 2^n (n je počet bitů).
- Rychlost převodu – v závislosti na rozlišení definuje kolik vzorků je převodník schopen převést za sekundu (SPS – samples per second)

Podle [17] pro rozlišení 24 bitů převodníky obvykle dosahují rychlostí od 1 SPS do 10 kSPS a pro rozlišení 8 bitů lze dosáhnout až 10 GSPS.

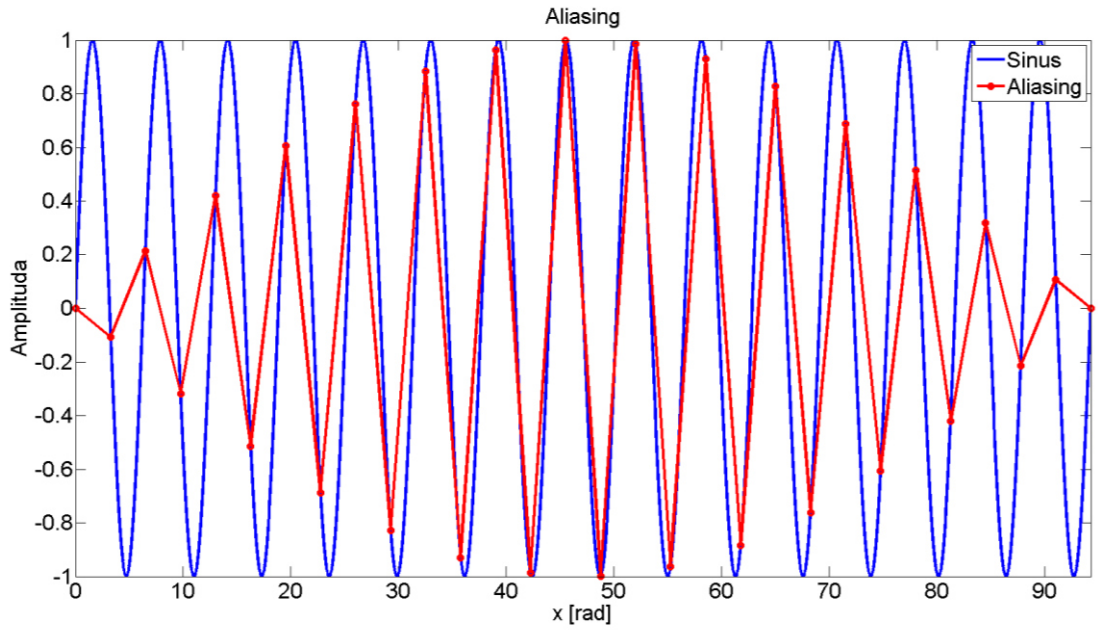
2.7.1. Vzorkování a kvantování signálu

Převod spojitého signálu na posloupnost jeho okamžitých hodnot v časově rovnoměrných krocích se nazývá vzorkování. Za splnění určitých podmínek plně reprezentuje signál, což souvisí zejména s omezením týkajícího se konečné rychlosti převodu. Následující text čerpá z [17] a [18].



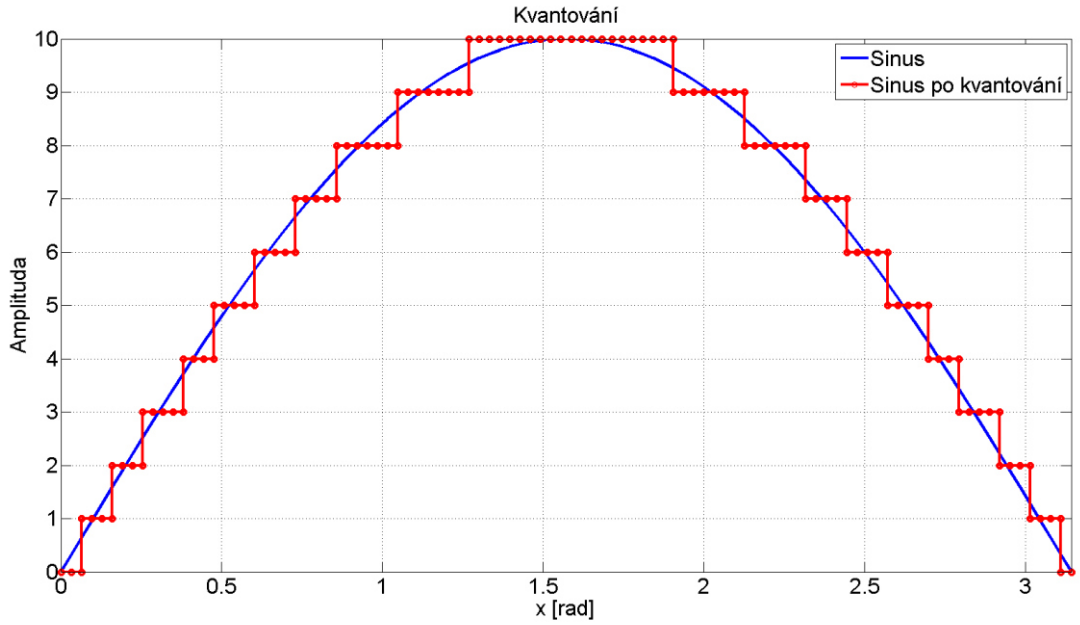
Obrázek 2.15 - Ukázka ideálního bodového vzorkování v Matlabu

Mezi další metody vzorkování patří například vzorkování s přidržením nebo klíčování. Je-li signál o frekvenci f_s vzorkován frekvencí f_{vz}, která není alespoň dvakrát vyšší než f_s, pak získaný signál neodpovídá tomu původnímu. Dochází ke zkreslení z důvodu nedodržení Nyquistova teorému a tento jev je nazýván aliasing. Tento teorém se zabývá pouze rekonstrukcí frekvence, pro zachování amplitudy je nutné vzorkovat alespoň desetkrát vyšší frekvencí.



Obrázek 2.16 - Ukázka aliasingu v Matlabu

Kvantování neboli diskretizace v amplitudě je dalším krokem, kdy dochází k aproximaci amplitudy nejbližší diskrétní hodnotou. Celkový rozsah je rozdělen na konečný počet hodnot, který odpovídá 2^n stupňů, kde n je rozlišení A/D převodníku.



Obrázek 2.17 - Ukázka výsledku kvantování v Matlabu

2.7.2. Převod spojitého stavového modelu na diskrétní

Spojitý stavový model reprezentovaný rovnicemi (16 a (17 nemůže být implementován na FPGA, proto je nezbytné provést převod do diskrétní podoby ve tvaru:

$$\boldsymbol{x}(k+1) = \boldsymbol{G}\boldsymbol{x}(k) + \boldsymbol{H}\boldsymbol{u}(k) \tag{43}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \tag{44}$$

kde k … index časového kroku

G … diskrétní stavová matice

H … diskrétní vstupní matice

C … výstupní matice

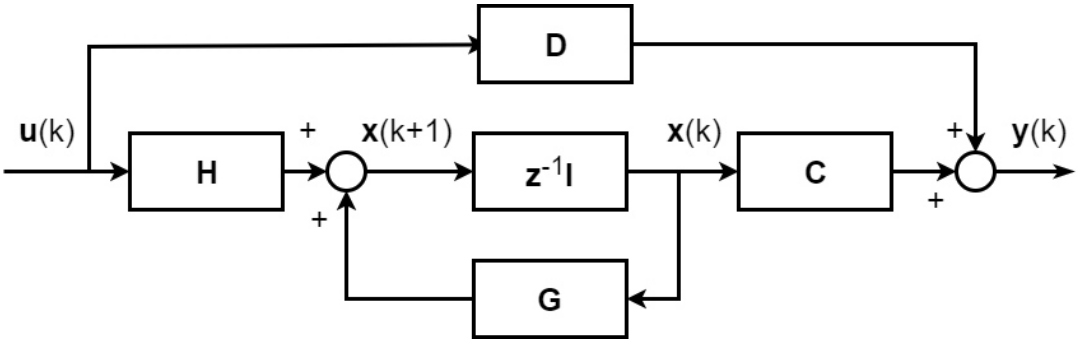
D … matice vazby vstupů na výstup

x … stavový vektor

u … vstupní vektor

y … výstupní vektor

Tímto krokem je spojitá operace integrace nahrazena operacemi v diskrétní podobě.



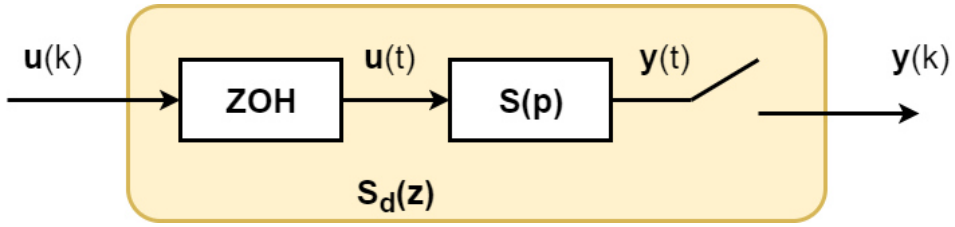
Obrázek 2.18 - Schéma stavového modelu v diskrétní podobě

Protože výše odvezené spojité modely (viz rovnice (16, (17 a (38) jsou v Matlabu ve struktuře stavového modelu, nabízí se možnost použít funkci c2d (převod stavového modelu nebo přenosové funkce ze spojité do diskrétní domény). Tato funkce vyžaduje jako vstup stavový model, periodu a metodu konverze ve tvaru:

soustava_diskrétní = c2d(soustava_spojitá, t_s, metoda)

V potaz byly vzaty následující metody diskretizace modelu.

Zero-Order-Hold (ZOH neboli přímá diference, viz [19] a [20]) nabízí výpočetně jednodušší řešení (matice **C** a **D** se převodem neměnní) a je vhodná pro soustavy, u nichž se očekává buzení schodovitým signálem. Nevýhodou je možnost ztráty stability po diskretizaci. Proto je potřeba uzavřenou smyčku testovat v diskrétním stavu.



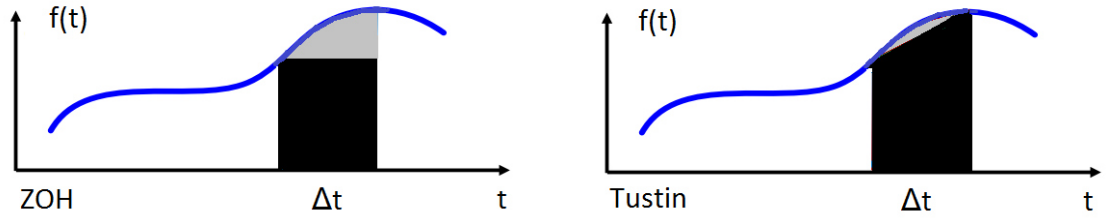
Obrázek 2.19 - Metoda ZOH (spojitá soustava je převedena do diskrétního tvaru)

ZOH blok generuje spojitý signál u(t), kdy na začátku nové periody přečte hodnotu na vstupu a po zbytek časového kroku ji drží konstantní. Obdobně je na výstupu soustavy po dobu trvání periody konstantní výstup y(k), který je získán vzorkováním signálu y(t). Blok S(p) představuje soustavu ve spojitém čase, zatímco $S_d(z)$ v diskrétní doméně.

Tustinova (neboli bilineární) metoda dosahuje velmi dobré shody mezi spojitou a diskrétní odezvou ve frekvenční oblasti. Algoritmus je tedy výhodný pro soustavy, u kterých hraje významnou roli dynamika v konkrétních frekvencích. Stabilita je převodem zachována ovšem za cenu vyššího počtu výpočtů. Jedná se o aproximaci vztahu mezi Laplaceovou a z transformací ve tvaru:

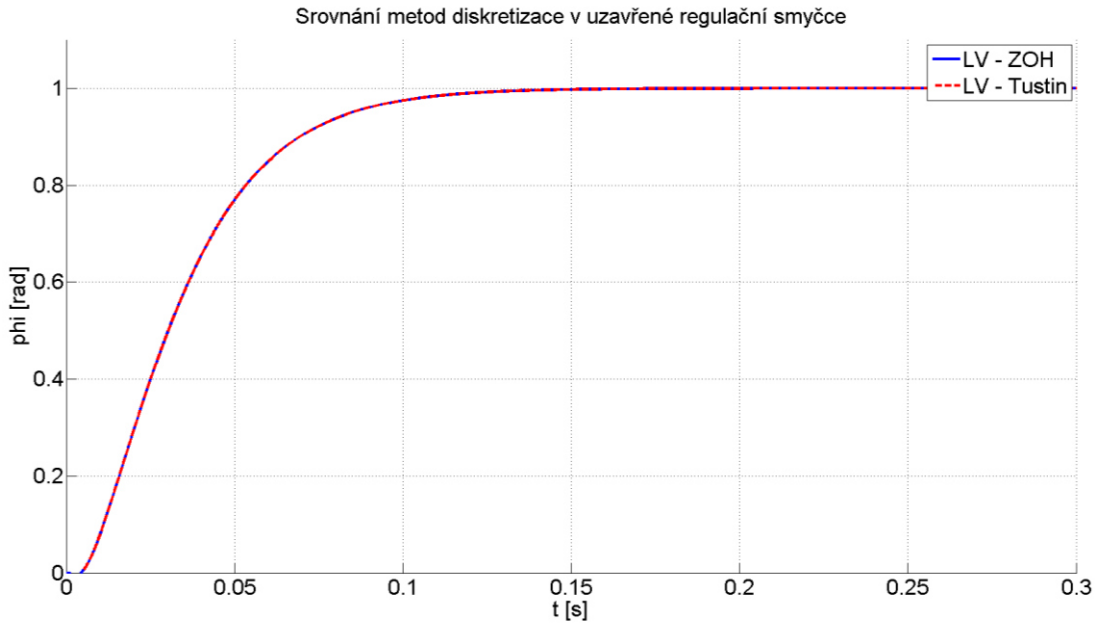
$$z = e^{pT_s} \approx \frac{1 + \frac{pT_s}{2}}{1 - \frac{pT_s}{2}} \tag{45}$$

Základním rozdílem mezi těmito metodami je výpočet integrace, kdy ZOH provádí tzv. čtvercovou a Tustinova lichoběžníkovou aproximaci pro daný segment.



Obrázek 2.20 - Aproximace integrace pomocí ZOH a Tustinovy diskretizační metody

Na obrázku Obrázek 2.21 šedá plocha společně s černou značí určitý integrál a černá jeho aproximaci pomocí dané metody. Z důvodu téměř identických výsledků při testování regulace v LabVIEW byla zvolena pro FPGA výpočtově jednodušší metoda ZOH.



Obrázek 2.21 - Srovnání vlivu volby metody diskretizace regulátoru na výslednou regulaci

2.7.3. Aritmetika s pevnou desetinnou čárkou

Dalším prvkem, který souvisí s prací v diskrétní doméně, je reprezentace dat v počítači. Nejlevnější mikroprocesory a FPGA nemají tzv. floating-point unit (FPU – jednotka pro výpočty v plovoucí desetinné čárce), proto je nutné reprezentovat čísla a provádět výpočty ve formátu s pevnou desetinnou čárkou (fixed-point).

Ve formátu s plovoucí desetinnou čárkou je číslo reprezentováno pomocí mantisy (udává přesnost čísla) a exponentu (udává velikost čísla), což odpovídá vědecké notaci (například $1{,}23456789 \times 10^3$). Čísla jsou na číselné ose rozložena nerovnoměrně. Jejich hodnota je pak výsledkem operace (uvažována koncepce, kdy desetinná čárka je umístěna za nejvýznamnější číslicí):

$$hodnota = m \cdot z^e \tag{46}$$

kde m … mantisa

z … základ soustavy

e … exponent

Oproti tomu u reprezentace čísel pomocí pevné desetinné čárky je pevně stanoven počet cifer před a za desetinnou čárkou. Čísla jsou tedy na číselné ose rozložena rovnoměrně.

Velkou výhodou reprezentace pomocí plovoucí desetinné čárky je dynamický rozsah čísel, což znamená, že je možné zvýšit rozsah (více bitů exponentu) nebo přesnost (více bitů mantisy), což u reprezentace s pevnou desetinnou čárkou není možné. Detailnější rozbor v [21].

