SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

PODPORNÉ RIADIACE SYSTÉMY NA BÁZE PREDIKTÍVNEHO RIADENIA

Diplomová práca

FCHPT-5414-61591

2017 **Bc. Simon Koniar**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Podporné riadiace systémy na báze prediktívneho riadenia

Diplomová práca

FCHPT-5414-61591

Študijný program:	Automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor:	5.2.14 automatizácia
Pracovisko:	Ústav informatizácie, automatizácie a matematiky
Vedúci práce:	doc. Ing. Michal Kvasnica, PhD.

Bratislava 2017 **Bc. Simon Koniar**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie

ZADANIE DIPLOMOVEJ PRÁCE

Autor práce:	Bc. Simon Koniar
Študijný program:	automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor:	5.2.14. automatizácia
Evidenčné číslo:	FCHPT-5414-61591
ID študenta:	61591
Vedúci práce:	doc. Ing. Michal Kvasnica, PhD.
Názov práce:	**Podporné riadiace systémy na báze prediktívneho riadenia**
Jazyk, v ktorom sa práca vypracuje:	slovenský jazyk
Špecifikácia zadania:	Cieľom projektu je vyvinúť, implementovať a overiť podporné riadiace systémy na báze prediktívneho riadenia. Ich úlohou je korigovať akčné zásahy hlavného regulátora (ktorým môže byť aj ľudský operátor) tak, aby bola za všetkých okolnosti garantovaná bezpečná prevádzka riadeného procesu. Podporné systémy budú zahŕňať odhad stavových veličín pomocou techniky moving horizon estimation, algoritmy prediktívneho riadenia ako i korekčné bloky. Úlohy: 1. Návrh a implementácia odhadu stavu na báze moving horizon estimation. 2. Návrh a implementácia prediktívneho riadenia. 3. Návrh a implementácia korekčných algoritmov. 4. Zostrojenie Simulinkových blokov. 5. Simulačná a experimentálna verifikácia navrhnutých algoritmov.
Dátum zadania:	13. 02. 2017
Dátum odovzdania:	21. 05. 2017

Bc. Simon Koniar
študent

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Touto cestou by som rád vyslovil poďakovanie doc. Ing. Michalovi Kvasnicovi, PhD. za pomoc, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej diplomovej práce.

Simon Koniar

Abstrakt

Prediktívne riadenie (MPC) sa v súčasnosti s istotou radí medzi najperspektívnejšie smery v oblasti radenia komplexných procesov, ktorých podiel neustále rastie. Tento fakt spolu s ďalšími výhodami spočívajúcimi hlavne v optimálnosti riadenia, zohľadnení ohraničení ako aj flexibilnosti a ľahkej modifikovateľnosti formulácie, poukazuje na svetlú budúcnosť MPC. Už v súčasnosti má prediktívne riadenie svoje stabilné postavenie priemysle, no existujú principiálne dve hlavné prekážky, ktoré brzdia jeho prenikanie do ďalších oblastí a aplikácií. Jednak to je vysoká výpočtová náročnosť, ale taktiež pomerne zložitá implementácia, ktorú však môže výrazne uľahčiť výsledok tejto práce - knižnica univerzálnych blokov podporných riadiacich systémov na báze MPC pre Simulink. V rámci práce sme vytvorili dva jednoducho použiteľné a nastaviteľné bloky implementujúce dve rôzne úrovne MPC - "Basic" a "Advanced", ktoré je možné aplikovať pre široké spektrum modelov a procesov a to len so základnými znalosťami o MPC, čo vytvára príležitosť pre veľmi široký okruh používateľov. Funkčnosť bloku ako aj vyvinutých algoritmov je demonštrovaná na nelineárnom modeli sústavy štyroch vzájomne prepojených zásobníkov kvapalín, ktoré predstavujú MIMO systém. Následne sme si položili otázku, prečo neísť ďalej a nevyužiť potenciál a flexibilitu MPC aj v iných oblastiach. A tak vznikol algoritmus spolu s ďalším simulinkovým blokom, implementujúci pozorovač stavu na báze Moving Horizon Estimation (MHE), ktorý nie je veľmi rozšírený, no jeho kvalita je minimálne porovnateľná s inými bežne používanými pozorovačmi, čo je podložené aj konkrétnymi výsledkami v rámci práce. Samozrejme, modifikáciou na MHE sa možnosti MPC nekončia a tým ani naša práca, ktorá prezentuje taktiež nový koncept nazvaný Korektor a k nemu prislúchajúci blok. Hlavnou myšlienkou bolo vytvorenie prvku regulačného obvodu, ktorý by bol pasívny až do momentu, kedy regulátorom generovaný akčný zásah vedie na porušenie ohraničení, čomu zamedzí práve Korektor, ktorý v nevyhnutnej miere upraví akčný zásah pred vstupom do systému. Korektor je vo svojej podstate bezpečnostný prvok, ktorý pracuje optimálne a zasahuje len v prípade nebezpečenstva, takže v bežnej prevádzke iba dohliada na proces a nezasahuje. Tento princíp je názorne demonštrovaný na nelineárnom modeli kvadrokoptéry AR.Drone 2.0, ktorý predstavuje mnohorozmerný silne nelineárny MIMO systém s výraznými vnútornými interakciami a rýchlou dynamikou. Nad rámec pôvode definovaných cieľov práce bol vyvinutý blok spolu s algoritmom implementujúci Reference governor. Tento krok bol logickým pokračovaním začatej práce, keďže reference governor je takisto koncepcia vychádzajúca z MPC, ktorej sa zároveň podarilo vo výraznejšej miere preniknúť aj do priemyselných aplikácií. Pravdepodobne najvyššiu hodnotu v tejto časti práce má popri algoritme a bloku práve experimentálna verifikácia na neutralizačnom reaktore, ktorá poukazuje na fakt, že využitie vytvorených blokov zďaleka nekončí pri simuláciách s modelmi, ale je pomerne jednoduché ich implementovať aj pri riadení skutočných procesov.

Abstract

Model predictive control (MPC) is nowadays certainly one of most perspective approaches to control of complex systems and its share is steadily growing. This fact along with other main advantages like the optimality of control actions, and explicit support for constraints, and easily adjustable formulation foretell a perspective future for MPC. Predictive control already has a stable position in process industries. However, there are two main barriers which decelerate penetration into new applications. The first one is the requirements of having a sufficient computational power available. The second one is a not so straightforward implementation of MPC. Therefore this thesis introduces a Simulink library of universal blocks based on MPC. Within this project we have created two easily-implementable and tunable blocks that offers two different levels of MPC - "Basic" and "Advanced". These blocks are suitable for a wide spectrum of models and processes with only a negligible knowledge of MPC. This introduces an opportunity for very wide range of users. The functionality of developed blocks and algorithms is demonstrated on a non-linear model of four-tank system, which represents a MIMO system. Subsequently, we have asked ourselves the question why not to go further and utilize the potential and flexibility of MPC in other areas as well. Therefore we came up with an idea to develop another block with an algorithm for state estimation based on Moving Horizon Estimation (MHE). This lesser known application of MPC provides results that are at least as good, and often better, than ordinary used state observers, which is also demonstrated in the thesis. Of course, the abilities of MPC do not end up with an extension to MHE. Our work also introduces a new concept called the Corrector. The main idea was to create a control loop element that would be passive until the moment when the primary controller generates an action that leads to constraints violation. The corrector will prevent constraint violation by a suitable modification of control actions before they enter the controlled system. The corrector is basically a safety element that works optimally and only intervenes in case of danger, so in normal operation it only oversees the process and does not interfere with the primary controller. This principle is demonstrated on a non-linear AR.Drone 2.0 quadrotor model that represents a multidimensional strongly non-linear MIMO system with distinct internal interactions and rapid dynamics. Beyond the originally defined goals and objectives of the thesis, the thesis also introduces a Reference governor block. This step was a logical continuation of the work, since the reference governor is also a concept based on MPC, which has also penetrated into industrial applications. Probably the highest value in this part of the thesis, besides algorithm and block, is the experimental verification on the neutralisation reactor, which points to the fact that the use of developed blocks is not ending with a model simulations, but it is relatively easy to implement them even for the real processes control.

Obsah

Úvod

Už v bakalárskej práci sme sa venovali prediktívnemu riadeniu, ktoré je momentálne na vzostupe a razí si cestu stále k širšej oblasti aplikácií. Množstvo nesporných výhod ako garancia optimality, alebo zohľadnenie ohraničení je hybnou silou, ktorá napomáha prediktívnemu riadeniu (MPC) presadiť sa v tvrdej konkurencii s rokmi overenými spôsobmi riadenia. Na druhej strane MPC so sebou prináša aj komplikácie, respektíve obmedzenia v podobe vysokej výpočtovej náročnosti, zložitej implementácie a nevyhnutnosti poznať matematický model procesu. Avšak s rastúcou komplexnosťou procesov, tlakom na znižovanie nákladov a zvyšujúcou sa výrobnou kapacitou sa návrh riadenia značne komplikuje. Preto ak chceme využiť napríklad osvedčené PID regulátory je potrebné vynaložiť značné úsilie, no napriek tomu nemusí byť výsledok uspokojivý, keďže riadeniu chýba nadhľad nad procesom ako celkom. Práve v takýchto aplikáciách má MPC navrch, keďže dokáže proces riadiť koordinovane, so znalosťou väzieb medzi veličinami a ich predpokladaného budúceho vývoja.

S cieľom uľahčiť implementáciu a sprístupniť prácu s MPC širšiemu okruhu používateľov vnikla idea vytvorenia univerzálnych, ľahko implementovateľných blokov pre Simulink, ktorý umožňuje simulácie riadenia jednak matematických modelov, ale aj reálnych procesov. Pre takýto blok je potrebné najskôr naprogramovať univerzálny algoritmus pre výpočet optimálneho akčného zásahu, ktorý dokáže riadiť takmer ľubovoľný proces a následne vytvoriť interface pre zadávanie a spracovanie parametrov špecifikujúcich riadený proces. Následne musí byť vyvinutý blok otestovaný a musia byť vyriešené všetky zistené nedostatky a problémy. Posledným krokom bude vytvorenie dokumentácie, ktorá oboznámi používateľa s funkciami bloku a pomôže mu so zadávaním potrebných parametrov. Formulácia MPC je flexibilná a ľahko modifikovateľná nakoľko sa jedná o optimalizačný problém s ohraničeniami, takže vhodnou úpravou je možné zmeniť aj účel jej použitia. Takýmto spôsobom získame rôzne deriváty odvodené od MPC, ktoré dokážu optimálne plniť úlohy, pre ktoré boli naprogramované.

Prvým derivátom, ktorému sa budeme venovať v rámci práce je Moving horizon estimation (MHE), ktorý predstavuje pozorovač stavov slúžiaci na aproximáciu stavových veličín na základe informácií o vstupoch a výstupoch zo systému. Pozorovač stavov je zvyčajne potrebný aj pri implementácii MPC, ale aj pri iných úlohách spojených s riadením procesov a filtrovaním signálu, takže určite má význam sa zaoberať odhadom stavu. Pre MHE, rovnako ako pre MPC plánujeme vytvoriť kompaktný blok, no zároveň bude potrebné porovnať jeho výkon s inými bežne používanými pozorovačmi stavu, aby sa preukázali jeho výhody.

Ďalej sa v rámci práce plánujeme venovať principiálne novému konceptu, ktorý sme nazvali Korektor. Korektor je prvok regulačného obvodu, ktorý v prípade potreby upraví akčný zásah regulátora, ale inak nezasahuje. Potreba vzniká, keď aktuálny akčný zásah vedie na porušenie ohraničení procesných veličín, ktoré sú definované v rámci Korektora. Preto by sa dalo povedať, že Korektor predstavuje bezpečnostný prvok obvodu, ktorý je pasívny až do momentu, kedy riadenie vedie na nebezpečný stav procesu. Ďalšou prednosťou korektora je, že pracuje optimálne a akčný zásah ovplyvňuje len v nevyhnutnej miere. Pre testovanie bloku Korektora sme zvolili komplexný mnohorozmerný systém reprezentujúci v súčasnosti populárnu kvadrokoptéru, ktorej riadenie je veľkou výzvou.

Posledným produktom našej práce bude nad rámec zadania vyvinutý, no do konceptu výborne zapadajúci reference governor, ktorý predstavuje nadradený MPC regulátor určujúci referenciu pre podriadené PID regulátory. Vytvorenie bloku reference governora podnietilo hlavne jeho využívanie v priemysle a stále pretrvávajúci záujem aj z akademickej oblasti. Blok implementujúci reference governora bude otestovaný na reálnom procese - neutralizačnom reaktora, čím poukážeme na fakt, že využitie vyvinutých blokov nekončí len pri simuláciách modelov v Simulinku, ale dokážeme ich využiť aj pri riadení skutočných procesov.

Kapitola 1

Prediktívne riadenie - MPC

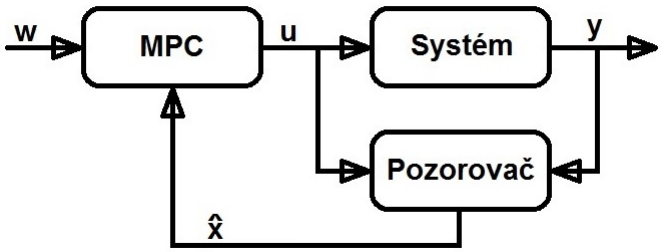
Prediktívne riadenie, respektíve MPC (z angl. *Model Predictive Control*) predstavuje pokročilú metódu riadenia procesov, ktorá zaznamenala v posledných rokoch výrazný vzostup a to hlavne z dôvodu nesporných výhod, ktoré so sebou prináša, ale taktiež kvôli zlepšeniu dostupnosti hardvéru, ktorý disponuje dostatočným výpočtovým výkonom za primeranú cenu. Ako to už v riadení býva zvykom, tak aj pri MPC musíme za lepšiu kvalitu riadenia a zohľadnenie ohraničení zaplatiť, v tomto prípade hlavne vyššou výpočtovou náročnosťou ako aj výrazne zložitejšou implementáciou. Z tohto dôvodu je preto potrebné najprv dobre zvážiť špecifiká danej aplikácie a až potom rozhodnúť o výbere vhodného spôsobu a prístupu k riadeniu toho daného procesu. Ak však uvažujeme komplexný mnohorozmerný MIMO systém s pomalou dynamikou a výraznými vzájomnými interakciami medzi procesnými veličinami, ktoré sú navyše ohraničené, tak potom je MPC takmer zaručene jasným víťazom, ktorý dokáže pri správnom nastavení dosahovať vysokú úroveň kvality riadenia.