FPGA nemá předdefinovanou žádnou délku slova, proto si uživatel může libovolně navolit formát čísel s pevnou desetinnou čárkou. Čím méně bitů bude použito, tím rychleji bude program

pracovat a s menší pravděpodobností dojde k vyčerpání kapacity hardwarových prostředků, na druhou stranu při příliš málo bitech dochází ke ztrátě přesnosti a stability výpočtu. Zejména operace násobení je problematická, protože aby byla zachována správnost výsledku, pak počet bitů součinu se musí rovnat součtu bitů obou součinitelů. Jelikož výsledný regulátor v sobě zahrnuje pozorovatele, který v sobě má zabudovanou korekci poruchy, objevuje se v něm spousta násobení, a proto úvahy o volbě formátu čísel nabývají velkého významu.

3. Vytvoření řídicího systému

3.1. Návrh a vytvoření regulátoru v prostředí Matlab/Simulink

Tato kapitola si klade za cíl navrhnout vhodný regulátor polohy, převést soustavu i regulátor do diskrétní podoby a otestovat kvalitu regulace.

3.1.1. Vytvoření modelu soustavy

Při vytváření modelu soustavy vyjdeme z dynamických rovnic DC motoru (viz 2.5.2), parametrů zadaného motoru Maxon (v technické dokumentaci typ 273759, viz [30]) a stavového popisu (viz 29). Parametry motoru v dané dokumentaci nejsou v jednotkách SI, proto je nejprve potřeba jednotky převést. Parametry po převodu jsou uvedeny v následující tabulce.

Parametr	U [V]	$c\phi_i$ [N·m·A^{-1}]	$c\phi_\omega$ [V·s]	R [Ω]	L [H]	J [kg·m^2]	t_a [s]
Hodnota	48	0,119	0,7453	11,5	0,00316	$6{,}55\cdot10^{-6}$	$2{,}8\cdot10^{-4}$

Tabulka 2 – Parametry použitého DC motoru

kde U … napětí

$c\phi_i$ … proudová konstanta motoru

$c\phi_\omega$ … otáčková konstanta motoru

R … odpor kotvy

L … indukčnost kotvy

J … moment setrvačnosti

t_a … elektrická časová konstanta

Stavový model sestává z dynamických rovnic, kdy na levé straně figurují derivace stavových proměnných a z pravých stran se formují jednotlivé matice **A**, **B**, **C** a **D**. Z důvodu aplikace polohového regulátoru bude třetí rovnicí vyjádření natočení rotoru jako integrál rychlosti. Stavové rovnice pak mají následující tvar:

$$\frac{d\theta}{dt} = \omega \tag{47}$$

$$\frac{d\omega}{dt} = \frac{1}{J}(c\phi_i \cdot i - mz) \tag{48}$$

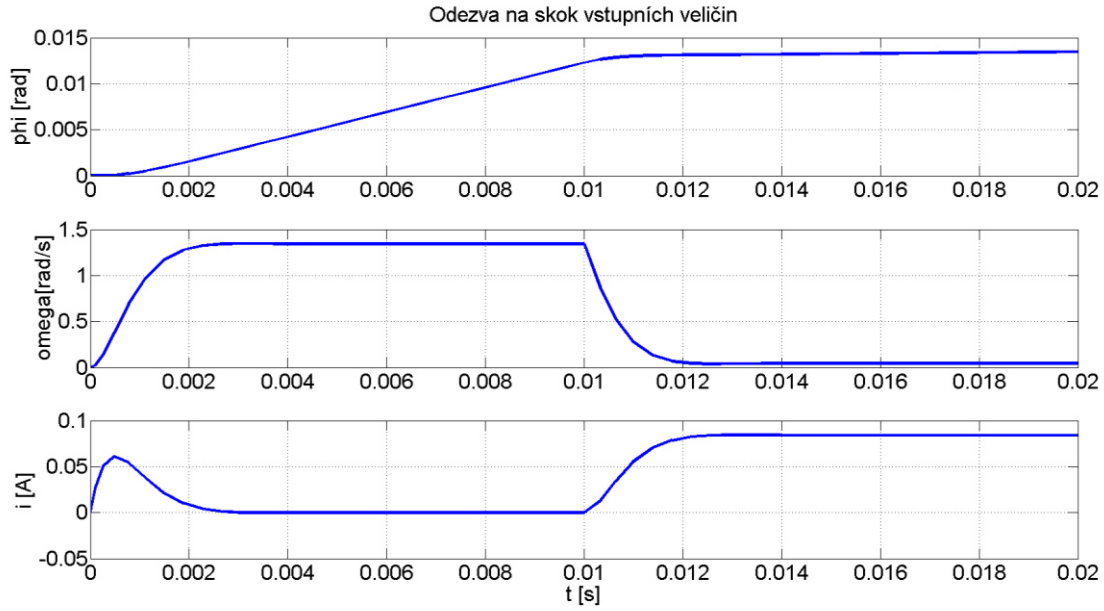
$$\frac{di}{dt} = \frac{1}{L}(u - R \cdot i - c\phi_\omega \cdot \omega) \tag{49}$$

Definujeme-li stavový vektor jako $\mathbf{x} = [\phi, \omega, i]^T$ a vektor vstupů jako $\mathbf{u} = [u, m_z]^T$, lze pak snadno z výše uvedených rovnic odvodit následující matice, které slouží jako vstup do funkce ss (stavový model) v Matlabu. Na tento model se pak lze snadno odkázat ze Simulinku pomocí bloku LTI (lineární časově invariantní model).

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & \frac{c\Phi_i}{J} \\ 0 & -\frac{c\Phi_\omega}{L} & -\frac{R}{L} \end{pmatrix} \quad \boldsymbol{B} = \begin{pmatrix} 0 & 0 \\ 0 & -\frac{1}{J} \\ \frac{1}{L} & 0 \end{pmatrix} \quad \boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{D} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Vstupem do soustavy je regulované napětí u a neznámý zátěžný moment m_z. Suché a viskózní tření budeme považovat za součást m_z, jelikož výsledný regulátor bude zátěžný moment odhadovat a korigovat.

Po kontrole řiditelnosti a pozorovatelnosti tedy získáváme model motoru, se kterým budeme dále pracovat při návrhu a testování regulátoru. Vlastní čísla matice A (pro úhlovou rychlost a proud) nabývají hodnot -1819,6 ± 986,9i. Jedná se tedy o poměrně rychlou a stabilní soustavu.

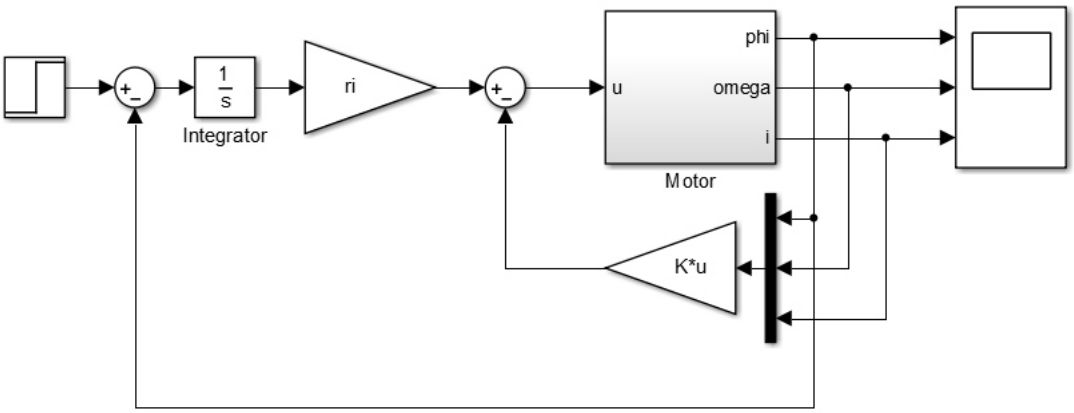


Obrázek 3.1 – Odezva modelu motoru na skok napětí a momentu

Chování soustavy dopadlo podle očekávání. V čase t = 0 bylo přivedeno napětí 1 V, nastal přechodový děj, otáčky se ustálily a hodnota natočení začala lineárně narůstat. V čase t = 0,01 byl přiveden skok momentu o velikosti m_z = 0,01, po kterém nastal další přechodový děj, a po ustálení se otáčky snížily a narostl proud.

3.1.2. Návrh zpětnovazebního stavového regulátoru s integrací na vstupu

Pro potřebu návrhu regulátoru byl vytvořen ještě další model motoru, který se liší pouze zanedbáním zátěžného momentu na vstupu, tím pádem je z matic **B** a **D** odebrán druhý sloupec. Pokud tedy naladíme regulátor na této soustavě, nebude si umět dobře poradit při regulaci motoru s proměnlivým zátěžným momentem, což bude řešeno dále. Následující schéma v Simulinku vychází z poznatků z 2.6.4.



Obrázek 3.2 – Schéma regulace s integrátorem na vstupu (Simulink)

Nyní je potřeba napočítat matici **K** a zesílení r_i. Vyjdeme z rovnice (27, kdy můžeme přímo aplikovat metodu pole placement. Ovšem pro intuitivnější návrh byla zvolena metoda LQR, která usnadňuje volbu pólů uzavřené smyčky.

```
%% Integrátor na vstupu
A_I = [A zeros(3, 1); -C 0];
B_I = [B; -D];
% Návrh regulátoru
k1 = p1 * 0.1592;
k2 = p2 * 0.0155;
k3 = p3 * 0.2396;
k4 = p4 * 1;

Q = [k1 0 0 0; 0 k2 0 0; 0 0 k3 0; 0 0 0 k4];   % Penalizace stavů
Ref = p5 * 0.0208;                               % Penalizace vstupů

K_LQR = lqr(A_I, B_I, Q, Ref);
ri = -K_LQR(:, 4);
K_LQR = K_LQR(1:3);
```

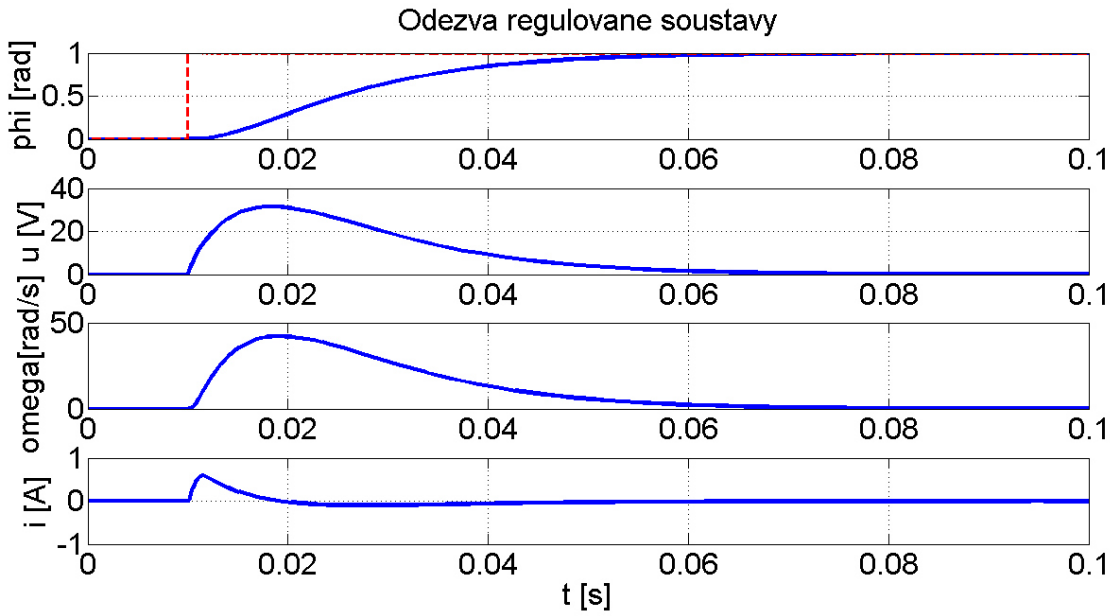
Obrázek 3.3 – Ukázka návrhu regulátoru metodou LQR v Matlabu

Z kódu na obrázku Obrázek 3.3 – Ukázka návrhu regulátoru metodou LQR je zřejmé, že koeficienty k_i a Ref jsou napočítány jako parametr krát normalizovaná hodnota penalizace. Normalizovanou hodnotou je myšlena převrácená hodnota rozsahu veličiny (pro $p_i = 1$ je tedy při součin maximální hodnoty dané veličiny a k_i roven jedné). Ladění regulátoru pak probíhá úpravou parametrů p_i. Nakonec je zavolána funkce lqr, která vrací hledanou matici **K** a zesílení r_i.

Požadavky na naladění polohového regulátoru:

- nulový překmit
- maximální rychlost ustálení na požadované hodnotě
- minimální hodnoty prvků matice **K** a zesílení r_i z důvodu výpočetní náročnosti operace násobení na FPGA

Tyto požadavky jsou vzájemně v rozporu, proto je výsledné naladění kompromisem s výjimkou nulového překmitu, který je vždy dodržen.



Obrázek 3.4 – Odezva uzavřené smyčky s regulátorem naladěným pomocí LQR

Na obrázku Obrázek 3.4 – Odezva uzavřené smyčky s regulátorem naladěným pomocí LQR je zobrazena reakce modelu motoru na jednotkový skok natočení (1 radián, který odpovídá cca. 57,3°), kdy je dosaženo nulového překmitu a soustava se ustálí na požadované hodnotě pod desetinu sekundy.

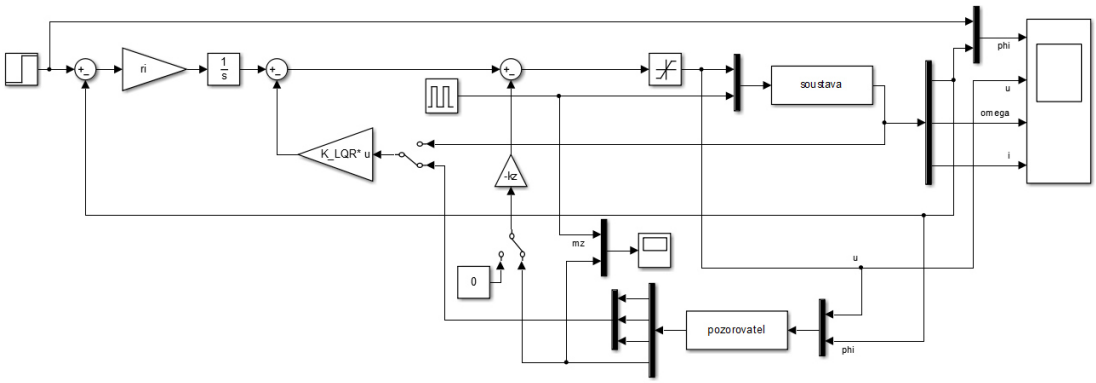
3.1.3. Vytvoření pozorovatele s korekcí poruchy

Regulátor navrhnutý v 2.6.4 vyžaduje na vstupu znalost všech stavových veličin. Měření všech těchto fyzikálních veličin na reálném motoru je nepraktické, drahé a nemusí být možné. Proto budeme uvažovat pouze snímání natočení inkrementálním enkodérem a zbytek stavů bude odhadován pozorovatelem, který bude navíc rozšířen o odhadování poruchy, v našem případě zejména zátěžného momentu. Následující kód vychází z (39.

```
%% Pozorovatel
E = [0; -1/J; 0];
pp = p*[-2000, -4000, -6000, -8000];
A_P = [A E; zeros(1, 3) 0];
B_P = [C 0];

H = place(A_P', B_P', pp)';

A_P = (A_P - H * B_P);
B_P = [B zeros(3, 1); 0 0] - H * [D -1];
C_P = eye(4);
D_P = zeros(4,2);

pozorovatel = ss(A_P, B_P, C_P, D_P);
kz = (R + K_LQR(3))/ci;
```

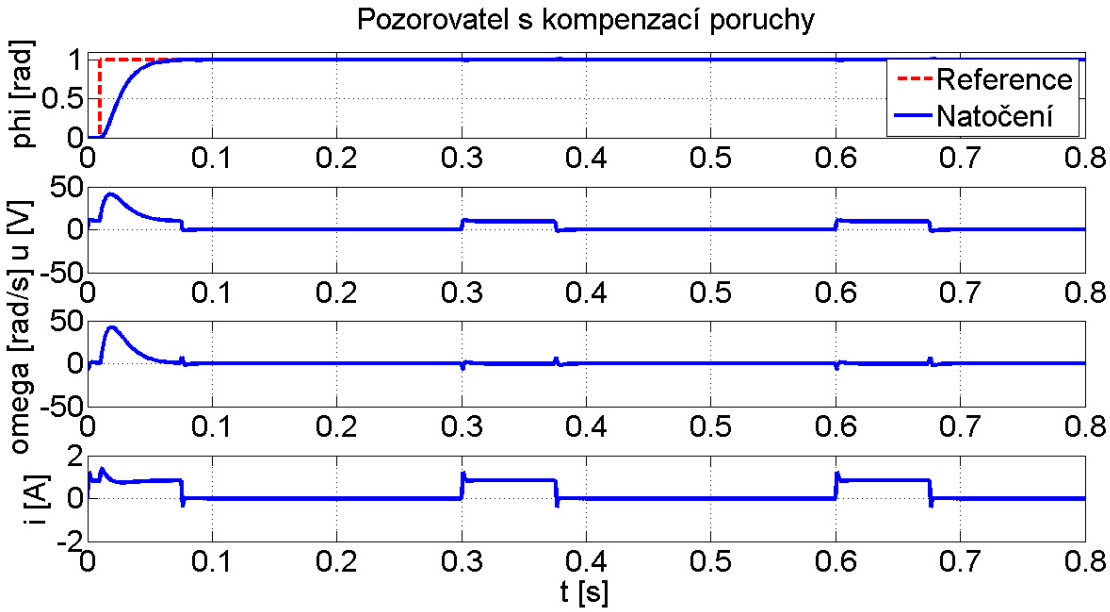
Obrázek 3.5 – Ukázka návrhu pozorovatele v Matlabu

Hledaná je matice **H**, kterou lze opět po vhodných předchozích úpravách získat pomocí funkce place. Pozorovatel musí mít rychlejší čas odezvy, proto jeho póly musí být zápornější (doporučuje se 4x až 10x, viz [14]), což je dosaženo parametrem p, který slouží k jejich optimalizaci (rychlejší soustavy jsou náročnější na výpočet). Zapojení v Simulinku je ukázáno na obrázku Obrázek 3.6 – Pozorovatel s kompenzací poruchy (Simulink).

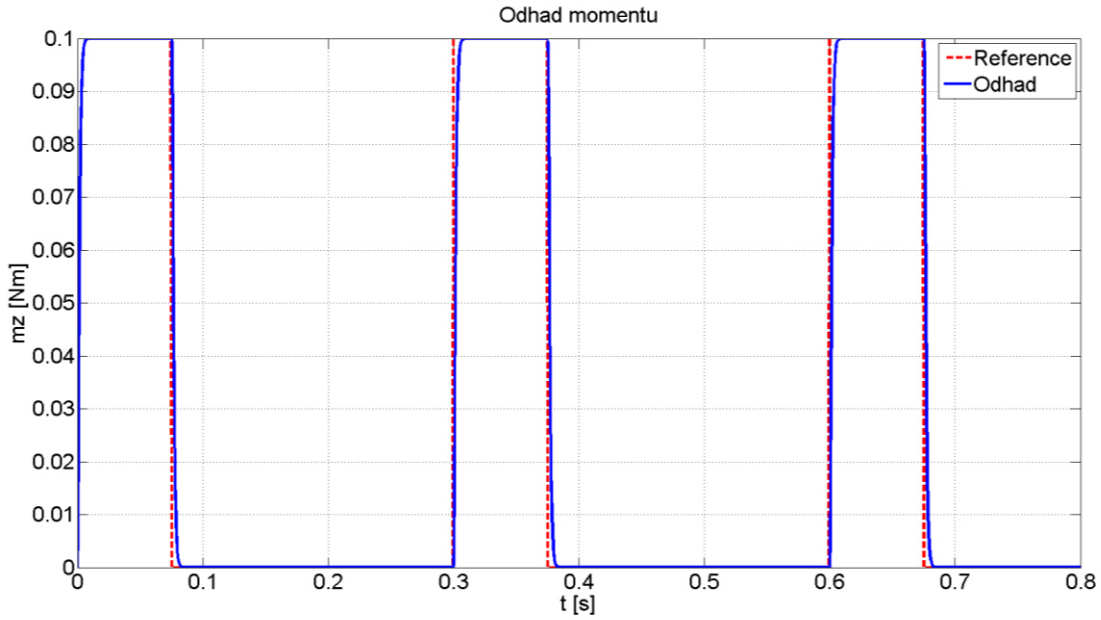


Obrázek 3.6 – Pozorovatel s kompenzací poruchy (Simulink)

Struktura byla rozšířena o pozorovatele, který pracuje s hodnotou napětí vstupujícího do motoru a měřeným natočením hřídele motoru. Na výstupu generuje odhady natočení, úhlové rychlosti, proudu a zátěžného momentu. S odhady stavových veličin poté pracuje regulátor navržený pomocí metody LQR s integrátorem na vstupu a odhad zátěžného momentu je konstantou k_z převeden na ekvivalentní napětí a připočten k výstupu regulátoru.

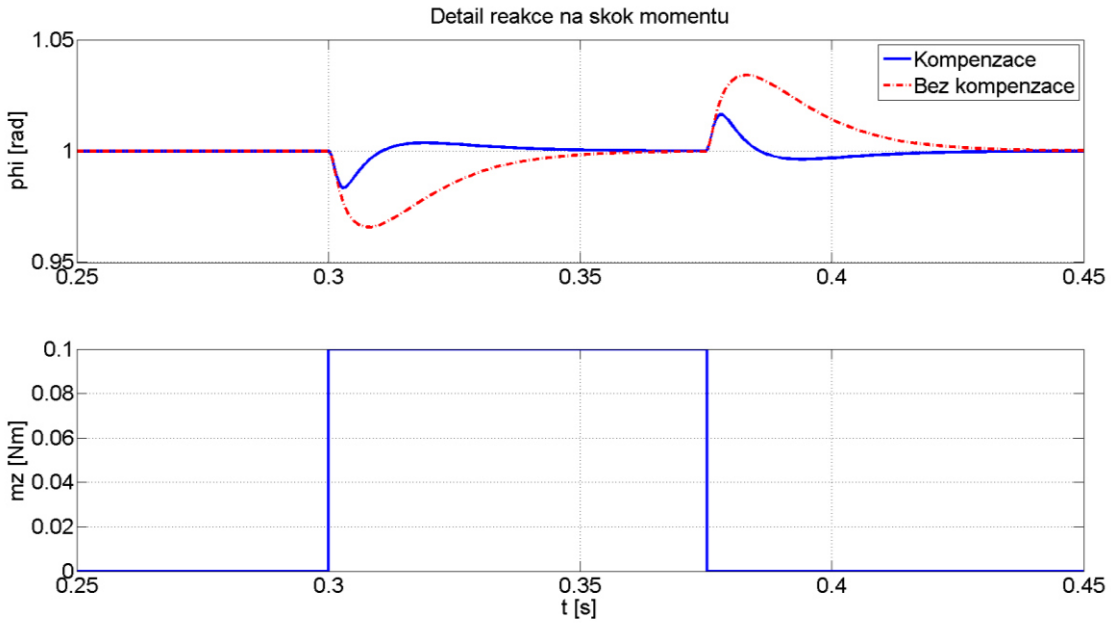


Obrázek 3.7 – Odezva na skok při periodickém zatěžování momentem



Obrázek 3.8 – Periodické zatěžování momentem a jeho odhad

Z obrázků Obrázek 3.7 a Obrázek 3.8 je patrné, že pozorovatel je dostatečně rychlý a nezhoršuje kvalitu regulace. Odhad zátěžného momentu velmi dobře aproximuje jeho reálný průběh. Amplituda cyklu zátěžného momentu je nastavena na 0,1 N·m, což podle dokumentace odpovídá přibližně jeho jmenovité hodnotě. Pro tuto simulaci byl použit řešič s variabilním krokem ODE45. Kvalita regulace polohy byla vyhodnocena kvadratickým integrálním kritériem a jeho hodnota dosáhla čísla 0,01162 (0,01242 bez kompenzace momentu, viz Obrázek 3.9 – Detail reakce při skokové změně momentu).



Obrázek 3.9 – Detail reakce při skokové změně momentu

Z výše uvedeného obrázku je zřejmé, že kompenzace poruchy výrazně zlepšuje kvalitu regulace. Maximální hodnota výkyvu se tak zmenší více než dvakrát.

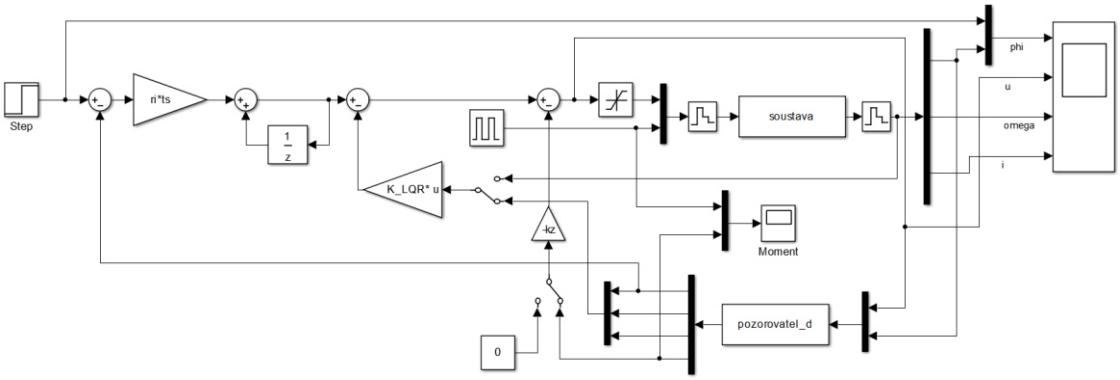
3.1.4. Převedení do diskrétní podoby a testování v uzavřené smyčce

Po úspěšném otestování ve spojitém čase spočívá další krok návrhu v převedení soustavy a regulátoru do diskrétní podoby. Jak už bylo předesláno v kapitole 2.7.2, bude použita funkce c2d s nastavením metody ZOH a periodou 100 µs (čemuž odpovídá frekvence 10 kHz). Model motoru i pozorovatele byl vytvořen jako stavový model v Matlabu, což umožňuje snadný převod následujícími příkazy.

```
%% Diskretizace

soustava_d = c2d(soustava, ts);
pozorovatel_d = c2d(pozorovatel, ts);

soustava_dt = c2d(soustava, ts, 'tustin');
pozorovatel_dt = c2d(pozorovatel, ts, 'tustin');
```

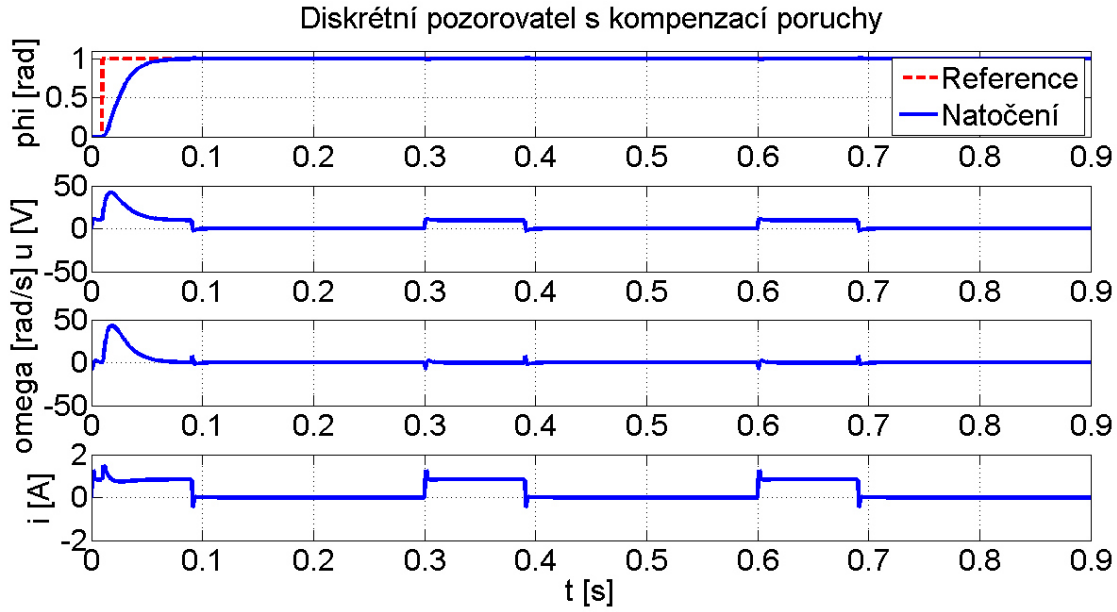
Obrázek 3.10 – Převod spojitého stavového modelu na diskrétní tvar v Matlabu

Výchozí metodou převodu je ZOH, proto u prvních dvou řádků je vynechán parametr volby algoritmu. Parametr p sloužil pro testování správnosti výsledků při různém časovém kroku simulace modelu motoru. Po sérii simulačních testů byl vybrán časový krok o velikosti 100 µs, kdy při zrychlení výpočtu se už dále kvalita regulace nezvyšovala. Model v Simulinku pro testování uzavřené smyčky v diskrétní podobě vypadá následovně.



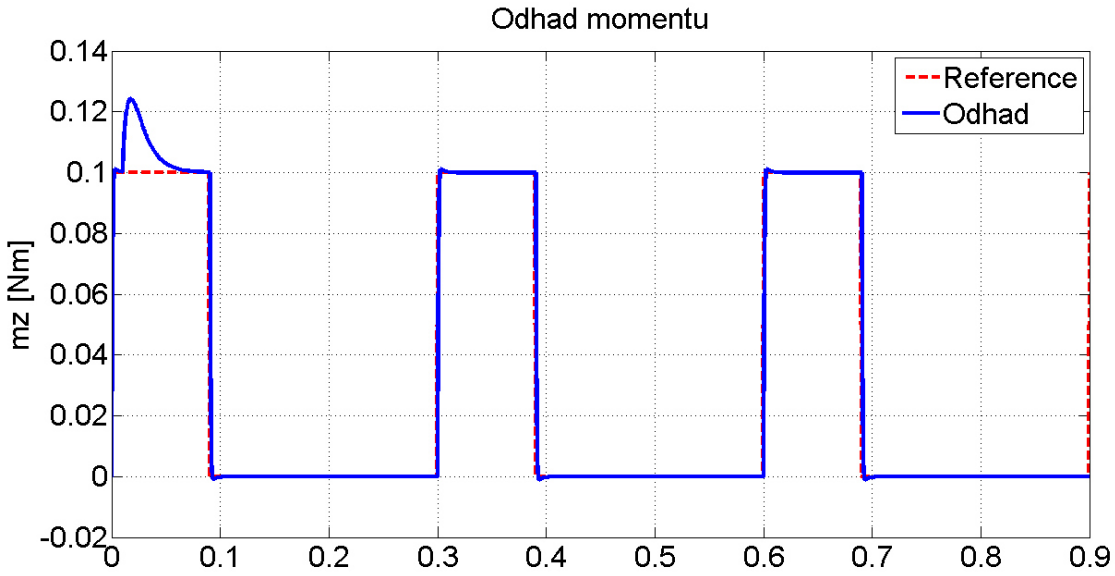
Obrázek 3.11 – Testování diskrétní uzavřené smyčky (Simulink)

Pro nahrazení integrace na vstupu byla ve smyslu ZOH použita nejjednodušší obdélníková aproximace integrace. Jinak schéma zůstává téměř totožné s obrázkem Obrázek 3.6 – Pozorovatel s kompenzací poruchy (Simulink).