1.1 Formulácia MPC

Prediktívne riadenie vo svojej podstate predstavuje riešenie optimalizačného problému s ohraničeniami v každej perióde vzorkovania, na celom predikčnom horizonte s cieľom získať optimálnu hodnotu riadiacej veličiny v aktuálnej perióde vzorkovania, ktorá bude aplikovaná do riadeného systému, ktorého matematický model je základom predikcie budúcich stavov. Práve vysoká flexibilita pri formulácii optimalizačného problému - účelovej funkcie a ohraničení umožňuje pomerne jednoduchú modifikáciu štandardnej formulácie MPC,

ktorá zohľadní špecifiká riadeného procesu a tým zvýši kvalitu riadenia, prípadne zníži výpočtovú náročnosť. Prediktívnym riadením, jeho modifikáciami ako aj implementáciou sme sa podrobnejšie zaoberali v rámci bakalárskej práce (Koniar, 2015), ktorá zároveň logicky a obsahovo prechádza tejto práci. Z tohto dôvodu je nevyhnutné porozumieť problematike, ktorú rieši BP, nakoľko táto práca sa venuje práve derivátom na báze MPC. Komplexný pohľad a hĺbkovú analýzu prediktívneho riadenia poskytuje napríklad Maciejowski (2002).

Prediktívne riadenie je založené na stavovej spätnej väzbe, ako to je znázornené v schéme 1.1.1, čo však môže v niektorých prípadoch skomplikovať jeho implementáciu. Problémom je, že stavové veličiny systému sa väčšinou nedajú priamo merať a preto je nevyhnutné ich aspoň aproximovať pomocou pozorovača stavov. V rámci tejto práce je riešená problematika odhadu stavu v kapitole 2, ale pri simulačnom overení funkčnosti bloku MPC v kapitole 1.3 uvažujeme situáciu, že všetky stavy systému (výšky hladín) sú merané a preto nie je potrebný pozorovač v uzavretom regulačnom obvode (URO).



Obr. 1.1.1: Schéma zapojenia MPC regulátora s pozorovačom stavu v URO.

Ako už bolo vyššie spomenuté, tak prediktívne riadenie vychádza pri predikcii budúcich stavov systému z jeho matematického modelu. V teórii riadenia poznáme rôzne modely a reprezentácie, ktoré viac, či menej spoľahlivo popisujú správanie sa reálneho procesu a tým určujú presnosť predikcie. Je zrejmé, že s kvalitným modelom (napr. nelineárnym) dosiahneme vyššiu kvalitu riadenia, no zároveň dôsledkom môže byť enormné zvýšenie výpočtovej náročnosti, keďže MPC je vo svojej podstate optimalizačný problém a jeho zložitosť súvisí okrem iného aj od typu zvoleného modelu. Štandardom pre MPC, ktorý je dodržaný aj v tejto práci, je lineárny stavový model v maticovej forme:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1.1.1a}$$

$$y(t) = Cx(t) + Du(t) \tag{1.1.1b}$$

ktorým je možné pomerne dobre aproximovať veľkú časť reálnych procesov s časovo invariantnými parametrami a približne lineárnou dynamikou v pracovnej oblasti.

Lineárny stavový model po diskretizácii zároveň poskytuje schodnú cestu pri riešení optimalizačného problému (1.1.3), do ktorého vstupuje vo forme ohraničení. Samotný model pozostáva zo štyroch matíc: A - opisuje dynamiku systému, B - vplyv riadiacich veličín (u) na stavy systému (x), C - obsahuje koeficienty lineárnej kombinácie stavov a matica D udáva priamy vplyv vstupov na výstupné veličiny (y). Všetky veličiny sú funkciami času (t), keďže sa jedná o dynamický matematický model. Člen $\dot{x}(t)$ predstavuje zmenu (deriváciu) stavových veličín v čase, čo znamená, že v ustálenom stave je nulový.

Vo väčšine reálnych procesov zároveň uvažujeme, že hodnoty stavových , výstupných a riadiacich veličín sú ohraničené:

$$x(t) \in \mathcal{X} \tag{1.1.2a}$$
$$y(t) \in \mathcal{Y} \tag{1.1.2b}$$
$$u(t) \in \mathcal{U} \tag{1.1.2c}$$

v rámci množín, respektíve intervalov $\mathcal{X}$, $\mathcal{Y}$, respektíve $\mathcal{U}$, čo dokáže MPC pomerne efektívne zohľadniť pridaním konvexných ohraničení v tvare nerovností.

V rámci tejto práce boli pre tvorbu univerzálnych blokov (kapitola 1.2) použité dve formulácie MPC. Prvou je základná formulácia:

$$\min \sum_{k=0}^{N-1} (r - y_k)^\top Q_\mathrm{x} (r - y_k) + u_k^\top Q_\mathrm{u} u_k \tag{1.1.3a}$$
$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.3b}$$
$$y_k = Cx_k + Du_k, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.3c}$$
$$x_0 = x(t) \tag{1.1.3d}$$
$$x_k \in \mathcal{X}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.3e}$$
$$u_k \in \mathcal{U}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.3f}$$
$$y_k \in \mathcal{Y}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.3g}$$

ktorá je nastavená pre riadenie výstupov na žiadanú hodnotu (r) s tvrdými ohraničeniami na procesné veličiny (x, y a u) a predikčným horizontom dlhým N períód vzorkovania, na ktorom sa predikujú budúce stavy systému x_k.

Druhá implementovaná formulácia prediktívneho riadenia:

$$\min \sum_{k=0}^{N-1} (r - y_k)^\top Q_\mathrm{x}(r - y_k) + u_k^\top Q_\mathrm{u} u_k + \Delta u_k^\top Q_{\Delta\mathrm{u}} \Delta u_k + s_k^\top Q_\mathrm{s} s_k \tag{1.1.4a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k + Ed_0, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4b}$$

$$y_k = Cx_k + Du_k + Fd_0, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4c}$$

$$x_0 = x(t) \tag{1.1.4d}$$

$$d_0 = d(t) \tag{1.1.4e}$$

$$\Delta u_k = u_k - u_{k-1}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4f}$$

$$x_{\min} - s_{\mathrm{x},k} \leq x_k \leq x_{\max} + s_{\mathrm{x},k}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4g}$$

$$y_{\min} - s_{\mathrm{y},k} \leq y_k \leq y_{\max} + s_{\mathrm{y},k}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4h}$$

$$u_k \in \mathcal{U}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4i}$$

$$\Delta u_k \in \Delta\mathcal{U}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4j}$$

$$s_k \geq 0, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{1.1.4k}$$

sa podstatne viac približuje praktickým aplikáciám, keďže umožňuje riešiť niektoré problémy s nimi spojené (Karas, 2007). Túto formuláciu, ktorá je rozšírená o *delta u* (Δu) formuláciu, modelovanie porúch, ako aj o mäkké stavové a výstupné ohraničenia využíva blok v prevedení *Advanced*. Podnetom pre vznik rozšírenej a modifikovanej formulácie MPC bol hlavne vplyv porúch a nepresnosť modelu (plant-model mismatch), ktorá je spravidla prítomná pri riadení reálnych procesov a tým negatívne ovplyvňuje kvalitu riadenia. Jedným z negatívnych prejavov nepresnosti modelu je TRO, ktorej odstránenie za pomoci modelovania porúch je demonštrované v kapitole 1.3.2. Porucha (d) je odhadovaná pomocou pozorovača stavu a je vstupom do MPC (1.1.4e), pričom jej vplyv na model systému udávajú matice E a F. Fatálne následky môže mať situácia, kedy daný optimalizačný problém nemá riešenie a tým pádom MPC nedokáže vypočítať hodnotu akčného zásahu. Neriešiteľnosť (tzv. infeasibility) je zapríčinená ohraničeniami, primárne tými na stavové a výstupné veličiny. Aby sme dokázali predísť tomuto scenáru, zavedieme mäkké ohraničenia, ktoré umožňujú čiastočnú, vysoko penalizovanú deviáciu ohraničení a to takým spôsobom, že krajné hodnoty ohraničení posunieme pomocou nezáporných doplnkových premenných (s), ktorých hodnota je zároveň penalizovaná v účelovej funkcii.

Popis jednotlivých členov a parametrov formulácie (1.1.4) je uvedený v nasledujúcej podkapitole 1.2. Dôležité je však poznamenať, že implementovaná formulácia obsahuje rôzne prepínače, ktoré aktivujú jednotlivé rozšírenia a tým pridávajú, alebo odoberajú príslušné členy z účelovej funkcie a ohraničení. Nie úplne zrejmou výnimkou je člen $u_k^\top Q_\mathrm{u} u_k$,

ktorý je v účelovej funkcii permanentne prítomný (aj po aktivácii Δu formulácie), keďže niekedy existuje požiadavka na penalizáciu absolútnej hodnoty vstupov (reprezentujúcich napr. energiu dodanú do systému). Na druhej strane, ak sa požaduje riadenie bez TRO je potrebné nastaviť hodnotu váhovej matice Q_{u} na nulu.

Obe implementované formulácie - (1.1.3) a (1.1.4) predstavujú konvexné (kvadratické) optimalizačné problémy nakoľko ich účelové funkcie sú kvadratické vzhľadom na optimalizované premenné. Ohraničenia sú výhradne lineárne, respektíve afinné, v tvare rovností ako aj neostrých nerovností. Tieto problémy sa preto dajú pretransformovať do tvaru QP:

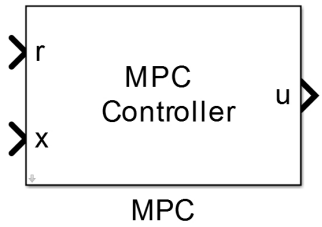
$$\min \quad {}^1\!/_2 \; z^\top P z + c^\top z \tag{1.1.5a}$$

$$\text{s.t.} \quad Gz \leq h \tag{1.1.5b}$$

kde optimalizovanou premennou je vektor z obsahujúci v prípade základnej formulácie $[u_0 \cdots u_{N-1}]$ a v rozšírenej verzii aj $[\Delta u_0 \cdots \Delta u_{N-1}]$ a doplnkové premenné $[s_0 \cdots s_{N-1}]$. Tento problém sa typicky rieši buď použitím active-set metódy, prípadne pomocou bariérových metód. Tieto metódy sú implementované vo voľne šíriteľných ako i komerčných softvérových balíkoch, ako sú napr. GUROBI, CPLEX, MOSEK, CLP, atď.

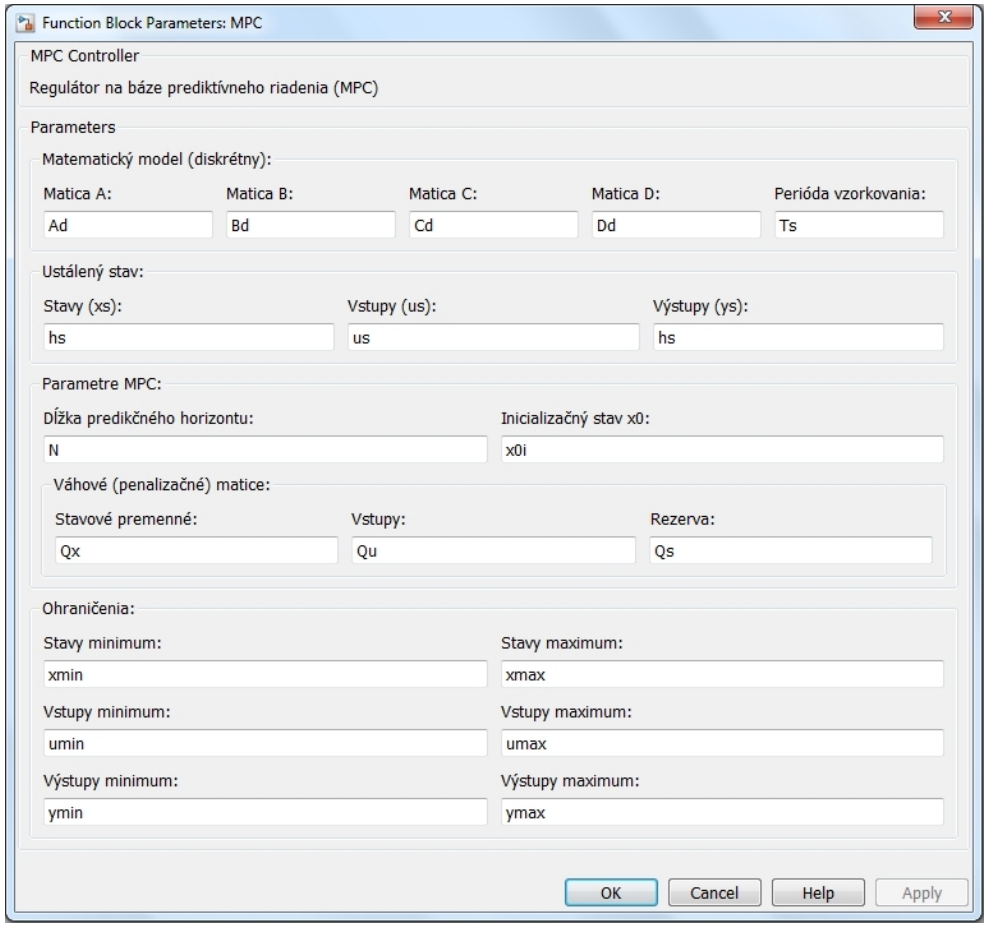
1.2 Blok MPC pre Simulink

Hlavným cieľom tejto práce v oblasti prediktívneho riadenia bolo skonštruovanie a naprogramovanie univerzálneho bloku implementujúceho algoritmy MPC v Simulinku. Nakoľko existuje mnoho formulácií a modifikácií prediktívneho riadenia, ktoré umožňujú jeho adaptáciu pre zvýšenie kvality riadenia, bol vytvorený spomínaný blok aj v prevedení *Advanced*, ktoré zahŕňa niektoré z týchto modifikácií. Samotný blok je zobrazený na obrázku 1.2.1. Z obrázka je zrejmé, že blok má dva vstupy: r - žiadanú hodnotu (referenciu) a x - aktuálny stav systému (respektíve y - aktuálny výstup, v prípade verzie *Advanced* s aktivovanou funkciou modelovania porúch). Výstupom je optimálny akčný zásah u, ktorého hodnota sa aktualizuje v každej perióde vzorkovania, inak je konštantná.

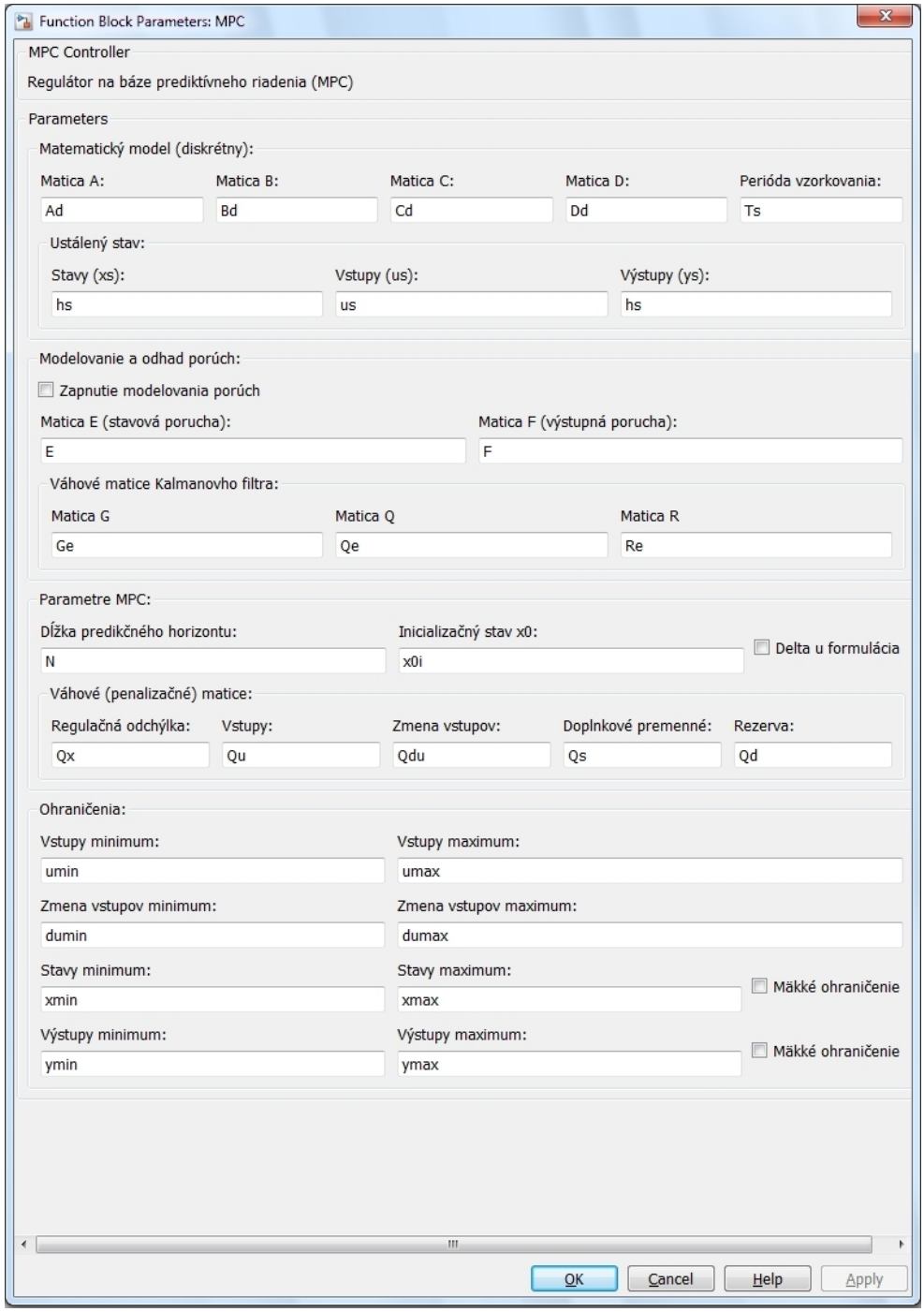


Obr. 1.2.1: Blok MPC regulátora v Simulinku.

Parametre potrebné pre správne fungovanie algoritmu sa zadávajú prostredníctvom masky bloku zobrazenej na Obr. 1.2.2, respektíve na Obr. 1.2.3 pre verziu *Advanced*. Postup tvorby masky bloku bližšie opisuje kapitola 2.3.1.



Obr. 1.2.2: Blok MPC regulátora v Simulinku.



Obr. 1.2.3: Blok MPC regulátora v Simulinku.

Popis jednotlivých parametrov ako aj princípu fungovania je súčasťou nápovedy a dokumentácie k bloku, ktorá je uvedená nižšie:

Tento blok plní funkciu diskrétneho stavového regulátora, ktorý pracuje na báze optimálneho riadenia - MPC.

Vstupom sú aktuálne hodnoty stavových (x), prípadne výstupných (y)* veličín a ich referencii - žiadaných hodnôt (r), ktoré sú v rámci bloku vstupnými parametrami optimalizačného problému. Výstupom sú hodnoty jednotlivých riadiacich veličín (u) vo forme stĺpcového vektora rovnako ako aj vstupné hodnoty.

Všetky konštantné parametre potrebné pre správne fungovanie algoritmu je možné zadať prostredníctvom masky tohto bloku (MPC Controller). Tu je krátky popis jednotlivých parametrov:

-Matematický model:

ktorý čo možno najpresnejšie opisuje vlastnosti a dynamiku riadeného (pozorovaného) systému je nevyhnutnou súčasťou MPC a jeho presnosť má zásadný vplyv na kvalitu riadenia. Očakáva sa lineárny matematický model v maticovej podobe, v diskrétnej časovej oblasti s periódou vzorkovania (Ts).

- Matica A: je matica popisujúca dynamiku pozorovaného systému. Očakáva sa štvorcová matica s rozmermi odpovedajúcimi počtu stavov - rádu systému.

- Matica B: opisuje vplyv riadiacej veličiny/ín (u) na hodnoty jednotlivých stavových veličín (x) systému. Počet riadkov musí byť zhodný s rozmerom matice A a počet stĺpcov reprezentuje počet vstupných (riadiacich) veličín. Táto matica má spolu s maticou A primárny vplyv na riaditeľnosť daného systému.

- Matica C: určuje vplyv jednotlivých stavov na hodnoty výstupných (riadených) veličín (y). Počet stĺpcov je zhodný s rozmerom matice A a počet riadkov je závislý na počte výstupných veličín.

- Matica D: udáva vplyv vstupných veličín na výstup. Väčšinou je reprezentovaná nulovou maticou s tým, že počet riadkov je zhodný s počtom výstupov a počet stĺpcov zas s počtom vstupov.

- Ustálený stav:

zodpovedá hodnotám procesných veličín v linearizačnom bode - bode, pri ktorom bol odvodený lineárny matematický model.

- Stavy (xs): hodnoty stavových veličín v ustálenom stave. Sú reprezentované stĺpcovým

vektorom s počtom prvkov rovným počtu stavov.

- Vstupy (us): hodnoty vstupných veličín v ustálenom stave. Sú reprezentované stĺpcovým vektorom s počtom prvkov rovným počtu vstupných veličín.

- Výstupy (ys): hodnoty výstupných veličín v ustálenom stave. Sú reprezentované stĺpcovým vektorom s počtom prvkov rovným počtu výstupných veličín.

- Modelovanie a odhad porúch:*

modelovanie porúch má svoje opodstatnenie vždy, keď matematický model plne nezodpovedá správaniu sa systému. Či už je rozdiel zapríčinený externou poruchovou veličinou (časovo variantná veličina vplývajúca na pozorovaný systém, ktorá nie je zadefinovaná ako vstupná veličina), alebo napríklad nelinearitou reálneho procesu, táto odchýlka sa prejaví v nepresnosti odhadu stavov. Principiálne rozlišujeme dve poruchy a to výstupnú a stavovú. Pri modelovaní môžeme uvažovať, že na systém vplýva jedna z nich prípadne aj obe súčasne, no počet porúch môže byt menší, nanajvýš rovný počtu riadených výstupov. Ich správna implementácia do modelu v spojení s vhodným nastavením Kalmanovho filtra vedie na riadenie bez trvalej regulačnej odchýlky. Modelovanie porúch je možné aktivovať zaškrtnutím možnosti "Zapnutie modelovania porúch".

- Matica E (stavová porucha): * je matica popisujúca vplyv poruchy na jednotlivé stavy systému. Určujúce je umiestnenie nenulových prvkov v matici (ktorý stav je v akom pomere ovplyvnený poruchou), veľkosť prvkov len nepriamoúmerne ovplyvní veľkosť stavovej poruchy, keďže táto je tiež len odhadovaná. Počet riadkov matice E musí byť zhodný s počtom stavových veličín a počet stĺpcov zas s počtom výstupných veličín. Ak neuvažujeme vplyv poruchy, tak zadáme nulovú maticu. Maticu E resp. F volíme tak, aby jej magnitúda bola menšia ako matice A, resp. C.

- Matica F (výstupná porucha): * má obdobnú funkciu ako matica E s rozdielom, že opisuje vplyv poruchy na výstupy zo systému a jej rozmer je rovný počtu výstupov.

-Váhové matice kalmanovho filtra: *

algoritmus MPC uvažuje konštantnú poruchu na celom predikčnom horizonte, pričom jej počiatočná hodnota je aproximovaná Kalmanovým filtrom, zrovna tak ako aj počiatočný stav systému (x_0). Z tohto dôvodu je potrebné venovať nastaveniu Kalmanovho filtra patričnú pozornosť. Matice G, Q a R predstavujú váhové matice kalmanovho filtra, ktorých bližší popis je uvedený v dokumentácii ku kalmanovmu filtru (v Matlabe príkaz `dlqe`).

-Parametre MPC:

- Dĺžka predikčného horizontu: udáva "ako ďaleko do budúcnosti sa má MPC pozerať". Zvyšovanie po určitú hranicu sa prejaví zlepšením kvality riadenia, odporúčaná hodnota je 5 -15 (s vhodne zvolenou periódou vzorkovania), keďže tento parameter má zásadný vplyv na výpočtovú náročnosť algoritmu a tým aj na čas potrebný pre riešenie optimalizačného problému.

- Inicializačný stav x0: reprezentuje aktuálny stav systému v čase keď je inicializovaný MPC regulátor. Tento parameter sa použije iba v prvej perióde vzorkovania, ďalšie hodnoty sa získavajú zo vstupu (x). Očakáva sa stĺpcový vektor s dĺžkou rovnou počtu stavov. Ak informácia o počiatočnom stave nie je k dispozícii, zadajte približné hodnoty stavov, alebo nulový vektor.

- Delta u formulácia: * aktiváciou tohto parametra je možné penalizovať a zároveň ohraničiť prírastky riadenia. Výsledkom môže byť menej kmitavý regulačný pochod (hlavne priebeh riadiacej veličiny), prípadne zohľadnenie dynamiky akčného člena.

- Váhové (penalizačné) matice:

vzhľadom na formuláciu MPC definujú požiadavky na kvalitu riadenia, určujú priority a rozlišujú hodnoty, respektíve ceny jednotlivých veličín. Je potrebné myslieť na to, že hodnoty jednotlivých váhových matíc majú hlavne relatívny význam vzhľadom na ostatné váhové matice. Umiestnenie jednotlivých nenulových prvkov vo váhovej matici a ich hodnoty určujú ktorý zo stavov/výstupov/porúch má byť prioritne penalizovaný. Je vhodné použitie diagonálnych matíc, prípadne skalárov. Ak sa nevyžaduje rozlišovať medzi jednotlivými stavmi/výstupmi/poruchami je možné použiť jednotkové matice s patričnými rozmermi.

- Regulačná odchýlka: nakoľko blok uvažuje s riadením výstupov na žiadanú (úloha sledovania), primárnym cieľom riadenia je znižovanie regulačnej odchýlky ($r - y$), ktorej kvadrát zároveň penalizuje táto váhová matica. Očakáva sa, že bude zadaná štvorcová matica s rozmerom rovným počtu výstupov, prípadne skalár. Ak je rozdiel medzi hodnotou výstupnej veličiny zo systému a referenciou, tak je potrebné zvýšiť hodnotu tejto váhovej matice.

- Vstupy: váhová matica penalizujúca kvadrát vstupných - riadiacich veličín, ktoré v praxi často krát predstavujú náklady (rôzne formy energie vložené do systému), preto je potrebné vhodne zvoliť jej hodnotu. Jej zvýšením dochádza k zníženiu akčných zásahov a tým k úspore energie, no zároveň k zhoršeniu kvality riadenia. V prípade diagonálnej matice, prvky na hlavnej diagonále udávajú ceny, respektíve pomer cien jednotlivých vstupov, preto je možné pomerne efektívne rozhodnúť, ktorá vstupná veličina sa má v

akej miere aplikovať do systému a tým šetriť energiu, v prípade, že je k dispozícii viacero vstupov s podobným vplyvom na riadený systém. Očakáva sa štvorcová matica s rozmerom odpovedajúcim počtu vstupov.

- Zmena vstupov: * pri použití delta u (Δu) formulácie sa penalizujú prírastky (zmena) riadenia, čiže rozdiel medzi predchádzajúcim a aktuálnym akčným zásahom. Táto váhová matica má vplyv na plynulosť priebehu akčného zásahu v čase, keďže penalizuje kvadrát Δu.