Obrázek 3.12 – Odezva diskrétní uzavřené smyčky

Simulace vyobrazená na obrázcích Obrázek 3.12 a Obrázek 3.13 proběhla za stejných podmínek jako test spojitého regulátoru v předchozí kapitole. Ze simulovaných dat vyplývá, že diskretizací se chování smyčky příliš nezměnilo. Pouze odhad momentu na začátku simulace se neumí dokonale vyrovnat se skokem zatížení v čase t = 0 a skokem žádané hodnoty v čase t = 10 ms. Díky malému kroku simulace není z grafů patrný schodovitý charakter signálu (simulace proběhla s pomocí řešiče ODE45 s maximálním krokem 100 µs).

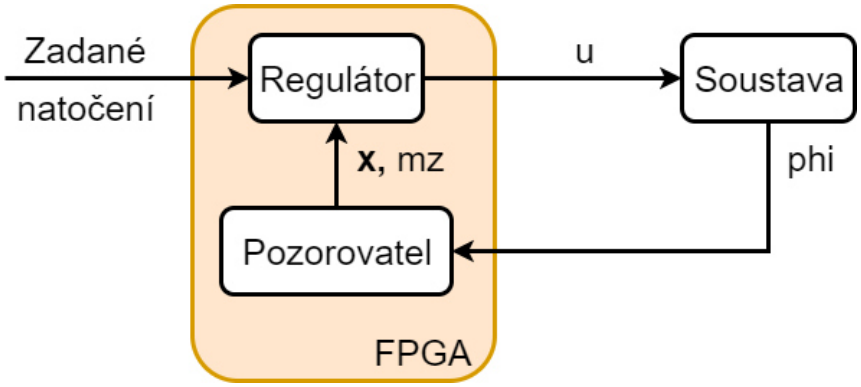


Obrázek 3.13 – Odhad momentu u diskrétní uzavřené smyčky

Tímto byla v Matlabu simulačně ověřena funkce regulátoru a pozorovatele v diskrétní podobě. U řízené soustavy nedochází k překmitu (pouze při velkém skokovém zatížení momentem) a doba ustálení na požadované hodnotě se pohybuje pod 0,1 sekundy.

3.2. Implementace regulátoru pro platformu cRIO

Cílem této kapitoly je převést celou uzavřenou smyčku z prostředí Matlab/Simulink do LabVIEW. Model motoru bude sloužit pouze pro verifikaci převodu, proto stačí využít aritmetiky s plovoucí desetinnou čárkou a implementovat soustavu na real–time procesor. Regulátor včetně pozorovatele však musí být transformován s využitím aritmetiky s pevnou desetinnou čárkou a to tak, aby bylo možné algoritmus nahrát na FPGA.



Obrázek 3.14 – Struktura funkcí vytvořená v LabVIEW

Celkem byly vytvořeny tři funkce (subVI) podle obrázku Obrázek 3.14, kde soustava může být simulována na FPGA, real-time procesoru nebo nahrazena reálným motorem.

3.2.1. Převedení modelů do prostředí LabVIEW

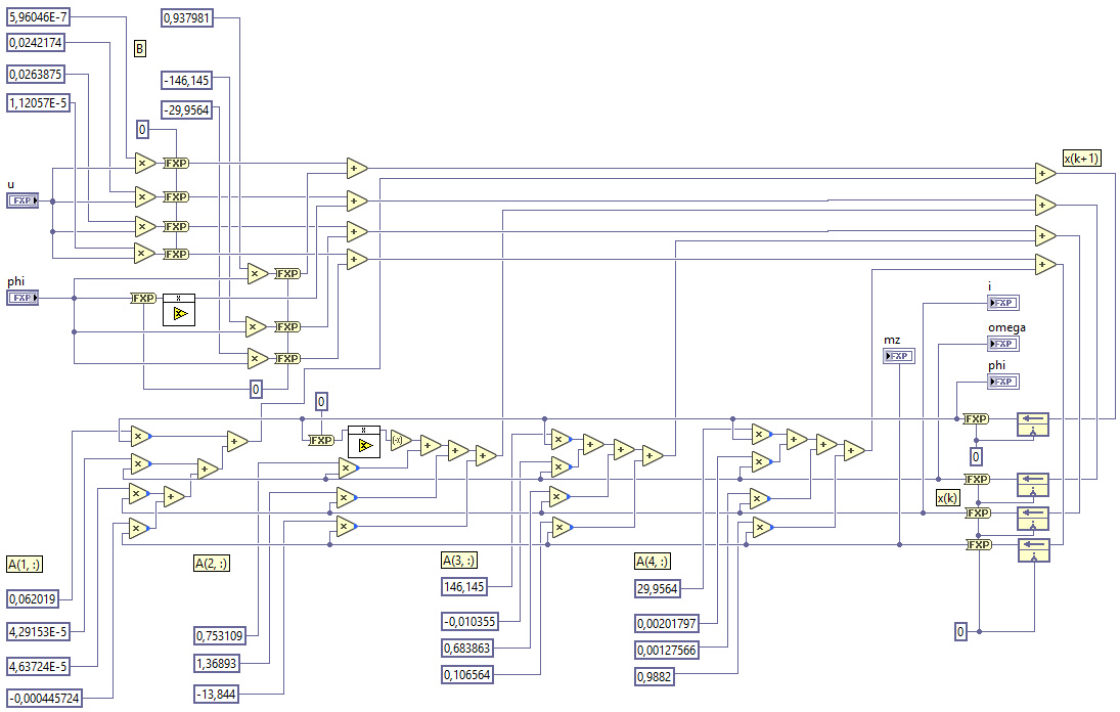
Jelikož FPGA nepodporuje maticový počet (umožněna je práce pouze s jednorozměrnými poli) je nutné stavové matice rozložit na stavové rovnice. Nahrazení maticového počtu je provedeno

tím způsobem, že každý prvek matice je násoben příslušnou stavovou proměnnou a dílčí výsledky jsou pak sečteny podle pravidel pro násobení matic a vektorů. Bohužel tímto nezbytným krokem se kód znepřehlední a v graficky programovatelném prostředí nabude na ploše. Funkce však není ovlivněna.

Z obrázku Obrázek 3.15 – Blok diagram stavového pozorovatele v LabVIEW je patrné, že při modelování jsou použity výhradně funkce dostupné na FPGA:

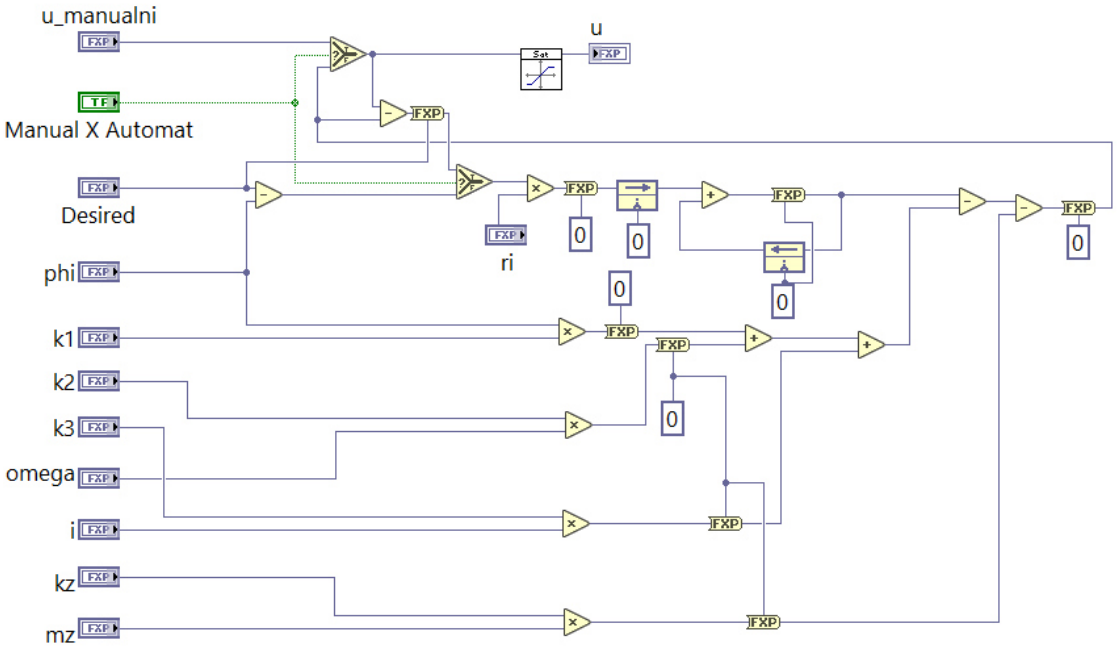
- Násobení
- Sčítání
- Uložení hodnoty z předchozí iterace (v z-transformaci odpovídá z^{-1})

Rozložení odpovídá maticovému zápisu, kdy vstupy u a phi, jsou násobeny maticí **B**, která je tvořena konstantami v horní části obrázku. Stavové proměnné jsou v prostřední části obrázku a jsou výsledkem součtu součinů **B**x**u** a **A**x**x**, kdy matice **A** je tvořena shlukem konstant ve spodní části obrázku.



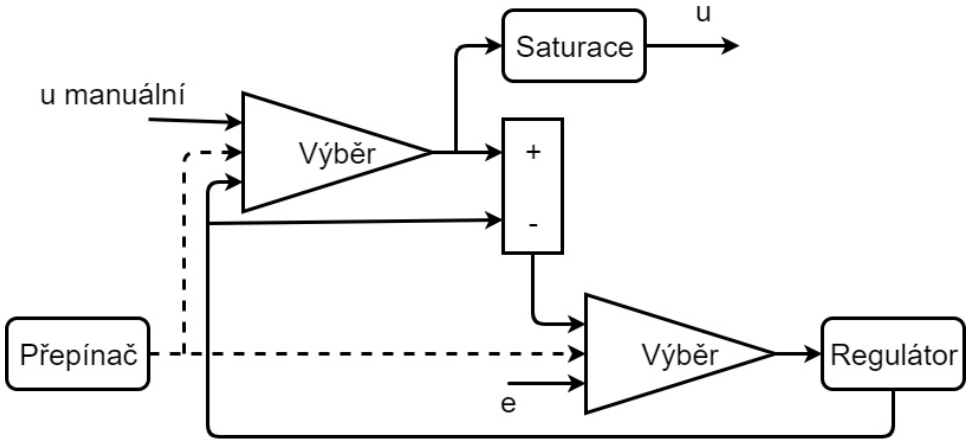
Obrázek 3.15 – Blok diagram stavového pozorovatele v LabVIEW

Struktura pozorovatele je totožná s modelem pozorovatele v Simulinku vyobrazeném na obrázku Obrázek 3.6. Místo datového typu double (plovoucí desetinná čárka, 64 bitů) je použit fixed-point (pevná desetinná čárka, proměnlivý počet bitů podle potřeby).



Obrázek 3.16 – Blok diagram regulátoru v LabVIEW

Regulátor na obrázku Obrázek 3.16 – Blok diagram regulátoru v LabVIEW pracuje na vstupu s žádanou hodnotou (signál Desired), měřeným natočením rotoru (phi), konstantami k_1, k_2 a k_3 danými návrhem zpětnovazebního stavového regulátoru (viz kapitola 3.1.2), odhady stavových veličin (výstup z pozorovatele – omega, i, m_z) a převodní konstantou k_z (převod momentu na ekvivalentní hodnotu napětí). Regulátor je navíc rozšířen o plynulé přepínání automatického módu (stavová regulace) a manuálního režimu, kdy uživatel může přímo zadat napětí na motoru. Kromě přepínače (datový typ boolean – logické hodnoty pravda a nepravda) je regulátor vytvořen z datového typu fixed-point.

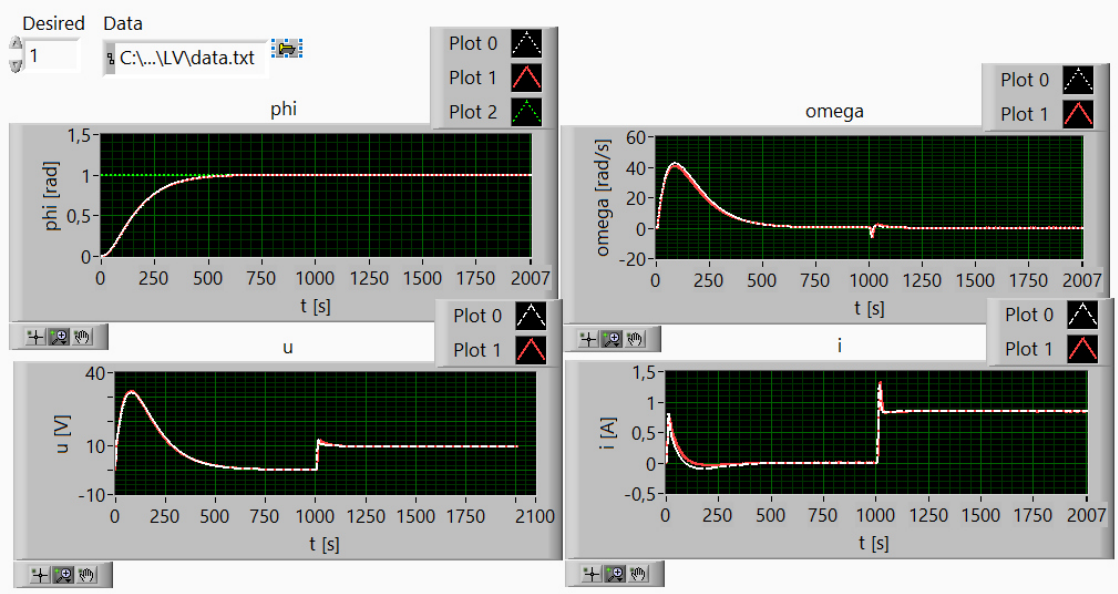


Obrázek 3.17 – Schéma přepínání automatického a manuálního módu

Stiskem tlačítka uživatel zapíná manuální režim, kdy blok přepínače generuje stav pravda. Bloky výběru pak sepnou horní přivedený signál a na výstupu regulátoru se objeví napětí zadané uživatelem, jehož hodnota je dána proměnnou u_manuální. Regulátor poté sleduje uživa-

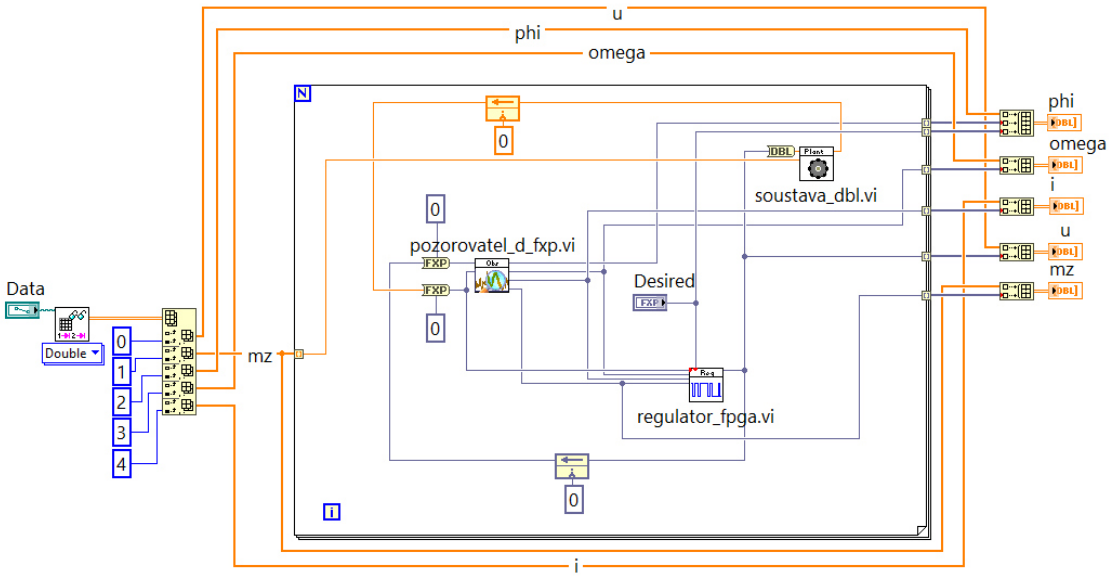
telem zapsanou hodnotu napětí a přizpůsobuje jí svůj výstup, což pak umožňuje plynulý přechod do automatického režimu. Pokud je přepínač vypnutý, uživatel zadává pouze požadovanou hodnotu natočení rotoru a regulátor pracuje ve standartním módu.