- Doplnkové premenné: * penalizuje kvadrát doplnkových premenných (tzv. slack-ov), ktoré sú použité pri posunutí krajných hodnôt ohraničení a tým transformujú dané ohraničenia na mäkké. Očakáva sa štvorcová matica s rozmerom odpovedajúcim počtu stavov a výstupov v súčte, prípadne skalár s dostatočne vysokou hodnotou.

- Rezerva: nakoľko blok umožňuje aj modifikáciu samotného algoritmu, bol v maske vytvorený priestor pre "rezervnú" váhovú maticu, ktorú je možné použiť pri úprave algoritmu - pridaní nového členu do účelovej funkcie.

- Ohraničenia:

sú hlavnou prednosťou prediktívneho riadenia. MPC regulátor generuje také akčné zásahy, ktoré rešpektujú dané ohraničenia a hodnota procesných veličín bude v rámci zadaného intervalu za predpokladu, že matematický model je dostatočne presný a predikčný horizont má postačujúcu dĺžku. Pre jednotlivé procesné veličiny je možné zadať minimálne a maximálne hodnoty, ktoré môžu nadobúdať. V prípade, že je určitých veličín viac, je potrebné zadať limitné hodnoty vo forme stĺpcového vektora, pričom poradie jednotlivých prvkov zodpovedá poradiu v matematickom modeli. Pre stavové ako aj výstupné veličiny je možné povoliť miernu deviáciu ohraničení aktiváciou mäkkých ohraničení*, čo predchádza neriešiteľnosti optimalizačného problému.

* - dostupné len vo verzii "Advanced".

Zdrojový kód samotnej funkcie - algoritmu MPC spolu s blokom je dostupný na internete a voľne stiahnuteľný [1].

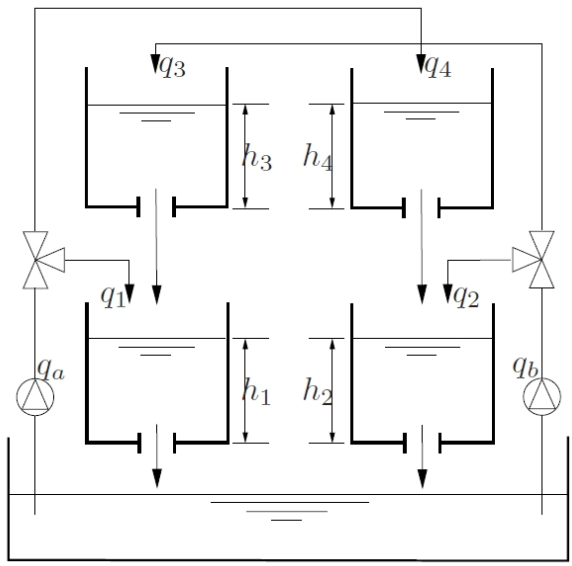
[1] https://Simon_Koniar@bitbucket.org/Simon_Koniar/podporne-riadiace-systemy-na-baze-mpc

1.3 Simulačné overenie

Vytovrený blok MPC regulátora bol testovaný v prostredí, pre ktoré bol vytvorený - v Simulink-u. Pre účel simulácie riadenia bol použitý nelineárny matematický model štyroch vzájomne prepojených zásobníkov kvapalín, ktorých schematický nákres je na Obr. 1.3.1. Simulácia bola realizovaná v spojitom čase nakoľko riadený systém - sústava zásobníkov je spojito pracujúce zariadenie, ale regulátor generoval diskrétne akčné zásahy s periódou vzorkovania $Ts = 1s$.

1.3.1 Simulačný model

Simulačný model, ako už bolo vyššie spomenuté, bol reprezentovaný sústavou štyroch zásobníkov kvapalín bez interakcie vyobrazených v schéme 1.3.1 (Holaza, 2015). Zo schémy vyplýva, že do sústavy vstupujú dva prúdy kvapaliny cez čerpadlá q_a a q_b, ktoré sa ďalej delia pomocou trojcestných ventilov v určitom, fixnom pomere.



Obr. 1.3.1: Schéma zapojenia zásobníkov.

Parameter	**Hodnota**	**Jednotka**	**Popis**
$F_{1,2,3,4}$	0.15	m^2	Plocha podstavy
k_1	8.7932×10^{-3}	m^2	Prierez výtok. štrbiny
k_2	7.3772×10^{-3}	m^2	Prierez výtok. štrbiny
k_3	6.3495×10^{-3}	m^2	Prierez výtok. štrbiny
k_4	4.3567×10^{-3}	m^2	Prierez výtok. štrbiny
γ_a	0.3	–	Deliaci pomer A
γ_b	0.4	–	Deliaci pomer B
g	9.81	$m.s^{-2}$	Gravitačné zrýchlenie
q_a^s	1.5×10^{-3}	$m^3.s^{-1}$	Prietok A
q_b^s	1.5×10^{-3}	$m^3.s^{-1}$	Prietok B
h_1^s	0.1202	m	Výška hladiny
h_2^s	0.2550	m	Výška hladiny
h_3^s	0.1024	m	Výška hladiny
h_4^s	0.2961	m	Výška hladiny

Tabuľka 1.1: Parametre modelu

Dynamický matematický model daného procesu možno popísať štyrmi diferenciálnymi rovnicami, keďže sa jedná o systém štvrtého rádu (každý zo zásobníkov je možné aproximovať systémom prvého rádu). Vychádzajúc z materiálových bilancií tohto technologického zariadenia získame nasledovný DMM:

$$q_a(t)\gamma_a + k_3\sqrt{2gh_3(t)} = k_1\sqrt{2gh_1(t)} + F_1\frac{dh_1(t)}{dt} \tag{1.3.1a}$$

$$q_b(t)\gamma_b + k_4\sqrt{2gh_4(t)} = k_2\sqrt{2gh_2(t)} + F_2\frac{dh_2(t)}{dt} \tag{1.3.1b}$$

$$q_b(t)(1-\gamma_b) = k_3\sqrt{2gh_3(t)} + F_3\frac{dh_3(t)}{dt} \tag{1.3.1c}$$

$$q_a(t)(1-\gamma_a) = k_4\sqrt{2gh_4(t)} + F_4\frac{dh_4(t)}{dt} \tag{1.3.1d}$$

$$h_1(0) = h_1^s; h_2(0) = h_2^s; h_3(0) = h_3^s; h_4(0) = h_4^s \tag{1.3.1e}$$

Pri zachovaní konštantných vstupných prietokov sa za určitý čas výšky hladín v jednotlivých nádržiach ustália a tým nedochádza k akumulácii kvapaliny v systéme. Tento

stav môžeme opísať matematickým modelom ustáleného stavu (MMUS):

$$\frac{q_a^s \gamma_a}{F_1} + \frac{k_3}{F_1}\sqrt{2gh_3^s} - \frac{k_1}{F_1}\sqrt{2gh_1^s} = 0 \tag{1.3.2a}$$

$$\frac{q_b^s \gamma_b}{F_2} + \frac{k_4}{F_2}\sqrt{2gh_4^s} - \frac{k_2}{F_2}\sqrt{2gh_2^s} = 0 \tag{1.3.2b}$$

$$\frac{q_b^s (1-\gamma_b)}{F_3} - \frac{k_3}{F_3}\sqrt{2gh_3^s} = 0 \tag{1.3.2c}$$

$$\frac{q_a^s (1-\gamma_a)}{F_4} - \frac{k_4}{F_4}\sqrt{2gh_4^s} = 0 \tag{1.3.2d}$$

Lineárny matematický model získame odčítaním MMUS od DMM, no zároveň je potrebné zadefinovať nové odchýlkové veličiny:

$$x_1(t) = h_1(t) - h_1^s \tag{1.3.3a}$$

$$x_2(t) = h_2(t) - h_2^s \tag{1.3.3b}$$

$$x_3(t) = h_3(t) - h_3^s \tag{1.3.3c}$$

$$x_4(t) = h_4(t) - h_4^s \tag{1.3.3d}$$

$$u_1(t) = u_1(t) - u_1^s \tag{1.3.3e}$$

$$u_2(t) = u_2(t) - u_2^s \tag{1.3.3f}$$

a taktiež nelineárne členy nahradiť aproximáciou taylorovým rozvojom do prvého rádu:

$$k_i\sqrt{2gh_i(t)} = k_i\sqrt{2gh_i^s} + k_i\frac{2g}{2\sqrt{2gh_i^s}}(h_i(t) - h_i^s) \tag{1.3.4}$$

$$k_i\sqrt{2gh_i(t)} - k_i\sqrt{2gh_i^s} = k_i\sqrt{\frac{g}{2h_i^s}}x_i \tag{1.3.5}$$

Po dosadení získavame linearizovaný matematický model v tvare:

$$F_1\frac{dx_1(t)}{dt} = u_1(t)\gamma_a + k_3\sqrt{\frac{g}{2h_3^s}}x_3 - k_1\sqrt{\frac{g}{2h_1^s}}x_1 \tag{1.3.6a}$$

$$F_2\frac{dx_2(t)}{dt} = u_2(t)\gamma_b + k_4\sqrt{\frac{g}{2h_4^s}}x_4 - k_2\sqrt{\frac{g}{2h_2^s}}x_2 \tag{1.3.6b}$$

$$F_3\frac{dx_3(t)}{dt} = u_2(t)\gamma_b - k_3\sqrt{\frac{g}{2h_3^s}}x_3 \tag{1.3.6c}$$

$$F_4\frac{dx_4(t)}{dt} = u_1(t)\gamma_a - k_4\sqrt{\frac{g}{2h_4^s}}x_4 \tag{1.3.6d}$$

$$x_1(0) = 0; x_2(0) = 0; x_3(0) = 0; x_4(0) = 0 \tag{1.3.6e}$$

Linearizovaný matematický model je možné teraz pomerne jednoducho previesť do

maticovej formy, kedy má nasledovný tvar:

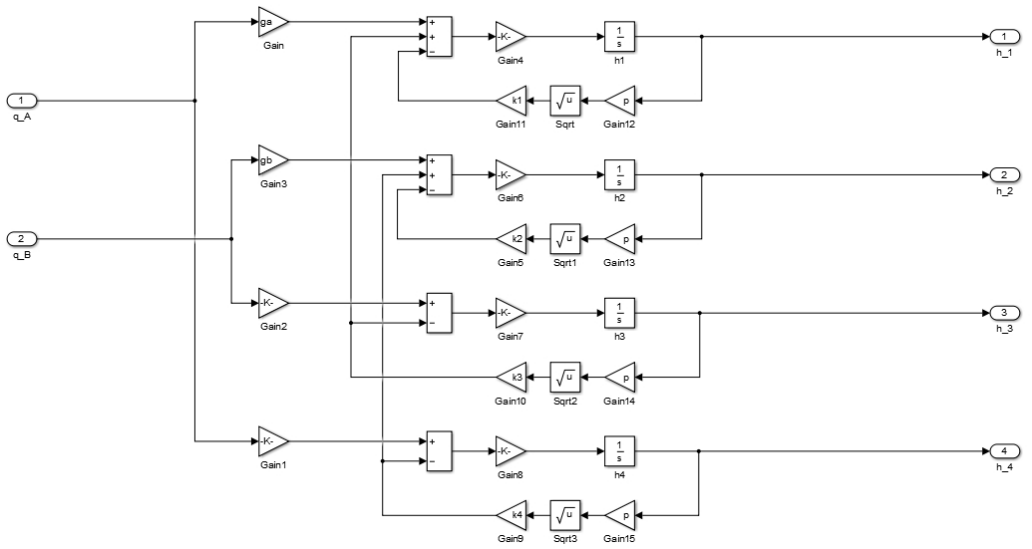
$$A = \begin{bmatrix} -\frac{k_1}{F_1}\sqrt{\frac{g}{2h_1^s}} & 0 & \frac{k_3}{F_1}\sqrt{\frac{g}{2h_3^s}} & 0 \\ 0 & -\frac{k_2}{F_2}\sqrt{\frac{g}{2h_2^s}} & 0 & \frac{k_4}{F_2}\sqrt{\frac{g}{2h_4^s}} \\ 0 & 0 & -\frac{k_3}{F_3}\sqrt{\frac{g}{2h_3^s}} & 0 \\ 0 & 0 & 0 & -\frac{k_4}{F_4}\sqrt{\frac{g}{2h_4^s}} \end{bmatrix} \tag{1.3.7}$$

$$B = \begin{bmatrix} \frac{\gamma_a}{F_1} & 0 \\ 0 & \frac{\gamma_b}{F_2} \\ 0 & \frac{1-\gamma_b}{F_3} \\ \frac{1-\gamma_a}{F_1} & 0 \end{bmatrix} \tag{1.3.8}$$

Matica C bola zadefinovaná ako diagonálna jednotková matica s rozmermi 4×4 a matica D je nulová matica s rozmerom 4×2.

Tento lineárny stavový model bol použitý pre návrh MPC, no ešte predtým bol prevedený do diskrétnej časovej oblasti, nakoľko prediktívne riadenie pracuje v diskrétnom čase.

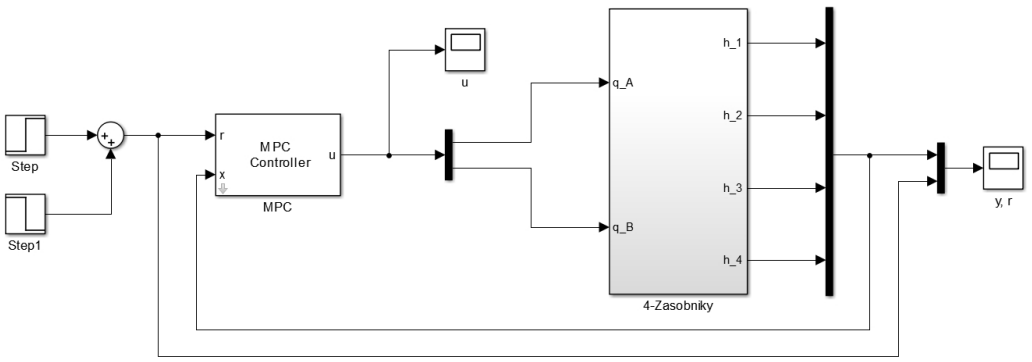
Nelineárny dynamický matematický model (1.3.1) reprezentuje v simuláciách riadený proces, preto bol zostavený aj z blokov v Simulinku (Obr. 1.3.2).



Obr. 1.3.2: Schéma DMM štyroch zásobníkov kvapalín.

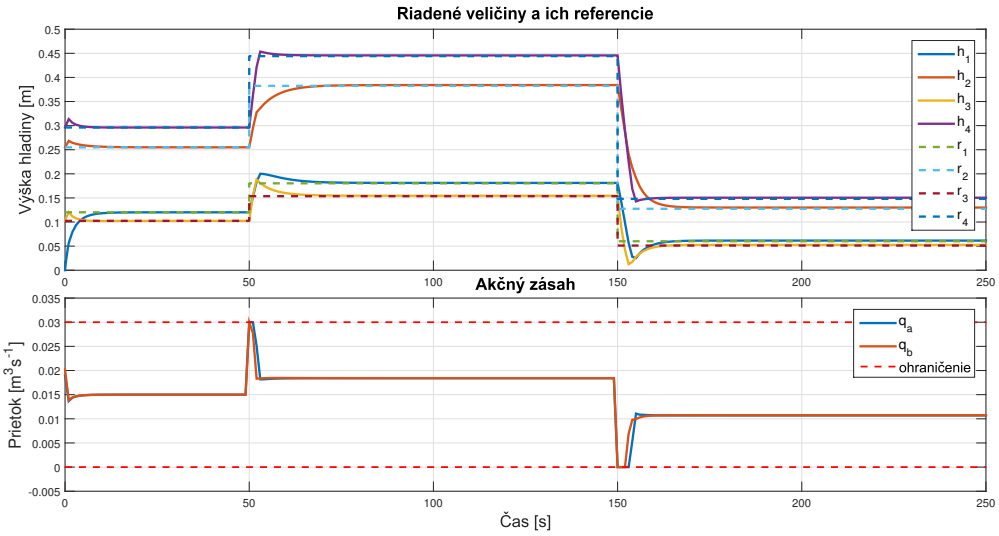
1.3.2 Úloha sledovania s MPC regulátorom

Prvým krokom k testovaniu bloku MPC regulátora bolo zostavenie zodpovedajúcej simulačnej schémy (Obr. 1.3.3) v Simulink-u, kde subsystém reprezentuje sústavu štyroch zásobníkov kvapalín zobrazenú na Obr. 1.3.1, ktorá je v Simulink-u zostavená z blokov ako na Obr. 1.3.2.



Obr. 1.3.3: Schéma URO s MPC regulátorom.

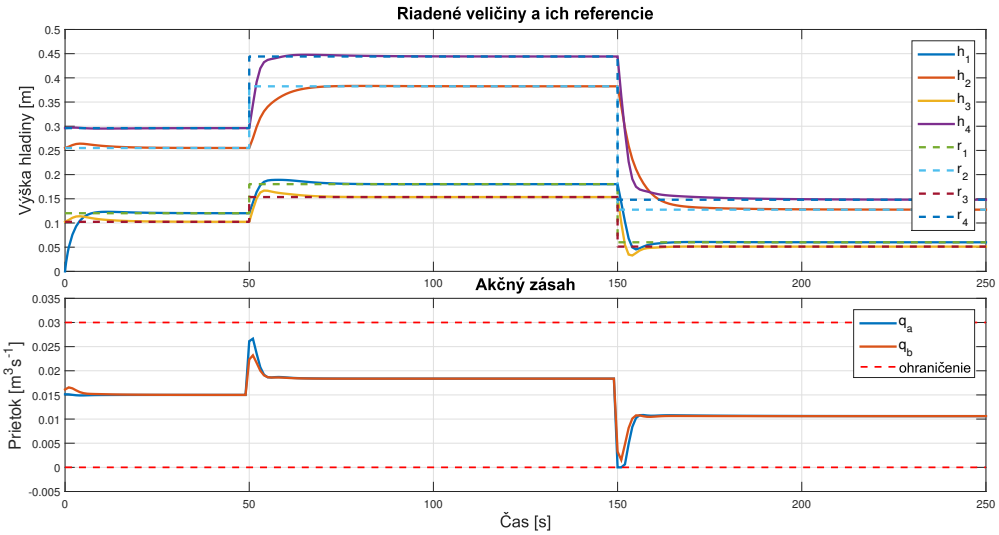
V simuláciách sme realizovali dve skokové zmeny referencie, ktoré zodpovedali n-násobku hodnôt v ustálenom stave, nakoľko riadime štyri výstupy, respektíve stavy prostredníctvom dvoch riadiacich veličín (systém nie je plne riaditeľný). Zároveň uvažujeme, že všetky výšky hladín - stavové veličiny sú priamo merateľné a predstavujú výstup zo systému, takže nie je potrebné do schémy pridať pozorovač stavov. Ďalej je potrebné poznamenať, že v čase $t = 0$ sú všetky výšky hladín v ustálenom stave, okrem výšky hladiny v prvom zásobníku (h_1), ktorá je nulová (zásobník je prázdny). Oba vstupné prietoky q_a aj q_b sú ohraničené a preto môžu nadobúdať len hodnoty z intervalu $< 0,\ 0.03 > m^3 s^{-1}$. Pri prezentovaných simuláciách je potrebné zohľadniť fakt, že ich účelom je preukázanie principiálnej funkcionality bloku a nie dosahovanie čo najlepšej kvality riadenia.



Obr. 1.3.4: Priebeh riadenia s MPC regulátorom.

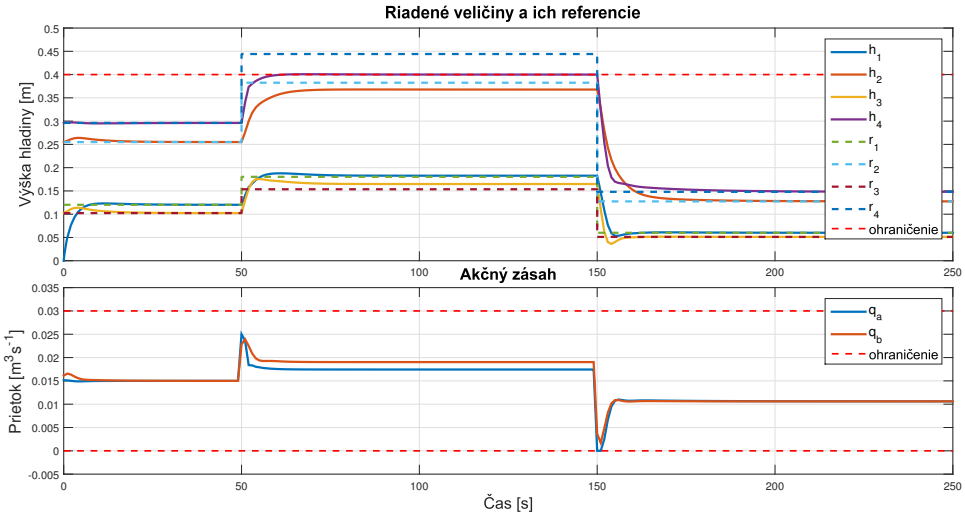
Prvá simulácia (Obr. 1.3.4) bola realizovaná so štandardným - "*Basic*" blokom MPC regulátora, ktorý implementuje len základnú formuláciu MPC (1.1.3). Z uvedeného grafu 1.3.4 je zrejmé, že regulátor najskôr naplní zásobník č. 1, čím dostane systém do ustáleného stavu a zároveň jednotlivé výstupy na príslušné žiadané hodnoty. Referencie sú dosiahnuté bez trvalej regulačnej odchýlky nakoľko tento ustálený stav je zároveň linearizačným bodom pre odvodenie lineárneho modelu pre MPC. Avšak v iných pracovných bodoch sa objavuje malá TRO, ktorá je zapríčinená nepresnosťou modelu. Ohraničenia na veľkosť akčných zásahov boli MPC regulátorom dodržané, no pri snahe ohraničiť stavové, respektíve výstupné veličiny štandardná formulácia MPC zlyhá. Zlyhanie je spôsobené neriešiteľnosťou daného optimalizačného problému, ktorá je zapríčinená nezlučiteľnou počiatočnou podmienkou x_0, ktorá do MPC vstupuje zo spätnej väzby. Počiatočný stav x_0 sa dostáva mimo ohraničení práve kvôli odchýlke medzi modelom a procesom - MPC na základe lineárneho modelu predpokladá, že sa s daným akčným zásahom tesne udrží v rámci ohraničení, no spätná väzba z procesu následne informuje MPC, že stavová veličina je už mimo ohraničení a tým spôsobí neriešiteľnosť (infeasibility) optimalizačného problému. Dôležité je poznamenať, že v prípade riadenia procesu s lineárnym správaním, ktoré plne zodpovedá modelu implementovanému v MPC, dokážeme riadiť bez TRO a všetky ohraničenia sú rešpektované.

Lineárne správanie procesu ako aj dokonalý model sú však iba abstrakciou a preto vznikol MPC regulátor v prevedení "*Advanced*", ktorý si dokáže poradiť s rôznymi praktickými aspektami implementácie prediktívneho riadenia. Verzia *Advanced* poskytuje možnosť aktivácie celkove troch rozšírení: delta u formulácie, modelovania a odhadu porúch, ako aj mäkké ohraničenia na stavové a výstupné veličiny, no cenou je vyššia výpočtová náročnosť algoritmu. Po aktivácii týchto rozšírení, nastavení váhovej matice Q_{u} na nulu a naladení Kalmanovho filtra dochádza k eliminácii TRO a tým k zlepšeniu kvality riadenia (Obr. 1.3.5), no hlavne k zvýšeniu stability regulačného obvodu. V porovnaní so štandardnou formuláciou je možné pozorovať riadenie bez TRO, menšie preregulovania, plynulejší a menej agresívny priebeh riadiacej veličiny, no aj o čosi dlhší čas regulácie. Samozrejme, kvalitu riadenia je možné ďalej zvyšovať ladením štyroch váhových matíc MPC, prípadne doladením kalmanovho filtra, ktorý má tiež v konečnom dôsledku významný vplyv na kvalitatívne ukazovatele.



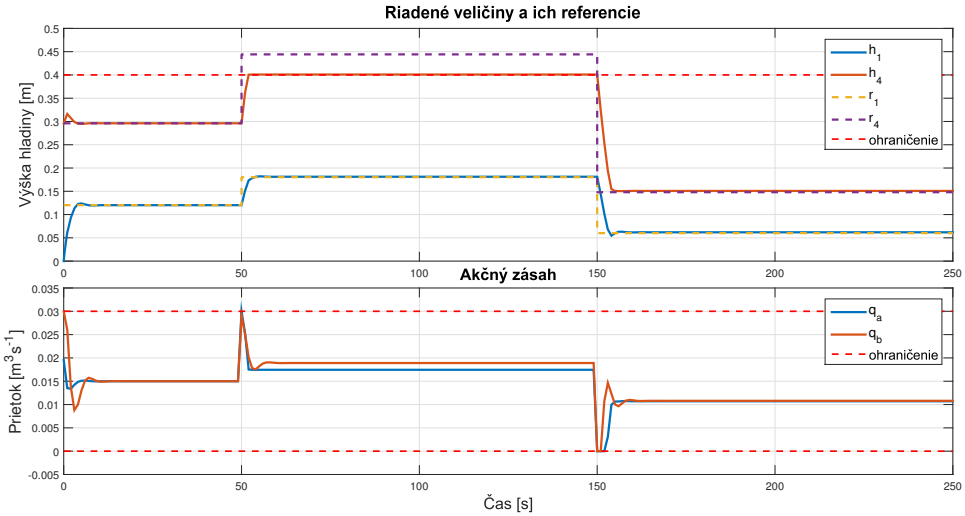
Obr. 1.3.5: Priebeh riadenia s MPC Advanced regulátorom.

Po pridaní ohraničení na výstupné veličiny, ktoré limitujú maximálne výšky hladín v zásobníkoch na hodnotu 0.4 m sa priebeh riadenia výrazne zmení (Obr. 1.3.6). Z grafu vyplýva, že ohraničenie primárne ovplyvní iba výšku hladiny v 4. zásobníku, no ako už bolo vyššie zmienené, daný systém nie je plne riaditeľný a preto sa zmena prejaví aj v ostatných výstupoch, nakoľko požadovaný ustálený stav nie je dosiahnuteľný.



Obr. 1.3.6: Priebeh riadenia s MPC Advanced regulátorom.

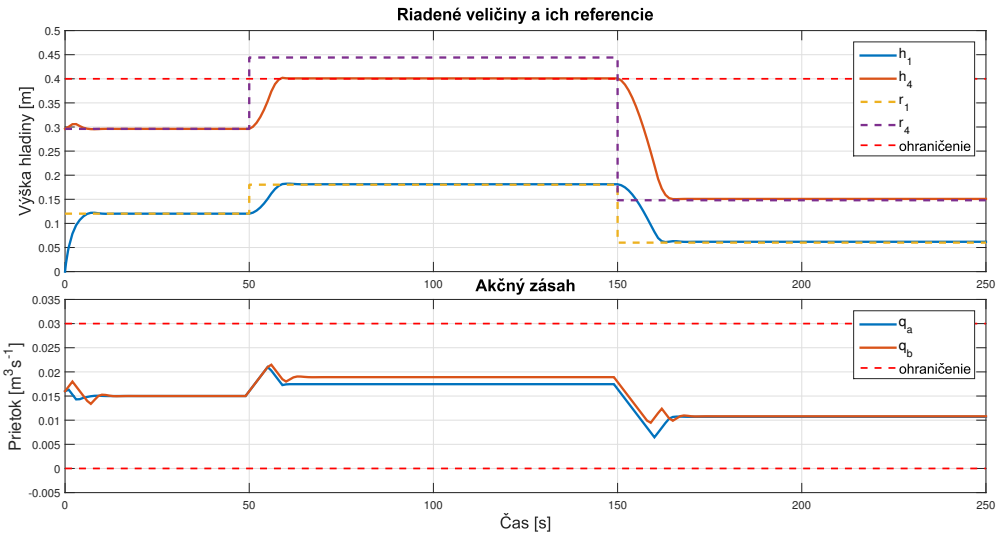
Keďže k dispozícii máme iba dve riadiace veličiny, tak nezávisle dokážeme riadiť iba dva výstupy. Preto pozmeníme matematický model - ako výstupné veličiny zadefinujeme iba výšku hladiny v prvom a štvrtom zásobníku, čo prakticky znamená modifikáciu matice C a D. Pri zachovaní ostatných nastavených parametrov a miernom pozmenení váhových matíc pozorujeme výraznejšie zlepšenie kvality riadenia (Obr. 1.3.7).