Tímto byly navrženy všechny nezbytné funkce potřebné pro vytvoření řídicího systému. Správnost převodu modelu z prostředí Matlab/Simulink do LabVIEW byla ověřena porovnáním výsledků simulací v Matlabu a LabVIEW pro zvolený stejný vstupní signál. Rozdíl mezi oběma modely je patrný z obrázku Obrázek 3.18 – Testování uzavřené smyčky v LabVIEW.



Obrázek 3.18 – Testování uzavřené smyčky v LabVIEW – front panel

Drobné rozdíly jsou dané změnou datového typu u funkce regulátor a pozorovatel, kde není možné počítat ve formátu s plovoucí desetinnou čárkou. Na následujícím obrázku je zobrazen vlastní testovací kód, kde soubor data obsahuje hodnoty vstupních i výstupních veličin získaných simulací v prostředí Matlab/Simulink.

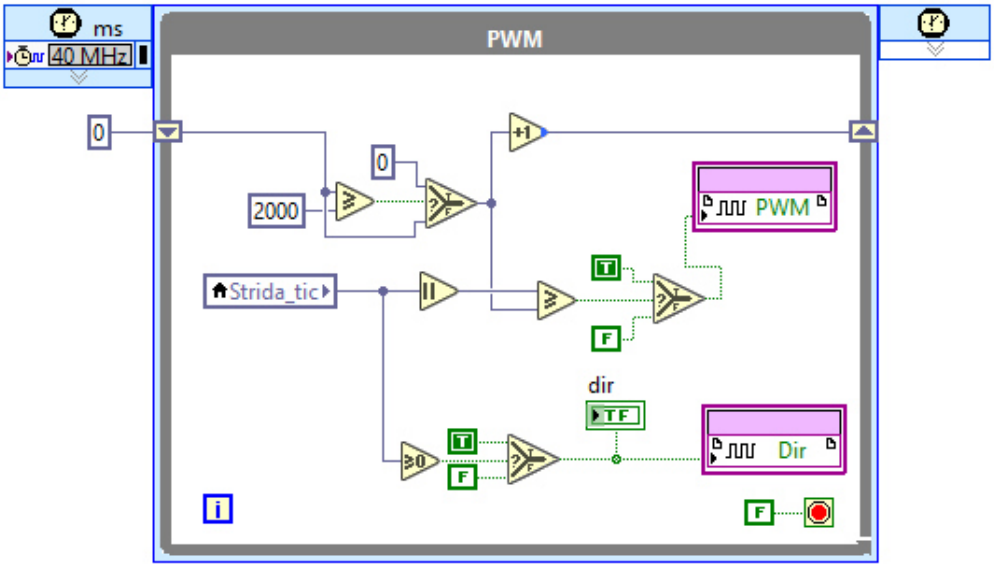


Obrázek 3.19 – Testování uzavřené smyčky v LabVIEW – block diagram

Všechny funkce byly tímto způsobem testovány i jednotlivě, aby bylo před připojením na reálnou soustavu vychytáno co nejvíce nedostatků, a nedošlo tak k poškození zařízení.

3.2.2. PWM modulace a zpracování dat z enkodéru

Dalším krokem je přizpůsobit výstupní napětí z bloku regulátoru do digitálních signálů PWM, Dir a PWM enable, které reprezentují fyzické kanály vstupující do obvodu DRV8402. Proto byl program na FPGA rozšířen o PWM modulaci, jejíž podoba v LabVIEW je zobrazena na následujícím obrázku. Signálem PWM enable je ovládáno napájení měniče, kdy logická hodnota pravda znamená vypnuto a nepravda zapnuto.



Obrázek 3.20 – PWM modulace v LabVIEW

Výstupní napětí u z regulátoru je nejprve normováno na rozsah <-1, 1>, který je pak vynásoben konstantou 2000. Toto číslo (strida_tic) reprezentuje počet kroků, pro které má hodnota stavu

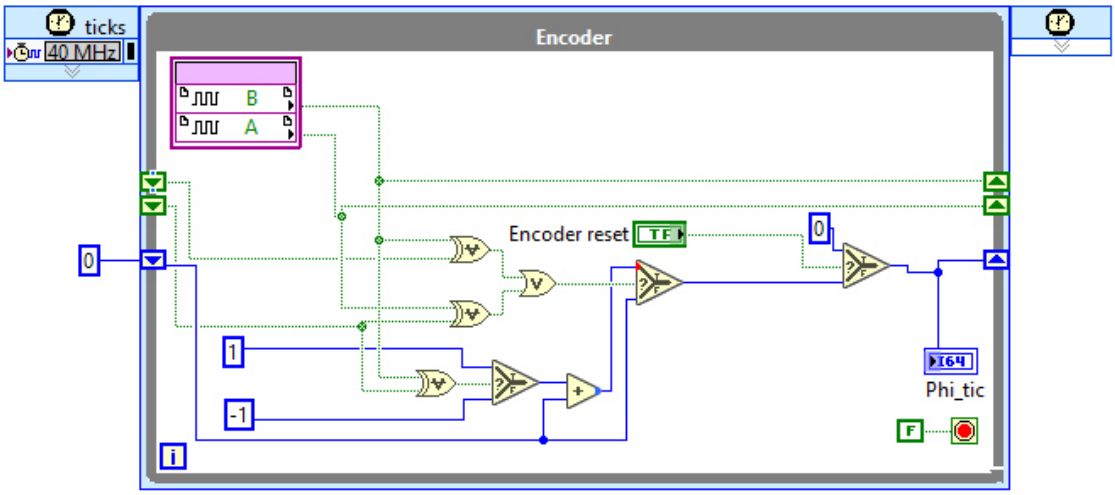
PWM nabývat logické hodnoty pravda. Ve vlastním modulátoru je pak absolutní hodnota proměnné strida_tic porovnávána s interním čítačem, který se po dosažení hodnoty 2000 resetuje na nulu. Pokud je strida_tic větší než hodnota čítače, na výstupu PWM je hodnota pravda, jinak je výstupem nepravda. Jelikož je doba kroku dána interními hodinami PWM laděnými na 40 MHz, pak jsou frekvence PWM a rozlišení PWM dány vztahy:

$$\frac{FPGA_{clock}}{2000} = f_{PWM} = 20\ kHz \tag{50}$$

$$\frac{U_{max}}{2000} = 24\ mV \tag{51}$$

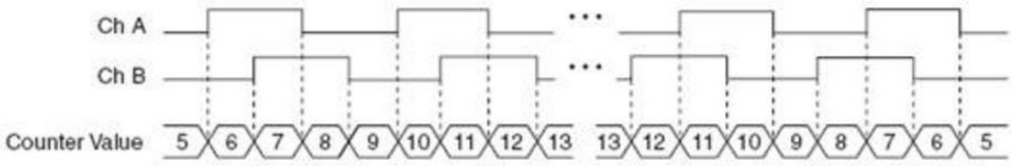
Signál Dir určuje směr otáčení a vychází ze znaménka proměnné strida_tic. Tímto jednoduchým algoritmem je zajištěno regulované napájení DC motoru.

Zpětná vazba je realizována měřením natočení rotoru pomocí kvadraturního enkodéru, jehož výstupem jsou signály A, B a Y, které je nutné převést na natočení hřídele motoru, se kterým na vstupu pracuje regulátor. Ukázka možné realizace v LabVIEW je na následujícím obrázku.



Obrázek 3.21 – Ukázka zpracování signálu z kvadraturního enkodéru

Tato část kódu má za úkol detekovat a čítat sestupné i náběžné hrany a z fázového posunu signálu A B určit smysl otáčení motoru. Proměnná phi_tic je poté převedena na radiány a slouží jako vstup do regulátoru. Logika čítání je zachycena na obrázku Obrázek 3.22.



Obrázek 3.22 – Schéma zpracování dat z kvadraturního enkodéru [22]

Rozlišení snímače v radiánech je pak dáno následujícím vztahem:

$$\theta = \frac{phi_{tic}}{x \cdot N \cdot i} 2\pi \tag{52}$$

kde x … typ enkodéru (pro kvadraturní x = 4)

N … počet pulsů na otáčku (parametr enkodéru)

i … převodový poměr převodovky

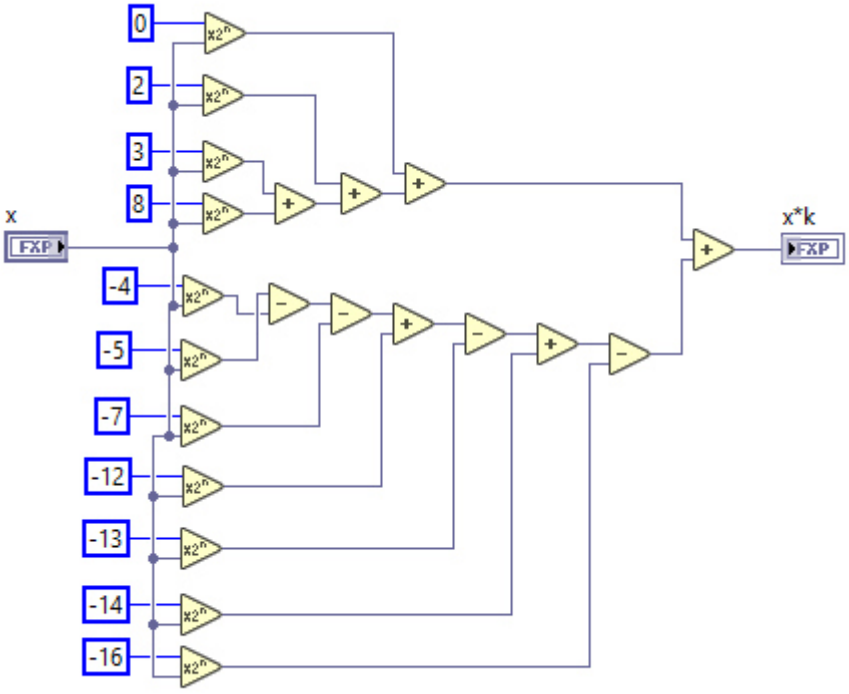
3.2.3. Implementace kódu na FPGA

Po úspěšném ověření funkčnosti regulátoru je dalším krokem nahrání algoritmu na FPGA. Pro tento účel je nezbytné program nejprve přeložit do formy bitové mapy, která v sobě obsahuje informace o propojení jednotlivých hardwarových prostředků tak, aby byla zajištěna správná funkce zařízení. Zde se bylo potřeba vypořádat s několika problémy.

- Kompilátor je oficiálně kompatibilní s operačním systémem Microsoft Windows 7 SP1 (autor pracuje ve verzi Windows 10).
- Chyba kompilátoru, který neumožňuje vypnout preferenci násobiček DSP48, což vede k chybě kompilace, kdy FPGA nemá dostatek těchto prvků pro provedení všech požadovaných násobení, i když by mohla být realizována pomocí jiných dostupných hardwarových prostředků.
- Příliš velká velikost kódu.
- Chyba pramenící z porušení časových podmínek.

Při odstraňování prvního problému se dá postupovat dvěma způsoby. První myšlenkou bylo vytvořit virtuální počítač pomocí technologie Hyper-V a nainstalovat na něj podporovaný systém Windows 7 SP1. Poté je možné kompilace provádět na tomto virtuálním počítači bez rizika neočekávaných chyb a s možností využít technické podpory firmy National Instruments, popř. Xilinx. Jako lepší řešení se ukázalo využít cloudové služby Compile Cloud Service od firmy National Instruments, kdy se kompilovaný program odešle na vysoce výkonný server firmy, kde proběhne kompilace a zpátky je poslán bitový soubor, který se už pouze nahraje do FPGA. Velkou výhodou tohoto přístupu je možnost paralelně kompilovat více programů ve stejný okamžik, dále pak mnohem rychlejší čas kompilace a šetření vlastních hardwarových prostředků na vývojovém PC. Tato služba byla využita v rámci licence na zkoušku, kterou je možno používat 90 dní (aktuální informace pro květen 2017).

U problému s nuceným používáním násobiček je jediným východiskem omezení počtu operací součinu. Toho se dá dosáhnout s využitím bitových posuvů (ekvivalentní operace jako násobení libovolnou mocninou čísla dvě) a následně jejich vzájemným součtem. Bitový posun totiž není interpretován jako násobení a přesnost součinu je daná množstvím sečtených posuvů. Tuto aproximaci součinu lze ovšem aplikovat pouze na násobení konstantou, proto je tento přístup použit pro násobení prvků stavových matic vstupním, popř. stavovým vektorem. Příklad takového násobení je zobrazen na následujícím obrázku.



Obrázek 3.23 – Ukázka násobení signálu x konstantou k (k = 3441,87186)

Po vyřešení problému s násobičkami se dalším problémem ukázala být velikost nahrávaného programu. Tento problém byl vyřešen citlivým laděním velikostí datových typů tak, aby výpočet byl stále proveden správně, ale aby každý jednotlivý signál měl minimální datovou velikost. Vycházelo se přitom z následujících poznatků:

- Maximální hodnoty napětí, otáček, proudu a momentu dané typem motoru určují velikost datového typu před desetinnou čárkou.
- Známé rozlišení enkodéru a převodu převodovky společně s rozlišením PWM modulátoru vede na stanovení optimálního počtu bitů za desetinnou čárkou.
- Velikosti prvků jednotlivých stavových matic a testování vlivu jejich zaokrouhlení na chování soustavy určuje minimální přesnost datového typu za desetinnou čárkou.

Tyto poznatky vedly k optimalizaci rozměru datových typů v celém modelu regulátoru a pozorovatele. S využitím větších datových typů by nebylo možné takto složitý regulátor na dané FPGA implementovat. I se všemi těmito úpravami se využití FPGA prostředků pohybuje za 85% celkové kapacity.

Posledním problémem, který se v souvislosti s překladem objevil, byla častá chyba související s porušením časových pravidel. Tato chyba nastávala velice nečekaně, protože hlášení kompilátoru při překladu ukazovala značné časové rezervy. Tento problém byl vyřešen nastavením preference optimalizace překladu na časování.

Device Utilization	Used	Total	Percent
Total Slices	4789	4800	99,8
Slice Registers	9562	19200	49,8
Slice LUTs	18449	19200	96,1
Block RAMs	3	32	9,4
DSP48s	32	32	100,0

Clocks	Requested (MHz)	Maximum (MHz)
MiteClk (Used by non-diagram components)	33,00	121,27
40 MHz Onboard Clock	40,00	53,82
80MHz (Used by non-diagram components)	80,00	140,94

Obrázek 3.24 – Výsledek kompilace prvního funkčního stavového regulátoru

Na obrázku Obrázek 3.24 – Výsledek kompilace je zobrazeno hlášení z kompilátoru, ve kterém jsou prezentovány informace o využití jednotlivých hardwarových prostředků a možnosti časování programu. Vidíme, že regulátor využívá významnou část dostupné kapacity FPGA čipu. Z hlášení o časování se dá usoudit, že by mohl být výpočet urychlen.

Device Utilization	Used	Total	Percent
Total Slices	4213	4800	87,8
Slice Registers	6879	19200	35,8
Slice LUTs	15165	19200	79,0
Block RAMs	3	32	9,4
DSP48s	32	32	100,0

Clocks	Requested (MHz)	Maximum (MHz)
MiteClk (Used by non-diagram components)	33,00	114,23
40 MHz Onboard Clock	40,00	46,60

Obrázek 3.25 – Hlášení kompilátoru po úspěšném překladu konečné aplikace

Výsledek kompilace finální aplikace, kde byla rozšířena komunikace s host VI (viz kapitola 3.2.4), je ukázán na obrázku Obrázek 3.25. Zvýšily se nároky na časování, ale dalšími vhodnými optimalizacemi došlo k zefektivnění využití hardwarových prostředků FPGA.

3.2.4. Vytvoření host aplikace

Mezi nezbytné součásti řídicího sytému patří uživatelské rozhraní, které uživateli umožňuje ovlivňovat chování řízené soustavy pomocí změny regulovaných veličin, sledovat a ukládat důležitá data a detekovat případné chyby a poruchy v systému. Pro tento účel byla vytvořena aplikace na real-time kontroléru.