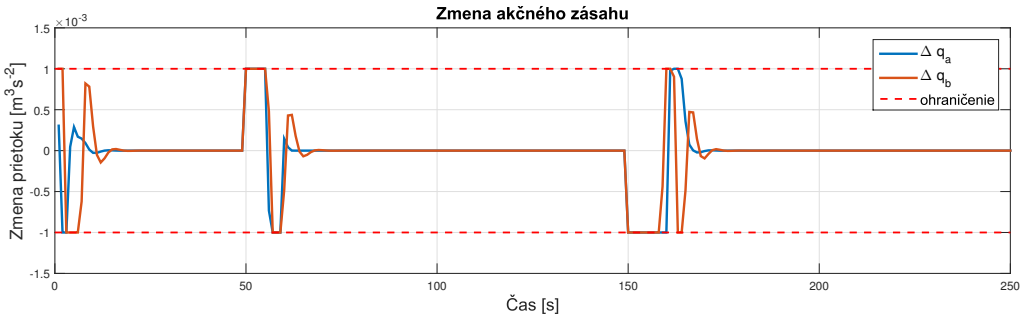
Obr. 1.3.7: Priebeh riadenia s MPC Advanced regulátorom.

Oba výstupy sledujú referenciu okrem situácie, kedy je referencia mimo ohraničení, prereguloványia sú zanedbateľné a čas regulácie je limitovaný veľkosťou akčného zásahu.

MPC regulátor v prevedení *Advanced* umožňuje pridanie ohraničení aj na veľkosť zmeny akčného zásahu (v prípade aktivovanej funkcie *delta u*), čím sa jednak docieli plynulejší priebeh riadenia (Obr. 1.3.8), ale taktiež umožní zohľadniť napríklad dynamiku akčného člena (ventilu). Ohraničenia pre Δu vymedzujú prípustné hodnoty z intervalu $< -1 \cdot 10^{-3}; 1 \cdot 10^{-3} >$, ktorých rešpektovanie je preukázané v grafe 1.3.9.



Obr. 1.3.8: Priebeh riadenia s ohraničeniami na zmenu akčného zásahu.



Obr. 1.3.9: Zmena akčného zásahu.

1.4 Zhrnutie

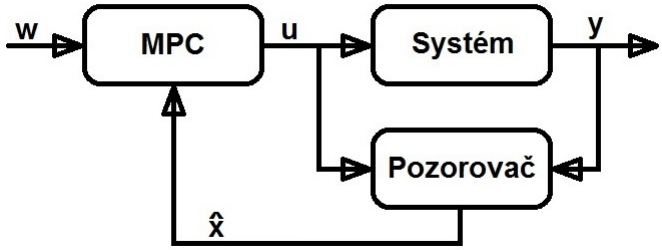
Prvá kapitola tejto práce bola venovaná prediktívnemu riadeniu zámerne, keďže všetky nasledujúce koncepty rozvíjané v rámci práce sú deriváty vychádzajúce práve z MPC. Zároveň práca mala za cieľ vytvoriť uplatnitelný produkt - knižnicu blokov podporných riadiacich systémov, v ktorej blok samotného MPC nemôže chýbať. Prakticky vznikli dva samostatné bloky implementujúce prediktívne riadenie, pričom blok v prevedení *Basic* pracuje so základnou formuláciou MPC (1.1.3) a vo verzii *Advanced* je možné aktivovať rôzne rozšírenia (1.1.4), ako napríklad modelovanie porúch, alebo mäkké ohraničenia. Hlavnou myšlienkou bolo vytvorenie platformy, ktorá je univerzálna, ľahko použiteľná, no zároveň sa dá ďalej rozvíjať, modifikovať a zdokonaľovať, aby čo najviac zodpovedala požiadavkám používateľa a aplikácie. Výkon bloku bol prezentovaný na nelineárnom modeli sústavy štyroch vzájomne prepojených zásobníkov kvapalín, ktorý je vzhľadom na interakcie medzi procesnými veličinami pomerne náročné riadiť štandardnými metódami (napr. PID regulátormi).

Kapitola 2

Odhad stavu na princípe posuvného horizontu - MHE

Pri implementácii regulátorov a riadiacich systémov sa často stretávame s rôznymi praktickými aspektami a obmedzeniami, ktoré je nevyhnutné vyriešiť, respektíve zohľadniť pre správne a efektívne riadenie. Jedným z týchto aspektov je aj odhad stavu - hodnôt stavových veličín, keďže častokrát je meranie stavových (vnútorných) veličín systému problematické, prípadne nemožné, no táto informácia je pre regulátor nevyhnutná. S týmto problémom sa stretneme aj pri implementácii prediktívneho riadenia (MPC), pre ktoré je jedným zo vstupov práve vektor aktuálnych stavových veličín (Obr. 2.1) a pri stavovom spätnoväzbovom regulátore je dokonca jediným vstupom. K dispozícii máme hneď niekoľko alternatívnych riešení. Asi najjednoduchšou voľbou je Luenbergerov pozorovač, potom sú tu rôzne modifikácie kalmanovho filtra, ktorý je asi najpoužívanejším, no v tejto práci sa budeme zaoberať pozorovačom stavov na báze posuvného horizontu - Moving Horizon Estimation (MHE), ktorý má tiež svoje prednosti, keďže je vo svojej podstate derivátom optimálneho MPC.

Vo všeobecnosti môžeme pozorovač stavu charakterizovať ako dynamický systém, ktorý generuje hodnoty aktuálnych stavov systému (x) na základe informácii o minulých a súčasných hodnotách vstupných (u) a výstupných (y) veličín. Dôležitou súčasťou pozorovača je tiež matematický model pozorovaného systému, na základe ktorého pracuje algoritmus. Samotný algoritmus výpočtu sa už líši na základe zvoleného typu pozorovača.



Obr. 2.1: Schéma zapojenia pozorovača stavu v URO s MPC regulátorom.

2.1 Formulácia MHE

Moving horizon estimation je typ pozorovača stavov, ktorý pracuje na princípe riešenia optimalizačného problému s ohraničeniami v každej perióde vzorkovania (Zavala, 2008). Odhad stavov je duálnym problémom vzhľadom k riadeniu, v tomto prípade by sme mohli povedať, že MHE je duálnym problémom vzhľadom k MPC, vychádzajúc z definícii pozorovateľnosti a riaditeľnosti systému. Na jeho analógiu s MPC poukazuje aj samotná formulácia optimalizačného problému, ktorá je do značnej miery podobná. Štandardnú formuláciu môžeme zapísať napríklad v takejto podobe:

$$\begin{aligned} \min \quad & \|Q_{x_0}(\hat{x}_{k-N} - x_0)\|_{\mathrm{p}} + \|Q_{\mathrm{yk}}(\hat{y}_k - y_k)\|_{\mathrm{p}} + \\ & + \sum_{i=0}^{N-1} (\|Q_{\mathrm{y}}(\hat{y}_{k-i} - y_{k-i})\|_{\mathrm{p}} + \|Q_{\mathrm{w}} w_{k-i}\|_{\mathrm{p}}) \end{aligned} \tag{2.1.1a}$$

$$\text{s.t.} \quad \hat{x}_{k-i+1} = A\hat{x}_{k-i} + Bu_{k-i}, \quad \text{pre} \quad i = 0, \ldots, N \tag{2.1.1b}$$

$$\hat{y}_{k-i} = C\hat{x}_{k-i} + Ew_{k-i}, \quad \text{pre} \quad i = 0, \ldots, N \tag{2.1.1c}$$

$$\hat{x}_{k-i} \in \mathcal{X}, \quad \text{pre} \quad i = 0, \ldots, N \tag{2.1.1d}$$

Význam jednotlivých členov a premenných je bližšie popísaný v nasledujúcich podkapitolách. MHE podobne ako MPC pracuje na určitom počte periód vzorkovania - na horizonte (N), ktorý zahŕňa aktuálnu periódu vzorkovania k a $N-1$ bezprostredne predchádzajúcich. Pre každú z týchto periód vzorkovania je potrebné poznať vstup do pozorovaného systému (u) a taktiež výstup (y). Nevyhnutná je tiež informácia o inicializačnom stave x_0, ktorá predstavuje hodnoty stavových veličín v čase $t - N\Delta t$, kde Δt reprezentuje veľkosť periódy vzorkovania. V uvedenej formulácii sú optimalizovanými premennými $\hat{x}_{k-i}$ - aproximácia stavov a w_{k-i} - výstupná porucha (v každej perióde vzorkovania vrámci horizontu).

2.1.1 Účelová funkcia

Účelovú funkciu (2.1.1a) základnej formulácie MHE môžeme rozdeliť na štyri členy, z ktorých každý je penalizovaný vlastným penalizačným koeficientom, prípadne maticou. Štandardne sa v účelovej funkcii používa kvadrát 2-normy, čiže penalizujeme kvadrát rozdielu, respektíve samotného čísla upravený pomocou váhovej matice.

Ako už bolo vyššie spomenuté, tak algoritmus MHE potrebuje ako jeden zo vstupných argumentov inicializačný stav systému x_0. Problémom však je, že tento stav nemusíme poznať (ak by sme ho poznali nepotrebujeme pozorovač stavu), preto sa predpokladá, že zadaná hodnota tohto stavu je len približná, alebo je to predchádzajúci výstup zo samotného pozorovača a tým pádom je zaťažený chybou. Prvý člen účelovej funkcie $\|Q_{x_0}(\hat{x}_{k-N} - x_0)\|_{\mathrm{p}}$ charakterizuje práve to, do akej miery je hodnota zadaného inicializačného stavu spoľahlivá, respektíve presná. Ak predpokladáme, že hodnota zadaného x_0 je presná, tak potom môžeme nastaviť vysokú hodnotu váhovej matice Q_{x_0} a tým pádom bude MHE zadaný stav v maximálnej miere rešpektovať, a odhadovaný stav v danej perióde vzorkovania bude nastavený na túto hodnotu.

MHE funguje na základe matematického modelu pozorovaného systému, najčastejšie lineárneho v maticovom tvare (2.1.1b)-(2.1.1c). To však so sebou prináša ďalší problém, keďže pozorovaný systém je častokrát nelineárny a spravidla sa nespráva totožne ako model. Preto rozlišujeme y, ktorý reprezentuje výstup z reálneho procesu, respektíve z meracieho člena a $\hat{y}$, ktorý je vypočítaný na základe matematického modelu. Snahou MHE je nájsť také aproximácie stavov, ktoré minimalizujú rozdiel $\hat{y} - y$, čo je základným predpokladom konzistentného odhadu stavu. Tento člen sa nachádza v účelovej funkcii dvakrát a to z toho dôvodu, že prvý penalizuje rozdiel v aktuálnej perióde vzorkovania, ktorá ovplyvňuje primárny výstup - aktuálny stav systému a druhý člen je pre všetky ostatné - minulé periódy vzorkovania. Ak odhadnutý stav systému, ktorý je zároveň výstupom nekopíruje výstup zo systému, tak je potrebné zvýšiť hodnoty váhových matíc Q_{yk} a Q_{y}.

Nepresnosť modelu a vplyv porúch na reálny systém je kompenzovaný pomocou poruchy vo výstupnej (w) (prípadne aj v stavovej) rovnici. Posledný člen účelovej funkcie penalizuje práve veľkosť tejto poruchy. Hodnota výstupnej a stavovej poruchy je tiež výstupom z MHE, preto môžeme sledovať ich vývoj a na jeho základe vhodne zvoliť váhové matice.

2.1.2 Ohraničenia

Ohraničenia sú azda hlavnou devízou MHE. Principiálne je možné zadefinovať ľubovoľné ohraničenia v podobe rovností, nerovností, intervalov, ohraničenia na typy premenných, prípadne mäkké ohraničenia a podobne. Práve ohraničenia umožňujú zohľadniť špecifiká daného pozorovaného systému. V uvedenej štandardnej formulácii (2.1.1) sú zahrnuté iba základné ohraničenia.

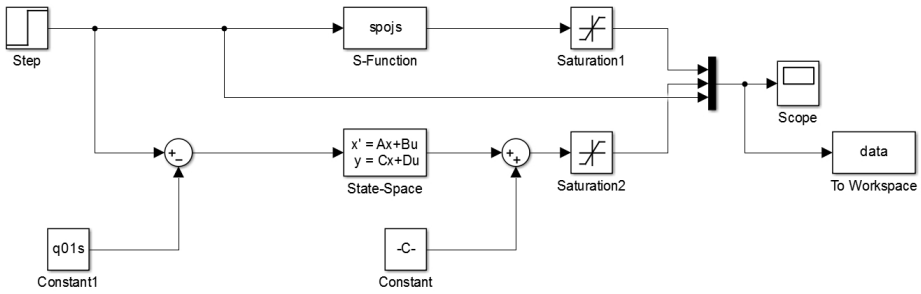
Prvé ohraničenie (2.1.1b) reprezentuje stavovú rovnicu pozorovača, ktorá dáva do súvisu optimalizované premenné s matematickým modelom pozorovaného systému. Ďalším nevyhnutným ohraničením je výstupná rovnica (2.1.1c), ktorej výstupom je odhad riadenej veličiny a v tomto prípade sa v nej implementuje aj porucha, použitie ktorej by malo viesť na konzistentnejší odhad stavu. Posledným použitým ohraničením je ohraničenie hodnôt stavových veličín(2.1.1d). Použitie ohraničení tohto typu vedie jednak na konzistentnejší odhad stavu a taktiež môže mať význam vzhľadom na stabilitu a ďalšie spracovanie výstupu (odhadnutých stavov).

2.2 Simulačné overenie

Pre implementáciu algoritmu na báze MHE bolo zvolené programovacie prostredie Matlab v kombinácii s rozširujúcim toolboxom YALMIP. Tento výber je opodstatnený jednak tým, že Matlab umožňuje simulačné overenie v rámci Simulinku, ale taktiež riadenie systémov v reálnom čase, čo sú cieľové oblasti uplatnenia tohto algoritmu. Jedným z cieľov projektu bolo taktiež vytvorenie kompaktného, “user-friendly” bloku pre Simulink, ktorý by plnil funkciu pozorovača stavov.

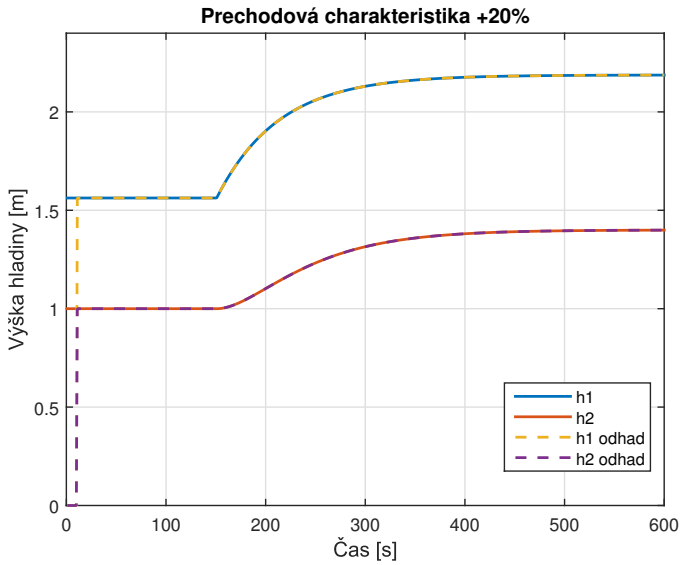
Pre účel návrhu, testovania a ladenia algoritmu MHE bol použitý model systému dvoch zásobníkov kvapalín bez interakcie, pričom výstupom je výška hladiny v druhom zásobníku a vzhľadom k tomu, že bol implementovaný pozorovač plného rádu, tak úlohou je odhadovať výšky hladín v oboch zásobníkoch. Pre porovnanie, overenie správnosti výsledkov a získanie nevyhnutných vstupov ($u(t)$, $y(t)$ a x_0) boli realizované simulácie v simulinku s lineárnym modelom (zhodný s tým, ktorý bol implementovaný v MHE), ale aj s nelineárnym modelom, ktorého úlohou bolo reprezentovať reálny proces.

Zásobníky kvapalín sú nelineárne systémy, preto bola naprogramovaná S-Funkia, ktorá presnejšie opisuje ich správanie. S-Funkcia bola implementovaná v rámci simulačnej schémy v prostredí Simulink-u (Obr. 2.2.1). Táto schéma je zároveň zdrojom procesných dát pre výpočtový script, pričom je možné získať dáta z lineárneho ako aj nelineárneho modelu.



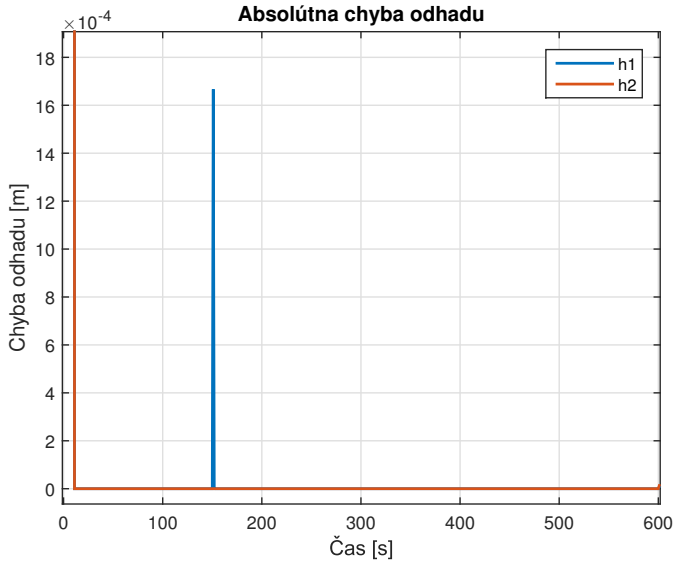
Obr. 2.2.1: Simulinková schéma - zásobníky.

Funkčnosť algoritmu bola najskôr overená na lineárnom modeli sústavy zásobníkov - predpokladali sme, že reálne zariadenie sa správa presne podľa lineárneho modelu, ktorý bol použitý v pozorovači stavov. Nasledujúci graf - Obr. 2.2.2 prezentuje výkon pozorovača v takomto prípade.

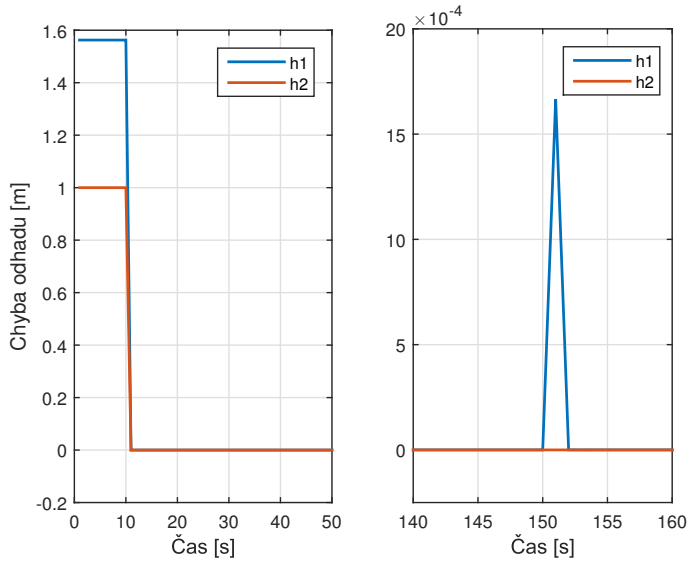


Obr. 2.2.2: Pozorovanie lineárneho modelu.

Pozorovať môžeme v podstate dokonalý odhad oboch stavov, čo potvrdzuje aj graf chyby odhadu Obr. 2.2.3. Na základe týchto výsledkov môžeme dedukovať, že chyba pri použití úplne presného matematického modelu je zanedbateľná a tým zároveň potvrdiť funkčnosť pozorovača stavov.



Obr. 2.2.3: Chyba odhadu stavu lineárneho modelu v 10^{-4} m.

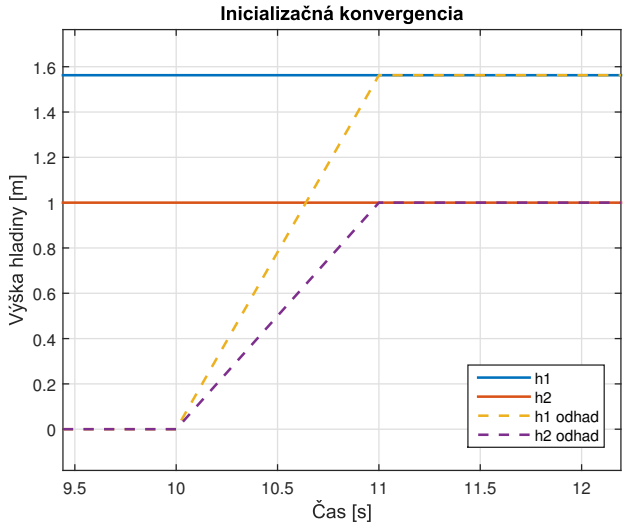


Obr. 2.2.4: Chyba odhadu stavu lineárneho modelu - detail.

Graf 2.2.3 však zachytáva aj dve udalosti zobrazené v grafe 2.2.4, ktoré vyvolali odchýlku pozorovaných stavov od skutočných, preto je potrebné toto správanie objasniť.

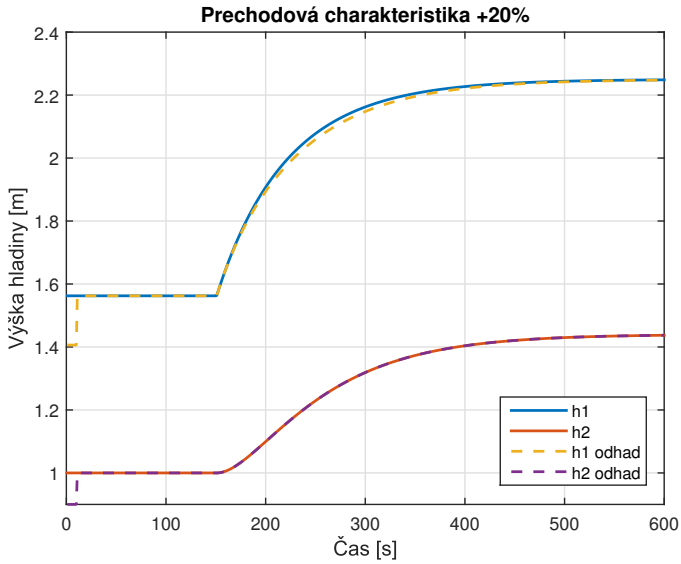
Odchýlka do času $t = 10$, ktorej stavový priebeh je zobrazený aj v grafe 2.2.5 je spôsobená nepresnými inicializačnými hodnotami stavových veličín. Ako už bolo vyššie spomenuté, tak pri štarte - inicializácii MHE je nevyhnutné zadať aproximáciu aktuálnych hodnôt stavových veličín, ktoré však spravidla nepoznáme. Z tohto dôvodu bol aj v tomto prípade algoritmus MHE inicializovaný nepresnými nástrelovými hodnotami $[0\ 0]^T$. Horizont MHE bol pri simulácii nastavený na 10 period vzorkovania, čo zodpovedá desiatim sekundám, počas ktorých prebieha inicializácia algoritmu a preto je výstupom konštantná hodnota pôvodne zadaného inicializačného stavu. Po nábehu MHE v čase 10 sekúnd (Obr. 2.2.5) pozorujeme okamžitú konvergenciu na skutočnú hodnotu stavových veličín v rámci jedinej periódy vzorkovania.

Druhá odchýlka, ktorá sa objavila v čase 150 sekúnd (Obr. 2.2.4) bola rádove nižšia (približne 1mm) a bola spôsobená skokovou zmenou vstupného prietoku v tomto čase. Je potrebné uvedomiť si, že MHE pracuje v diskrétnom čase, tým pádom má informácie o dianí v systéme iba v diskrétnych bodoch a nezohľadňuje vývoj medzi týmito bodmi. Poslednú informáciu o veľkosti vstupnej veličiny dostal MHE v čase 150 sekúnd, kedy bola ešte konštantná vzhľadom na predchádzajúce hodnoty, preto odhadol konštantný vývoj stavových veličín. Problémom však bolo, že bezprostredne po čase 150 sekúnd došlo k skokovej zmene vstupného prietoku, čo viedlo k nárastu výšky hladiny, hlavne v prvom zásobníku a tým vznikla odchýlka medzi odhadom a skutočnosťou. MHE zaregistroval zmenu prietoku až v nasledujúcej, 151. perióde vzorkovania a náležite pozmenil odhad stavov, čo sa prejavilo hneď v 152. perióde vzorkovania plne konzistentným odhadom stavu.



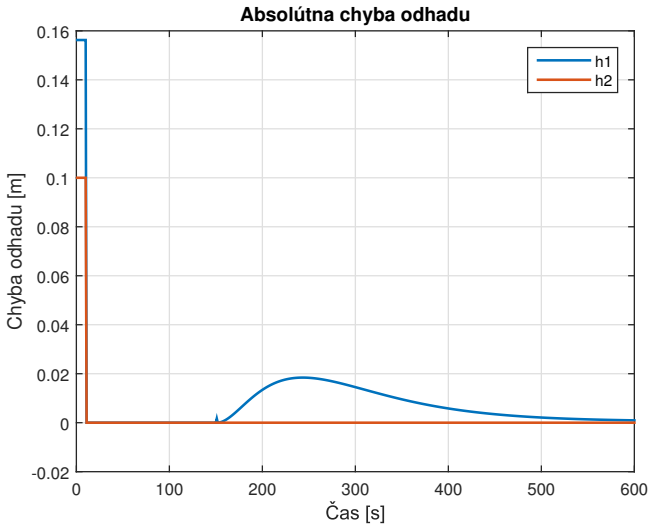
Obr. 2.2.5: Konvergencia po inicializácii.

Dokonale presný model je v prípade riadenia reálnych zariadení len abstrakcia a preto sa pozorovač stavov musí vysporiadať aj s nepresným matematickým modelom. Z tohto dôvodu už budeme pre získanie vstupných údajov (y, u) používať len nelineárny model, pri použití ktorého dosahuje MHE výkon prezentovaný v grafe 2.2.6.



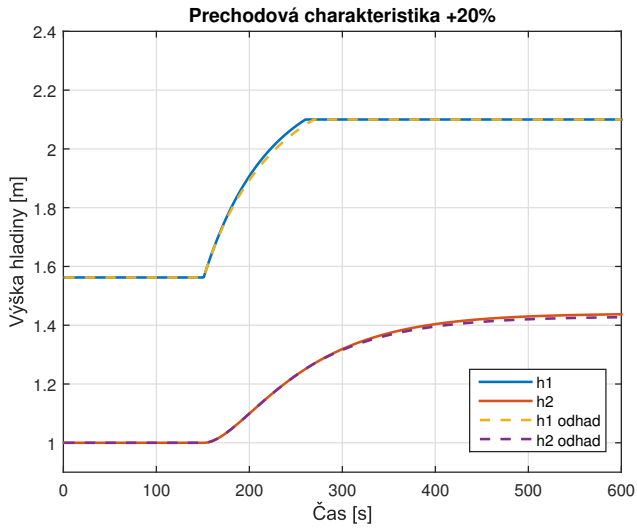
Obr. 2.2.6: Pozorovanie nelineárneho modelu.