Tato aplikace komunikuje s regulátorem na FPGA, umožňuje uživateli měnit požadovanou hodnotu natočení rotoru a sledovat a ukládat data do souboru. Informace jsou dopravovány skrz DMA FIFO frontu, kdy je nutné ošetřit fakt, že zápis ze strany FPGA probíhá řádově rychleji, a proto je nutné na straně hosta číst data po více prvcích, aby nedošlo k zahlcení fronty a ztrátě dat (host VI běží na frekvenci 1kHz a regulátor na 10kHz). Data jsou pak uložena do formátu .tdms (viz [23]). Se spuštěním host aplikace je automaticky zapnut také FPGA program. Zpracování dat pak probíhá off-line, kdy mohou být data načtena např. do Microsoft Excel nebo Matlabu.

4. Experimentování s reálným motorem

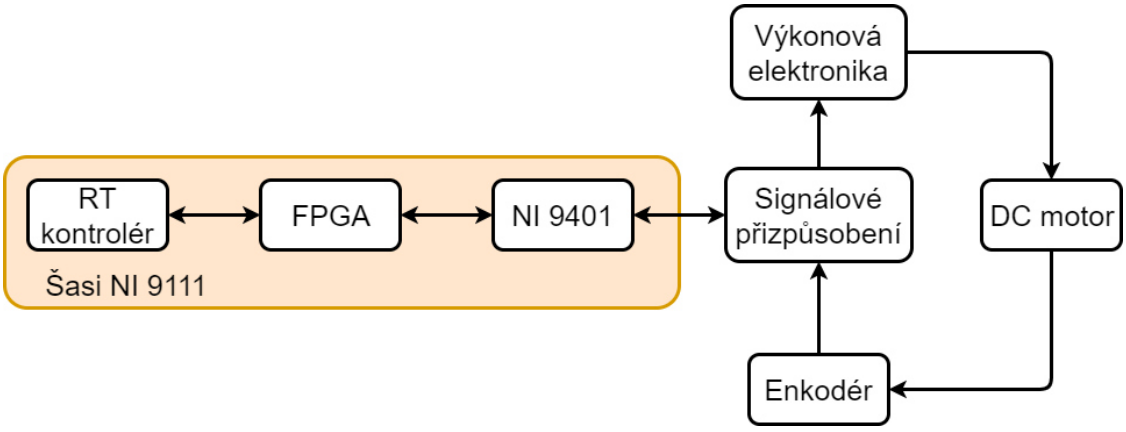
V této kapitole budou provedeny experimenty s řízením daného DC motoru regulátorem, který byl navržen na základě modelu tohoto motoru a simulačně ověřen v prostředí Matlab/Simulink.

4.1. Popis použitého hardwaru

K realizaci řídicího systému je použita platforma cRIO, která je složena ze tří částí, real-time kontroléru, šasi a C-modulů. C-moduly zajišťují propojení s okolními zařízeními (mohou být vstupní nebo výstupní, analogové, digitální, CAN apod.) a jsou ke kontroléru připojeny pomocí šasi, která se od sebe liší počtem slotů, protokolem komunikace mezi kontrolérem a C-moduly a dalšími parametry.

Obrázek 4.1 – cRIO v šasi s 8 sloty plně zaplněné C-moduly [24]

Schéma zapojení jednotlivých prvků měřicího a řídicího řetězce je zachyceno na obrázku Obrázek 4.2.

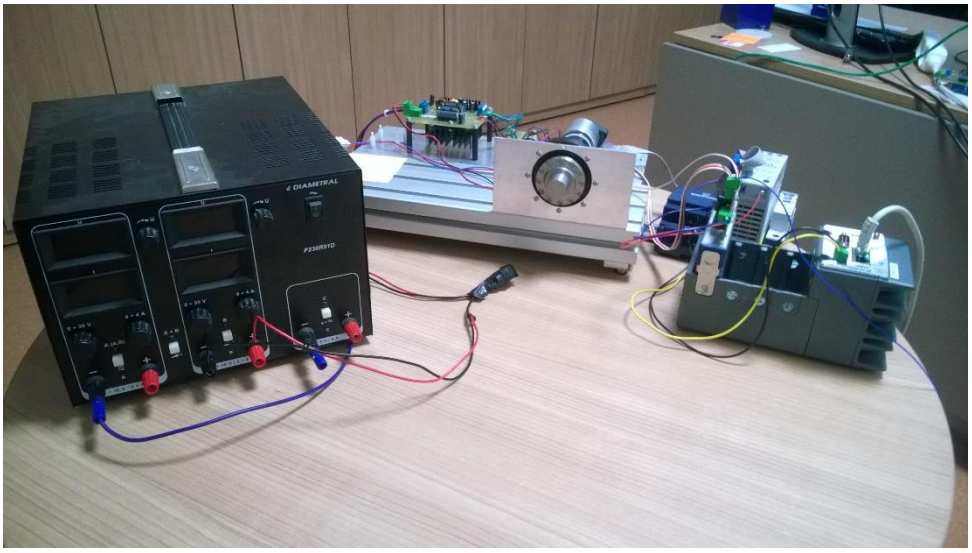


Obrázek 4.2 – Schéma zapojení

V nejvyšší vrstvě na real-time kontroléru je spuštěno tzv. host VI. Jedná se o aplikaci, která řídí chod celého systému. Uživatel zde zadává požadovanou hodnotu natočení rotoru, která je poté načtena regulátorem na FPGA, které vygeneruje řídicí signály, jež jsou poté skrz kartu NI 9401 přivedeny do silové části obvodu. Zde se měnič řízený PWM signálem stará o napájení motoru, u něhož je na hřídeli snímána pomocí enkodéru poloha rotoru. Signály generované tímto senzorem jsou pak vedeny zpět do karty NI 9401 a zpracovávány na FPGA.

Přehled jednotlivých komponent řídicího systému podle obrázku Obrázek 4.2 – Schéma zapojení je shrnut v následujícím textu:

- cRIO – 9022 – zabudovaný (embedded) kontrolér pracující v reálném čase (viz [25]) vsazený do šasi NI 9111.
    - Real–time část cRIO obsahuje host VI, které slouží k zápisu dat do souboru a nastavování požadované hodnoty.
    - Na FPGA čipu (Virtex-5 LX30 od firmy Xilinx, který je zakomponován do použitého šasi NI 9111, viz [26]) je implementován stavový regulátor s integrátorem na vstupu a kompenzací poruchy (viz kapitola 2.6.6). Vstupem do FPGA je zadaná požadovaná poloha rotoru v radiánech od uživatele, signály A, B a Y z inkrementálního enkodéru, a hodnota napětí nastavená regulátorem v předchozím kroku. Na výstupu je generován řídicí PWM signál, který určuje napětí na motoru.
    - Blok NI 9401 reprezentuje obousměrný digitální C–modul pracující s TTL logikou (5 V) se zabudovanými 10 MHz vnitřními hodinami (detaily v [27]). Tato karta je nastavena do módu, kdy porty o indexu 0 – 3 jsou výstupní a piny s indexem 4 – 7 jsou vstupní. Porty jsou řízeny přímo z FPGA.
- Součástí bloku signálové přizpůsobení je integrovaný obvod s čipem MC3486 (viz [28]), který slouží jako rozhraní mezi měřicí kartou 9401 a enkodérem. Čip MC3486 předzpracovává data z kvadraturního enkodéru a na výstupu generuje signály A, B (vzájemně posunuté o 90%) a Y (index, při otočení o 360° enkodér generuje puls).
- Hlavní součástí výkonové elektroniky je čip DRV8402 (detaily v [29]). Jedná se o řídicí silový obvod pro ovládání dvou motorů. Má v sobě zabudované dva čtyř-kvadrantové měniče a je řízen třemi signály. PWM (viz kapitola 3.2.2), Dir (určuje polaritu napětí) a PWM enable (zapínání napájení měniče).
- DC motor Maxon RE 35 (typ 273759, viz [30]), na kterém je namontována převodovka Harmonic drive AG (typ cpu-14A-100-M-SP, [31]) o převodovém poměru 100:1 a brzda Maxon AB 28 (typ 24 V, 0.4 N·m, [32]).
- Inkrementální enkodér HEDL 5540 (typ A02 1215A, [33]) s rozlišením 500 pulsů na otáčku a třemi kanály A, B a Y.
- Laboratorní zdroje Diametral P230R51D (viz [34]) pro napájení motoru a odjištění brzdy a zdroj PS5R-SC12 (detaily v [35]) pro napájení obvodu MC3486.



Obrázek 4.3 – Pracovní stůl s kompletním hardwarem

Pokud budeme popisovat obrázek Obrázek 4.3 – Pracovní stůl s kompletním hardwarem směrem zleva doprava, tak vidíme laboratorní zdroj Diametral, uprostřed vzadu přípravek s úchytem na DC motor, kde v levém zadním rohu je vidět výkonová elektronika a v pravé části je vlastní motor s převodovkou, brzdou a enkodérem připojeným na vzdálenějším konci motoru. V pravé části se nachází šasi 9111 se zapojenou digitální kartou nalevo a real-time kontrolérem napravo.

4.2. Konfigurace systému

Před implementací finálního regulátoru, který v sobě zahrnuje PWM modulaci, načítání dat z enkodéru, pozorovatele atd., byly provedeny jednodušší testy, které měly za cíl ověřit správnou funkci jednotlivých součástí konečného programu.

Po otestování správného chodu PWM modulátoru a ověření vyhodnocování natočení enkodérem bylo nutné sladit tyto prvky dohromady, aby pro dané znaménko regulační odchylky výstup PWM generoval správný směr akčního zásahu. Pro tyto účely posloužil jednoduchý PI regulátor, který byl následně nahrazen podstatně složitějším regulátorem stavovým.

Další úpravy si vyžádal fakt, že byl motor napájen ze zdroje, který generuje maximálně 30 V (jmenovitá hodnota napětí motoru je 48 V), proto tomu musel být oproti simulacím uzpůsoben výstup regulátoru. Napětí totiž není měřeno a pozorovatel jeho hodnotu čte přímo z výstupu regulátoru v rámci FPGA. Došlo by tak ke zkreslení, kdy by na motoru bylo menší napětí, než s jakým pracuje pozorovatel, čímž by byl odhad stavových veličin zkreslený a do řetězce by byla vnesena chyba.

Protože při modelování motoru byla zanedbána spousta vlivů (zejména možnosti zdroje napájení, převodovka, brzda, suché tření atd.) dá se očekávat, že parametry regulátoru optimalizované na jeho modelu bude nutné upravit. Proto byly nejprve konstanty regulátoru definovány pomocí prvků control, jejichž hodnoty lze za běhu dynamicky měnit a tím je uživateli umožněno doladit parametry regulátoru experimentálně přímo na řízené soustavě a dosáhnout tak co nejlepšího výsledku.

Parametry regulátoru získané ze simulací byly použity jako počáteční hodnoty konstant regulátoru při doladění na reálné soustavě jak ukazuje Tabulka 3.

parametr	k_1	k_2	k_3	r_i	k_z
simulace	281,46	0,58	2,99	1,55	121,72
po doladění	512,00	0,58	2,99	0,16	0,00

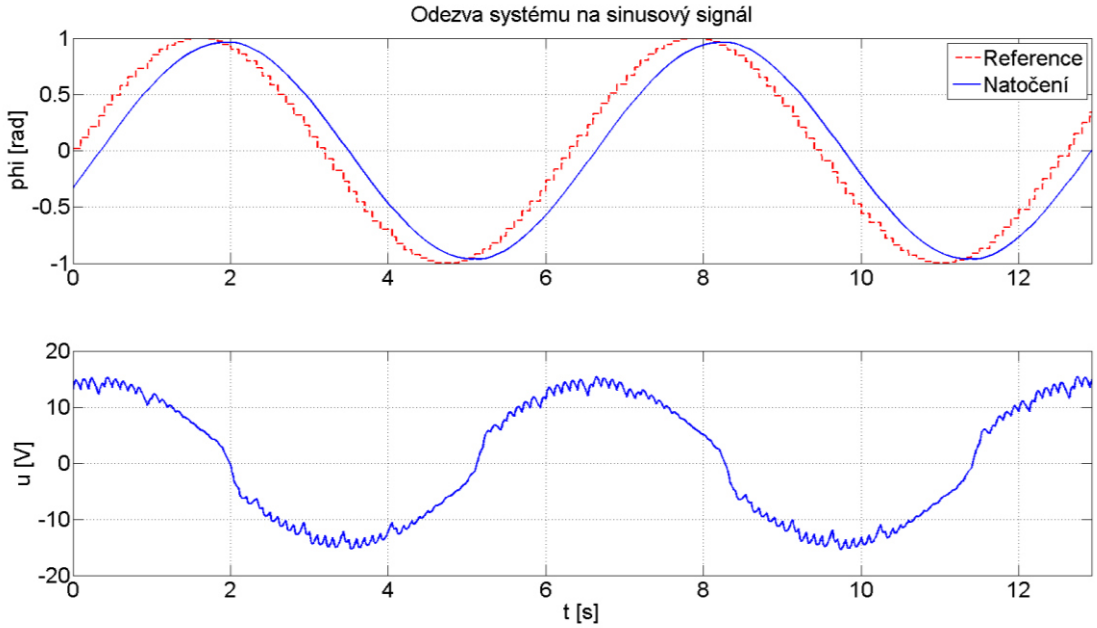
Tabulka 3 – Výsledné nastavení parametrů regulátoru

Při použití původních parametrů ze simulace soustava vykazovala výraznou kmitavou odezvu i na malou změnu požadované hodnoty. Kvůli zvýšení stability byla ve zpětnovazební regulační matici K zvýšena hodnota parametru k_1 korespondující s natočením. Dále u integrátoru na vstupu bylo zmenšeno zesílení a nakonec byla vypnuta kompenzace poruchy. Těmito úpravami se odezva systému výrazně zpomalila, ale vykazuje stabilní chování a prakticky nedochází k překmitu.

4.3. Testování řídicího systému

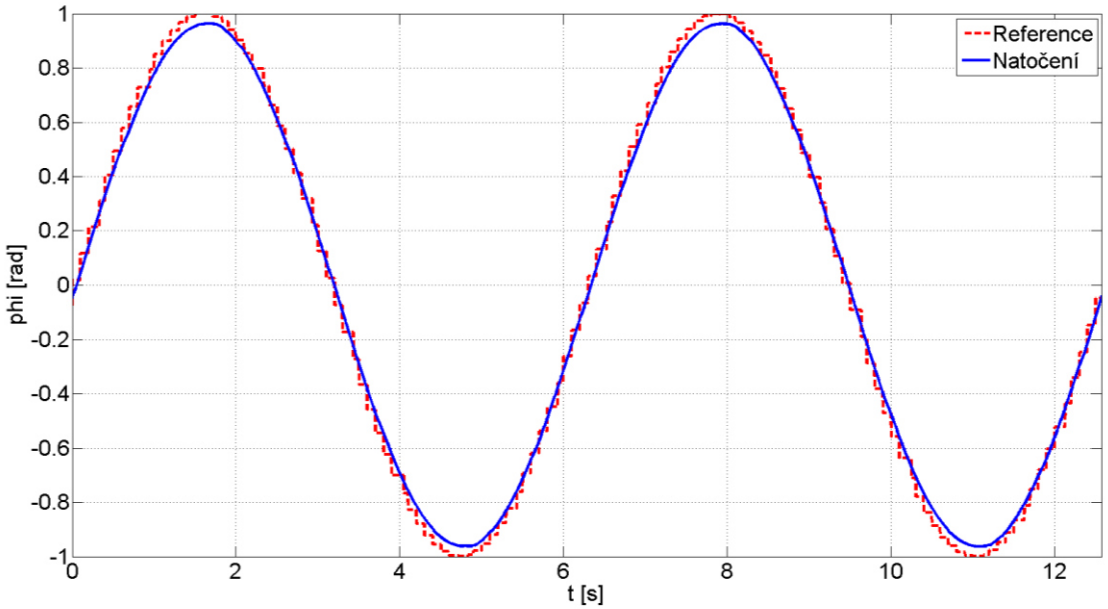
Po správné konfiguraci systému a zprovoznění regulační smyčky byly vybrány dva referenční signály (schodovitý a sinusový signál), které byly přivedeny na vstup regulátoru jako žádaná hodnota natočení. Odezva systému byla zaznamenána a je vykreslena na obrázcích Obrázek 4.4 – Odezva systému na sinusový signál a Obrázek 4.5 – Srovnání tvaru požadovaného sinusového signálu a v čase posunutých naměřených hodnot.

Pro první test byl vybrán sinusový referenční signál o amplitudě 1 radián a periodě 2π sekund. Tento experiment má za úkol otestovat schopnost regulátoru sledovat pozvolna se měnící trajektorii.



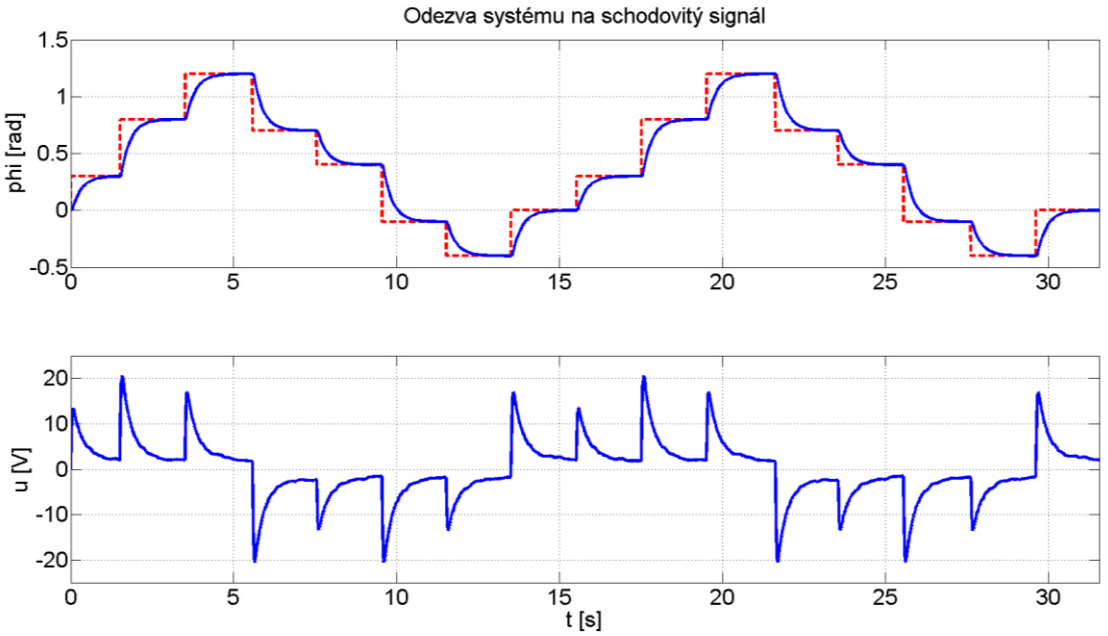
Obrázek 4.4 – Odezva systému na sinusový signál

Pro názornější srovnání byly naměřené hodnoty natočení rotoru na obrázku Obrázek 4.5 – Srovnání tvaru požadovaného sinusového signálu a v čase posunutých naměřených hodnot v čase posunuty o -0,3 sekundy, aby došlo k překrytí signálu s žádanou hodnotou.



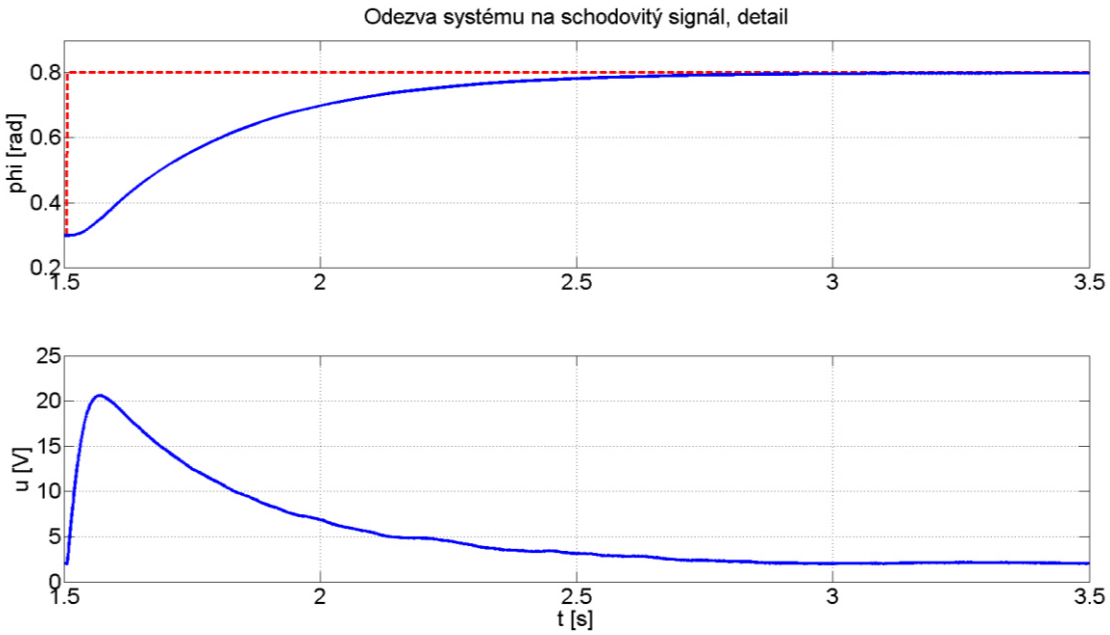
Obrázek 4.5 – Srovnání tvaru požadovaného sinusového signálu a v čase posunutých naměřených hodnot

Druhý pokus se zaměřil na schopnost soustavy reagovat na skokovou změnu žádané hodnoty, proto byl vytvořen schodovitý signál o periodě dvě sekundy se skoky 0,5 radiánu. Výsledky jsou prezentovány na obrázcích Obrázek 4.6 a Obrázek 4.7.



Obrázek 4.6 – Odezva systému na schodovitý signál

Následující obrázek představuje detail skoku v čase 1,5 sekundy o 0,5 radiánu.



Obrázek 4.7 – Odezva na skok požadované hodnoty

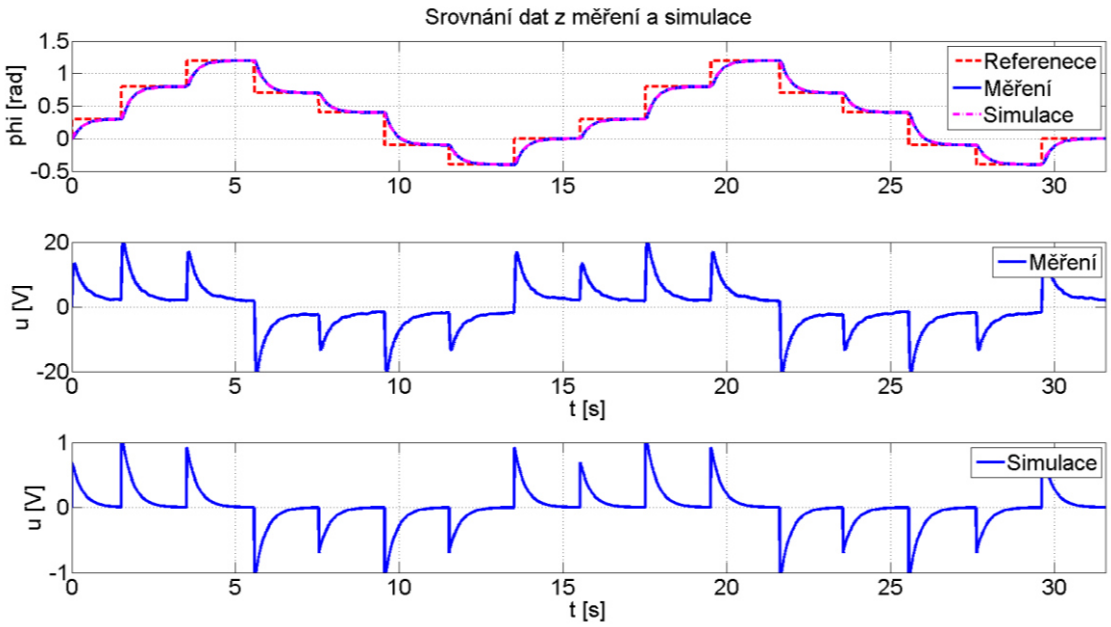
4.4. Vyhodnocení výsledků měření

Z experimentů se sinusovým signálem byly vyvozeny následující závěry. Regulátor dokáže velmi dobře sledovat měnící se trajektorii (viz obrázek Obrázek 4.5), kromě oblasti maxim funkce sinus. Vlivem zpoždění nedokázal polohu uregulovat přesně na hodnotu jednoho radiánu, což je dáno dobou trvání píku, která je větší než ustálení na požadované hodnotě plynoucí z grafu na obrázku Obrázek 4.7. Při sledování dochází k časové prodlevě, která je v daném případě přibližně 0,3 sekundy. Tato prodleva je dána rychlostí odezvy na jednotkový skok a zpožděním daným jednotlivými prvky systému (např. měnič).

Z testů se schodovitým signálem vyplývá, že doba potřebná k ustálení na nové požadované hodnotě při její prudké změně je menší než 2 sekundy a nedochází k prakticky žádnému překmitu (jednotky tiků enkodéru, což odpovídá řádově tisícinám až setinám stupně). Zde by mohla být zavedena necitlivost regulátoru (vypínání regulátoru pro velmi malé odchylky), případně by motor mohl být zabrzděn, pokud by aplikace vyžadovala pevné držení polohy.

Výše uvedené vlastnosti jsou částečně zapříčiněny faktem, že je dynamika systému negativně ovlivněna laboratorním zdrojem Diametral, který napájí DC motor a nedokáže generovat na svém výstupu jeho jmenovité napětí. Regulátor tedy nemůže nastavovat více než 30 V, i když je motor možné řídit až 48 volty.

Srovnání dat se simulacemi je poměrně problematické, poněvadž do modelu soustavy není zahrnuta převodovka, brzda a zpoždění signálů není známé. Model tedy neuvažuje žádné zatížení motoru, což vede na velmi rozdílné hodnoty akční veličiny, kdy napětí získané simulací je výrazně nižší oproti reálnému experimentu. V realitě je motor zatížen zejména harmonickou převodovkou, která společně se třením vykazuje nelineární chování [36]. Přesto se podle obrázku Obrázek 4.8 řízené natočení velmi dobře shoduje jak u měření na reálné soustavě, tak u simulace, kde byly pozměněny parametry regulátoru podle reálného systému a jako vstupní signál byla použita naměřená data.



Obrázek 4.8 – Srovnání dat získaných měřením a simulací

5. Závěr

V první části je provedena rešerše odborných článků spojených s tématem práce, jež si kladla za cíl zjistit různé možnosti postupu při návrhu a poučit se ze zkušeností získaných předchozími badateli. Z výstupů rešerše vyplývá, že práce je svým způsobem unikátní ve smyslu implementace stavového pozorovatele pro DC motor na FPGA, kdy nebyla nalezena žádná studie zabývající se tímto konkrétním problémem (použití platformy cRIO).

Dále práce pokračuje teoretickým rozborem, kdy jsou nejprve rozebrány postupy návrhu řídicího členu na základě modelu, simulace v reálném čase, popis součástí FPGA a přehled programovacích nástrojů užívaných na FPGA. Následuje kapitola pojednávající nejprve obecně o stejnosměrných motorech a poté se zaměřuje na matematický popis přechodové děje, který je základem pro vytvoření simulačního modelu DC motoru.

V další části se práce zabývá stavovým popisem soustav, rozebírá metody návrhu zpětnovazebního stavového regulátoru a pozorovatele a pokládá tím základy, ze kterých je v následujících kapitolách vytvořen konečný řídicí algoritmus. Následuje pojednání o možnostech převodu modelů do diskrétní formy a o datovém typu fixed-point, jež je nezbytný pro implementaci aplikace na FPGA.

Ve třetí kapitole je podrobně rozebrán návrh regulátoru v prostředí Matlab/Simulink, který začíná vytvořením modelu na základě odvozených přechodových rovnic DC motoru. Parametry jsou získány z technické dokumentace motoru. Tento model slouží jako základ pro vytvoření stavového pozorovatele a dále jsou na něm testovány uvažované řídicí algoritmy. Po optimalizaci kódu je regulátor s pozorovatelem převeden do diskrétní podoby a probíhá finální testování uzavřené smyčky před implementací v prostředí LabVIEW.

V následující části je navržený regulátor a pozorovatel přeprogramován do grafického prostředí LabVIEW, a to bez využití maticového počtu ve formě elementárních matematických funkcí. V modulu LabVIEW FPGA totiž nejsou dostupné funkce „vyšší“ úrovně a uživatel si proto musí vystačit se základními matematickými operacemi. Po ověření správnosti převodu porovnáním výstupů simulací v Matlabu a LabVIEW pro stejné vstupní signály, je aplikace rozšířena o zpracování signálu z enkodéru na vstupu a o PWM modulaci výstupního napětí, která je nezbytná pro řízení použité výkonové elektroniky. Nakonec je program optimalizován tak, aby mohl být implementován na konečný FPGA cílový hardware. Pro možnost sledování a ladění parametrů regulátoru byla vytvořena host aplikace, která slouží jako uživatelské rozhraní a umožňuje zápis dat do souboru pro pozdější zpracování.

Posledním krokem práce je testování řídicího systému na reálném zařízení. Nejprve se tato kapitola zabývá popisem použitého hardwaru a konfigurací systému a poté jsou provedeny praktické experimenty a jejich vyhodnocení.

Výstupem práce je stavový regulátor s integrací na vstupu a pozorovatelem poruchy implementovaný na FPGA za použití platformy cRIO. Kvalita výsledného řídicího systému může být posouzena zejména z obrázků Obrázek 4.4 a Obrázek 4.6, kde jsou zobrazeny odezvy na sinusový a schodovitý testovací signál. Pozorovatel a regulátor běží na frekvenci 10 kHz a motor je řízen pomocí PWM modulátoru s frekvencí 20 kHz. Aby bylo dosaženo stabilní regulace, musely být parametry regulátoru experimentálně doladěny, neboť při přímém použití hodnot ze simulace odezva soustavy vykazovala silný kmitavý charakter. Po úpravě parametrů regulátoru bylo dosaženo při skoku požadované veličiny (počítalo se se skokem do 1 radiánu, při větších

skocích dochází k překmitům) odezvy bez překmitu a čas ustálení se pohybuje pod dvěma sekundami.

Klíčovým prvkem práce byla optimalizace kódu, kdy z důvodu nedostatku hardwarových prostředků hrozila neúspěšná kompilace a nemožnost otestovat navržený regulátor v praxi. Precizní prací s datovými typy fixed-point a optimalizací implementace matematické operace násobení se podařilo kód s malou rezervou hardwarových prostředků nahrát a regulátor následně otestovat.

Mezi nepříjemnosti při vývoji patří dlouhá kompilace kódu a nekompatibilita kompilátoru s operačním systémem Microsoft Windows 10. Tyto problémy byly vyřešeny kompilací na serverech firmy National Instruments použitím služby NI LabVIEW FPGA Compile Cloud Service. Další nepříjemnost způsobila chyba kompilátoru, která nutí uživatele omezit četnost operace násobení, jelikož překladač umí používat pouze předpřipravené násobičky DSP48 a i když na FPGA zbývá spousta hardwarových prostředků, kompilace se nezdaří z důvodu nedostatku násobiček (viz obrázek Obrázek 3.25). Některá násobení proto musela být realizována za pomoci součtu bitových posuvů násobeného signálu.

Jako možné pokračování autor vidí implementaci měření proudu tekoucího do motoru, což by mohlo vést ke zlepšení vlastností řídicího systému, kdy by pozorovatel pracoval nejen s polohou, ale i proudem a zpřesnil by se tak odhad stavových veličin. Na druhou stranu by musela být použita analogová měřící karta, která by zvedla náklady na vývoj. Rozšířením pozorovatele by navíc hrozilo nebezpečí, že by se výsledný program nevešel na daný FPGA čip.

Dalším námětem je změna laboratorního zdroje, který nebyl schopen dodávat jmenovité napětí do motoru, což se negativně projevilo na dynamice systému a nutilo k použití konzervativnějšího nastavení regulátoru.

6. Zdroje

[1] OGATA, K.: Modern control engineering, 5. vydání, Boston: Prentice-Hall, 2010. ISBN 0-13-615673-8.

[2] ÇOLAK, İ. a kolektiv: Controller design for a limited angle torque motor and dsPIC implementation, Electrical Machines & Power Electronics (ACEMP) 2015 Intl Conference on Optimization of Electrical & Electronic Equipment (OPTIM) & 2015 Intl Symposium on Advanced Electromechanical Motion Systems (ELECTROMOTION) 2015 Intl Aegean Conference on, pp. 649-654, 2015. Dostupné z: http://ieeexplore.ieee.org.ezproxy.lib.vutbr.cz/document/6859832/citations?part=1

[3] PONCE, P. a kolektiv: Real Time Simulation for DC and AC Motors Based on LabVIEW FPGAs, [online] 2012 [citace 1. 5. 2017]. Dostupné z: http://www.sciencedirect.com.ezproxy.lib.vutbr.cz/science/article/pii/S1474667016334085#

[4] JIA, L., ZHIQIANG, W.: High Dynamic Synergetic Control Method of High Precision Position Servo System, [online] 2012 [citace 1. 5. 2017], DOI: 10.1109/ISICT.2012.6291616. Dostupné z: http://ieeexplore.ieee.org/document/6291616/

[5] ZIA, M. a kolektiv: Application of LabVIEW and cRIO for high precision positioning of Mars rover using DC motors, [online] 2011 [citace 1. 5. 2017], DOI: 10.1109/ICSpT.2011.6064681. Dostupné z: http://ieeexplore.ieee.org.ezproxy.lib.vutbr.cz/stamp/stamp.jsp?arnumber=6064681

[6] KAMENAR, E., ZELENIKA, S.: Micropositioning mechatronics system based on FPGA architecture, Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on, [online] 2013 [citace 1. 5. 2017]. Dostupné z: http://ieeexplore.ieee.org.ezproxy.lib.vutbr.cz/stamp/stamp.jsp?arnumber=6596237

[7] Model-based design, www.en.wikipedia.org, [online] [citace 1. 5. 2017]. Dostupné z: https://en.wikipedia.org/wiki/Model-based_design

[8] Studijní opory předmětu Simulace a řízení v reálném čase (RPO), studijní materiály VUT v Brně. Dostupné z e-learningu VUT.

[9] HOUŠKA, P., O. ANDRŠ a J. VETIŠKA. Mechatronic Approach to Absolute Position Sensors Design. Mechatronics, Berlin, Heidelberg: Springer Berlin Heidelberg, [online] 2012 [citace 1. 5. 2017], s. 219, DOI: 10.1007/978-3-642-23244-2_26, ISBN 978-3-642-23243-5. Dostupné z: http://link.springer.com/10.1007/978-3-642-23244-2_26

[10] FPGA Fundamentals, National Instruments, [online] 2012 [citace 1. 5. 2017]. Dostupné z: http://www.ni.com/white-paper/6983/en/

[11] Klopný obvod, www.cs.wikipedia.org [online] [citace 1. 5. 2017]. Dostupné z: https://cs.wikipedia.org/wiki/Klopn%C3%BD_obvod

[12] KOLÁČNÝ, J.: Elektrické mikropohony, skriptum VUT v Brně

[13] SKALICKÝ, J.: Elektrické regulované pohony, skriptum VUT v Brně, 2007

[14] SKALICKÝ, J.: Teorie řízení, skriptum VUT v Brně, druhé vydání, 2002

[15] Studijní opory předmětu Inteligentní řídicí systémy (RIR), studijní materiály VUT v Brně. Dostupné z elearningu VUT.

[16] Linear–quadratic regulator, www.en.wikipedia.org, [online] [citace 1. 5. 2017]. Dostupné z: https://en.wikipedia.org/wiki/Linear%E2%80%93quadratic_regulator

[17] Studijní opory předmětu Senzorika a prvky umělé inteligence (GSE), studijní materiály VUT v Brně. Dostupné z elearningu VUT.

[18] Studijní opory předmětu Základy zpracování signálu (RSZ), VUT v Brně, [online] [citace 1. 5. 2017]. Dostupné z: http://www.umt.fme.vutbr.cz/~ruja/vyuka/ZZS/zzs.html
[19] Zero-Order Hold, Mathworks, [online] 2017 [citace 1. 5. 2017]. Dostupné z: https://www.mathworks.com/help/control/ug/continuous-discrete-conversion-methods.html#bs78nig-2
[20] ŠEBEL, M.: Diskrétní modely spojitých systémů, studijní materiály k předmětu Automatické řízení na ČVUT v Praze [online] 2017 [citace 1. 5. 2017]. Dostupné z: http://www.polyx.com/_ari/slajdy/Pr-ARI-21-Sampled.pdf
[21] Studijní opory předmětu Návrh počítačových systémů (INP), studijní materiály VUT v Brně, [online] [citace 1. 5. 2017]. Dostupné z: http://pub.eyim.net/zirficka/inp/inp06fp.pdf
[22] Encoder Measurements: How-To Guide, návod National Instrumens, [online] 2017 [citace 1. 5. 2017]. Dostupné z: http://www.ni.com/tutorial/7109/en/
[23] The NI TDMS File Format, National Instruments, [online] [citace 1. 5. 2017]. Dostupné z: http://www.ni.com/white-paper/3727/en/
[24] Increase System Performance with New CompactRIO Offerings, National Instruments, [online] 2016 [citace 1. 5. 2017], Dostupné z: http://www.ni.com/newsletter/50704/en/
[25] Technická dokumentace NI cRIO – 9022, National Instruments , [online] 2014 [citace 1. 5. 2017]. Dostupné z: http://www.ni.com/datasheet/pdf/en/ds-202
[26] Technická dokumentace FPGA čipu Virtex-5, Xilinx, [online] 2016 [citace 1. 5. 2017]. Dostupné z: https://www.xilinx.com/support/documentation/data_sheets/ds202.pdf
[27] Technická dokumentace měřicí karty NI 9401, National Instruments , [online] 2015 [citace 1. 5. 2017]. Dostupné z: http://www.ni.com/pdf/manuals/374068a_02.pdf
[28] Technická dokumentace čipu MC3486, Texas Instruments, [online] 2002 [citace 1. 5. 2017]. Dostupné z: http://www.ti.com/lit/ds/symlink/mc3486.pdf
[29] Technická dokumentace obvodu DRV8402, Texas Instruments, [online] 2009 [citace 1. 5. 2017]. Dostupné z: http://www.ti.com/lit/ds/symlink/drv8402.pdf
[30] Maxon RE motory parametry, [online] [citace 1. 5. 2017]. Dostupné z: http://www.maxonmotor.com/medias/sys_master/root/8824846942238/17-DE-130.pdf
[31] Technická dokumentace převodovky Harmonic drive AG, Harmonic Drive AG, [online] 2014 [citace 1. 5. 2017]. Dostupné z: http://harmonicdrive.de/mage/media/catalog/category/2014_11_ED_1015770_CPU_M_H_S.pdf
[32] Technická dokumentace brzdy Maxon AB 28, [online] 2016 [citace 1. 5. 2017]. Dostupné z: http://www.maxonmotor.com/medias/sys_master/root/8821076623390/16-445-446-447-EN.pdf
[33] Technická dokumentace enkodéru Maxon HEDL 5540, [online] 2016 [citace 1. 5. 2017]. Dostupné z: http://www.maxonmotor.com/medias/sys_master/root/8821074427934/16-401-402-403-404-EN.pdf
[34] Technická dokumentace laboratorního zdroje P230R51D, Diametral, [online] 2017 [citace 1. 5. 2017]. Dostupné z: http://www.diametral.cz/ac-dc-zdroje/dc-regulovatelne-zdroje/laboratorni/laboratorni-zdroj-p230r51d-2x-030v/4a-1x-5v/3a.html
[35] Technická dokumentace zdroje PS5R-SC12, Idec, [online] 2016 [citace 1. 5. 2017]. Dostupné z: http://asia.idec.com/en/download/download.aspx?download=QSwitchingPSPS5RS
[36] GARAMI, B.: Dynamický model harmonické převodovky. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, [online] 2016 [citace 1. 5. 2017], 97 s.

Vedoucí diplomové práce doc. Ing. Zdeněk Hadaš, Ph.D. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=128844

7. Seznam použitých symbolů a veličin

A … matice soustavy

B … matice vstupů

b … koeficient viskózního tření [N·m·s]

B … magnetická indukce [T]

C … matice výstupů

$c\phi$ … konstanta motoru [V·s]

$c\phi_i$ … proudová konstanta motoru [N·m·A-1]

$c\phi_\omega$ … otáčková konstanta motoru [V·s]

D … matice vazby vstupů na výstup

e … exponent

F … síla [N]

F … systémová matice pozorovatele

f_s … frekvence signálu [Hz]

f_{vz} … frekvence vzorkování [Hz]

f_{PWM} … frekvence PWM modulátoru [Hz]

G … diskrétní stavová matice

H … diskrétní vstupní matice (strana 44)

H … zpětnovazební matice pozorovatele (strana 38)

i … převodový poměr [-]

i_a, I_a … proud kotvou [A]

I … proud vodičem [A]

J … moment setrvačnosti [$kg \cdot m^2$]

K … matice zpětných vazeb

l … aktivní délka vodiče [m]

L_a … indukčnost [H]

λ … kořen charakteristické rovnice

$\mathbf{M_c}$ … matice řiditelnosti

$\mathbf{M_o}$ … matice pozorovatelnosti

m … mantisa

M	…	moment [N·m]
m_z	…	zátěžný moment [N·m]
N	…	počet pulzů na otáčku u enkodéru
n	…	řád soustavy
ω	…	úhlová rychlost [rad/s]
$\mathbf{Q}$	…	matice vah stavů
$\mathbf{R}$	…	matice vah vstupů (strana 43)
$\mathbf{R}$	…	matice zpětnovazebního stavového regulátoru
R_a	…	odpor kotvy [Ω]
r_i	…	zesílení integrátoru na vstupu
S	…	soustava
S_d	…	diskrétní soustava
t_a	…	elektrická časová konstanta [s]
t_s	…	délka kroku simulace [s]
U_i	…	indukované napětí [V]
U	…	napětí [V]
u	…	napětí [V]
$\mathbf{u}$	…	vstupní vektor
$\mathbf{x}$	…	stavový vektor
$\mathbf{y}$	…	výstupní vektor
$\mathbf{z}$	…	porucha
z	…	základ soustavy

8. Seznam obrázků

Obrázek 1.1 - Schéma pohonu 15
Obrázek 2.1 - V – diagram 18
Obrázek 2.2 - Schéma MIL simulace 19
Obrázek 2.3 - Schéma HIL simulace 20
Obrázek 2.4 - Srovnání metod návrhu a testování řídicí jednotky 20
Obrázek 2.5 - Vyhledávací tabulka (LUT) pro funkci AND 22
Obrázek 2.6 - Násobení dvou 4-bitových čísel pomocí kombinačního obvodu [11] 23
Obrázek 2.7 - Srovnání běhu výpočtu na mikroprocesoru a na FPGA 24
Obrázek 2.8 - Princip vytvoření momentu u kartáčového DC motoru 26
Obrázek 2.9 - Matematický model DC motoru za pomoci přenosových funkce 29
Obrázek 2.10 - Grafické zobrazení stavových rovnic 31
Obrázek 2.11 - Grafické zobrazení soustavy se stavovým regulátorem 31
Obrázek 2.12 - Struktura s integrátorem na vstupu 32
Obrázek 2.13 - Struktura s pozorovatele 33
Obrázek 2.14 - Schéma pozorovatele se sledováním poruchy 35
Obrázek 2.15 - Ukázka ideálního bodového vzorkování v Matlabu 37
Obrázek 2.16 - Ukázka aliasingu v Matlabu 38
Obrázek 2.17 - Ukázka výsledku kvantování v Matlabu 38
Obrázek 2.18 - Schéma stavového modelu v diskrétní podobě 39
Obrázek 2.19 - Metoda ZOH (spojitá soustava je převedena do diskrétního tvaru) 40
Obrázek 2.20 - Aproximace integrace pomocí ZOH a Tustinovy diskretizační metody 40
Obrázek 2.21 - Srovnání vlivu volby metody diskretizace regulátoru na výslednou regulaci 41
Obrázek 3.1 – Odezva modelu motoru na skok napětí a momentu 44
Obrázek 3.2 – Schéma regulace s integrátorem na vstupu (Simulink) 45
Obrázek 3.3 – Ukázka návrhu regulátoru metodou LQR 45
Obrázek 3.4 – Odezva uzavřené smyčky s regulátorem naladěným pomocí LQR 46
Obrázek 3.5 – Ukázka návrhu pozorovatele 47
Obrázek 3.6 – Pozorovatel s kompenzací poruchy (Simulink) 47
Obrázek 3.7 – Odezva na skok při periodickém zatěžování momentem 48
Obrázek 3.8 – Periodické zatěžování momentem a jeho odhad 48
Obrázek 3.9 – Detail reakce při skokové změně momentu 49
Obrázek 3.10 – Převod spojitého stavového modelu na diskrétní tvar 49
Obrázek 3.11 – Testování diskrétní uzavřené smyčky (Simulink) 50
Obrázek 3.12 – Odezva diskrétní uzavřené smyčky 50
Obrázek 3.13 – Odhad momentu u diskrétní uzavřené smyčky 51
Obrázek 3.14 – Struktura funkcí vytvořená v LabVIEW 51
Obrázek 3.15 – Blok diagram stavového modelu DC motoru v LabVIEW 52
Obrázek 3.16 – Blok diagram regulátoru v LabVIEW 53
Obrázek 3.17 – Schéma přepínání automatického a manuálního módu 53
Obrázek 3.18 – Testování uzavřené smyčky v LabVIEW 54
Obrázek 3.19 – Blok diagram k obrázku 3.19 54
Obrázek 3.20 – PWM modulace v LabVIEW 55
Obrázek 3.21 – Ukázka zpracování signálu z kvadraturního enkodéru 56
Obrázek 3.22 – Schéma zpracování dat z kvadraturního enkodéru [34] 56
Obrázek 3.23 – Ukázka násobení signálu x konstantou k (k = 3441,87186) 57

Obrázek 3.24 – Výsledek kompilace prvního funkčního stavového regulátoru 58
Obrázek 3.25 – Hlášení kompilátoru po úspěšném překladu konečné aplikace 58
Obrázek 4.1 – cRIO v šasi s 8 sloty plně zaplněné C-moduly [13] 61
Obrázek 4.2 – Schéma zapojení 61
Obrázek 4.3 – Pracovní stůl s kompletním hardwarem 63
Obrázek 4.4 – Odezva systému na sinusový signál 64
Obrázek 4.5 – Srovnání tvaru požadovaného sinusového signálu a v čase posunutých naměřených hodnot 65
Obrázek 4.6 – Odezva systému na schodovitý signál 65
Obrázek 4.7 – Odezva na skok požadované hodnoty 66
Obrázek 4.8 – Srovnání dat získaných měřením a simulací 67

9. Seznam přiložených souborů

Soubory Matlab/Simulink:

- MIL_inicializace.m
- MIL_model.slx

Soubory LabVIEW:

- DP_LV.lvproj
- main_flpga.vi
- pozorovatel_FPGA.vi
- regulator_fpga.vi
- soustava_dbl.vi
- real_time_host.vi
- TDMS_read.vi
- test_casti.vi
- test_celkem.vi
- nasobeni_B22_A21.vi
- saturace.vi

Uložená data:

- data.txt – obsahuje vstupní a výstupní signály získané simulací soustavy v Matlabu a slouží pro porovnání se simulacemi v LabVIEW ppmocí VI test_casti.vi a test_celkem.vi
- schody.tdms – naměřená data na reálném DC motoru pro schodovitý vstupní signál požadované hodnoty
- sin.tdms – naměřená data na reálném DC motoru pro sinusový vstupní signál požadované hodnoty