Ako je z vyššie uvedeného grafu 2.2.6 zrejmé, tak nepresnosť modelu sa mierne prejaví na kvalite odhadu, a to hlavne v oblasti, respektíve čase, kedy je dynamika najvýraznejšia. Ak sa pozrieme na chybu odhadu, ktorá je vynesená v grafe 2.2.7, tak môžeme pozorovať, že chyba nepresahuje 2 cm, čo je pri výške hladiny cca 2 metre menej ako 1% a navyše asymptoticky klesá k nule. Táto chyba sa však prejavuje v odhadovanom stave a stav, ktorý reprezentuje výstup zo systému a zároveň charakterizuje kvalitu (chybu) pozorovača stavu je odhadovaný úplne presne. Je potrebné poznamenať, že váhové matice boli upravené pri prechode z lineárneho na nelineárny pozorovaný systém.



Obr. 2.2.7: Chyba odhadu pre nelineárny model.

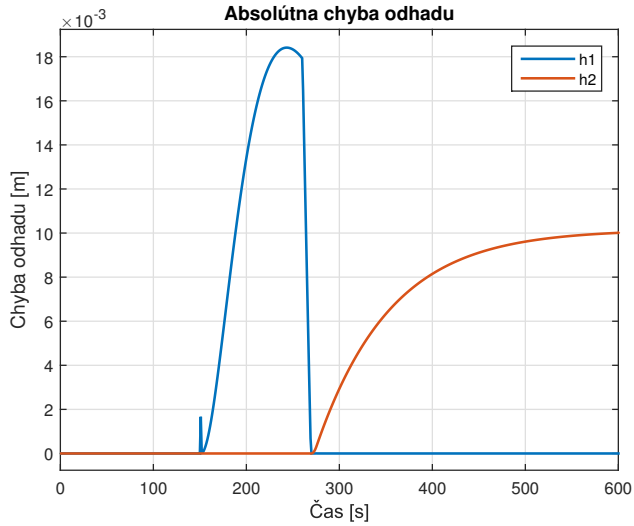
Ak aplikujeme ohraničenia na stavy - inými slovami určíme minimálnu a maximálnu reálne dosiahnuteľnú výšku hladiny v daných zásobníkoch, ktorá je daná ich konštrukciou, tak dostaneme výsledky prezentované v nasledujúcom grafe 2.2.8.



Obr. 2.2.8: Pozorovanie systému s ohraničením pomocou MHE.

Z grafu 2.2.8 vyplýva, že MHE rešpektuje dané ohraničenie na maximálnu výšku

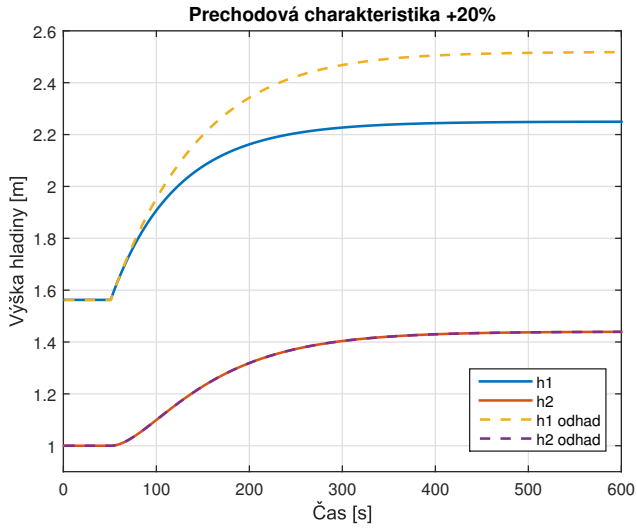
hladiny nastavené na 2,1 metra, čo v praxi môže udávať konštrukčnú výšku zásobníka. Minimálna výška hladiny bola nastavená na 0 metrov - prázdny zásobník, čo sa však v tomto prípade vzhľadom na veľkosť skokovej zmeny vstupu neprejaví a neovplyvní prechodovú charakteristiku.



Obr. 2.2.9: Chyba odhadu pre systém s ohraničením v 10^{-3} m.

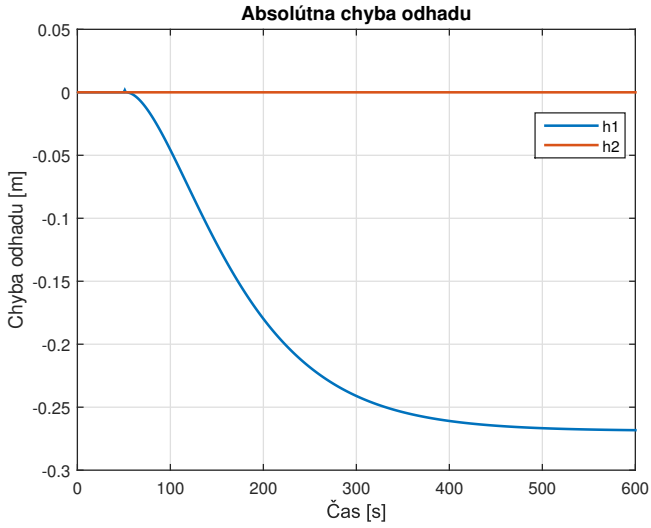
Z grafu chyby (Obr. 2.2.9) je možné vyčítať, že po tom ako sa odhadovaný stav saturuje na ohraničení, tak ho odhadujeme presne, ale chyba vyplývajúca z nelinearity pozorovaného systému (plant-model mismatch) sa prejaví v odhade stavu, ktorý reprezentuje výstup. Tento príklad poukazuje na fakt, že chyba modelu sa niekde prejaví a práve váhové matice rozhodnú o tom kde. Avšak použitím modelovania výstupných a stavových porúch v kombinácii s vhodne zvolenými váhovými maticami je možné túto chybu znížiť, prípadne aj úplne eliminovať.

Význam váhových matíc je možné demonštrovať aj na nasledujúcom príklade. Za predpokladu, že k dispozícii je iba menej presný model (použijeme maticu A prenásobenú koeficientom 0,99), tak pri zachovaní nastavenia váhových matíc (z príkladu s nelineárnym procesom) poskytne pozorovač odhad vynesený v grafe 2.2.10.



Obr. 2.2.10: Výkon pozorovača MHE s nepresným modelom.

Vzhľadom k tomu, že váhová matica Q_y je nastavená na vysokú hodnotu, tak stav reprezentujúci výstup je odhadovaný presne a odchýlka modelu sa prejaví v odhadovanom stave - výške hladiny v prvom zásobníku.

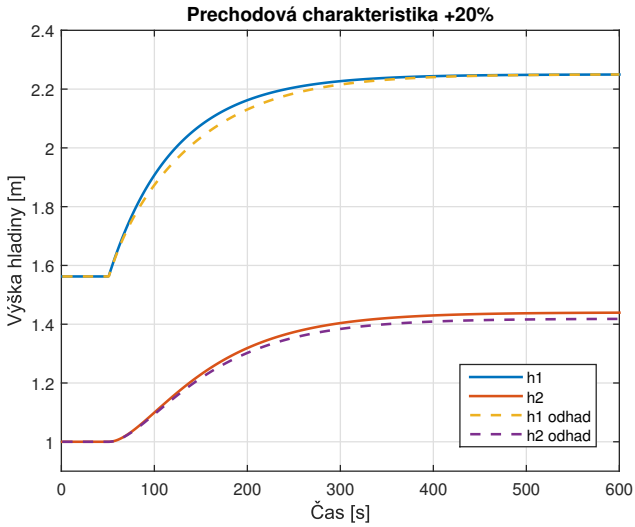


Obr. 2.2.11: Chyba odhadu s nepresným modelom.

Chyba odhadu sa pohybuje na úrovni približne štvrť metra Obr. 2.2.11, čo predstavuje

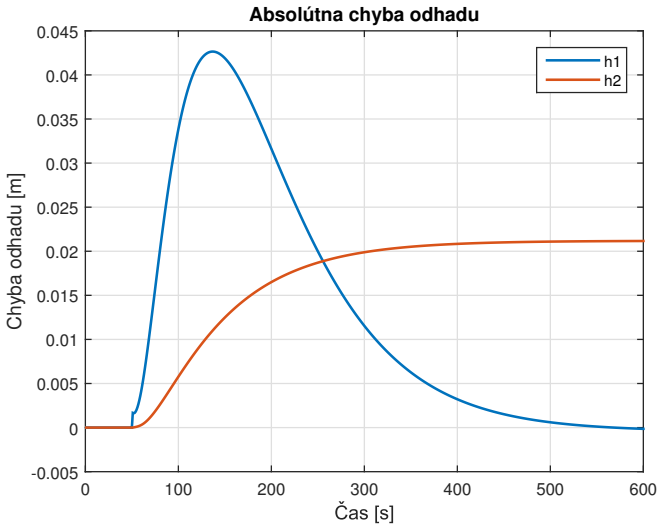
zhruba 13%, takže môžeme hovoriť už o pomerne signifikantnej chybe.

Preto zmeníme nastavenie váhových matíc, čo má za následok výraznejšie zlepšenie prezentované v grafe 2.2.12:



Obr. 2.2.12: Odhad stavov s nepresným modelom.

Absolútna chyba pri odhadovanom stave asymptoticky klesá k nule, no mierna odchýlka sa prejavuje pri stave h_2, ktorého význam je avšak v konečnom dôsledku zanedbateľný, keďže primárnou úlohou je získať hodnotu h_1. Zaujímavé a veľmi užitočné je, že ak zachováme model a nastavenie váhových matíc, tak obdobné výsledky získame aj pri iných skokových zmenách (testované boli skokové zmeny od -50% do +70%), pri ktorých je chyba odhadu h_1 minimálna. Praktický význam tohto poznatku spočíva v tom, že ak je odhadovaný stav ťažko merateľný, prípadne je možné jeho hodnotu zistiť len analyticky v laboratóriu, a pod., tak postačuje jedno meranie v ustálenom stave, v určitej vzdialenosti od linearizačného bodu, pre ktoré odladíme váhové matice a tým získame spoľahlivý odhad v pomerne širokom rozsahu.

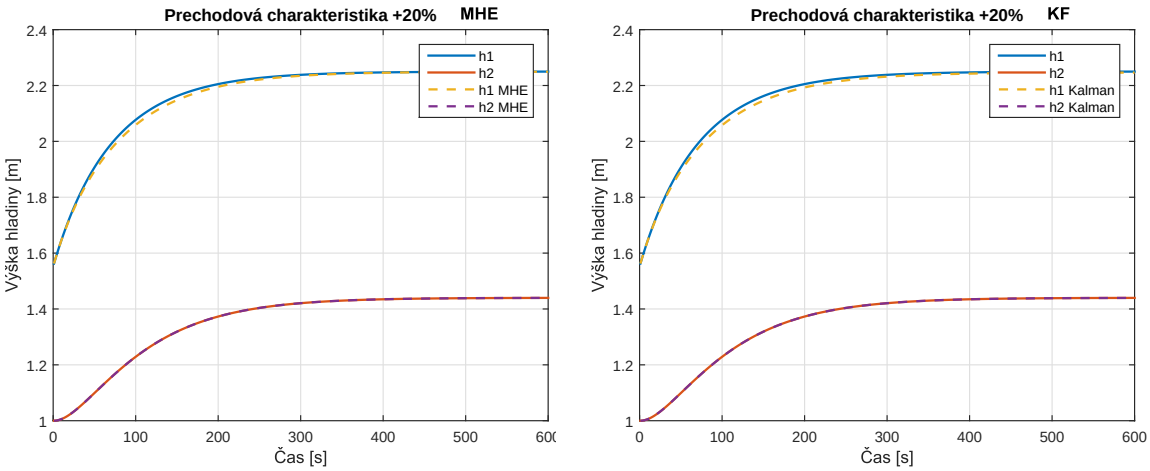


Obr. 2.2.13: Chyba odhadu s nepresným modelom.

2.2.1 Porovnanie s inými vybranými pozorovačmi

Problematika odhadu stavu ponúka väčšie množstvo osvedčených riešení napríklad v podobe rôznych modifikácii kalmanovho filtra, prípadne jednoduchšieho Luenbergerovho pozorovača, preto je potrebné preukázať, že aj MHE je plnohodnotnou alternatívou, alebo v niektorých prípadoch dokonca optimálnou cestou, pre odhad stavu. Podrobnejšie sa problematike odhadu stavu s vybranými typmi pozorovačov venuje aj Kalman (1960) a Besancon (2007).

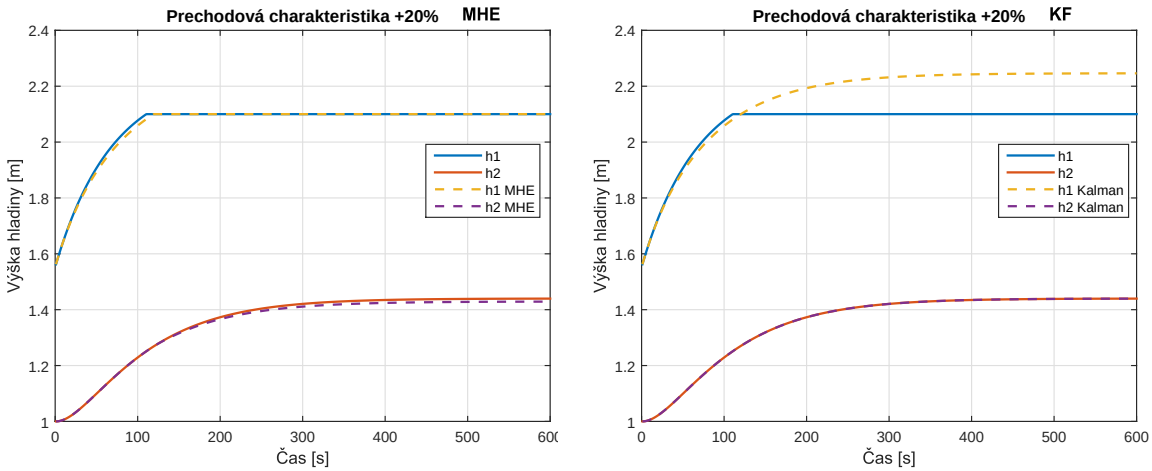
Ako prvý budeme porovnávať výkon MHE s Kalmanovým filtrom pri štandardných podmienkach.



Obr. 2.2.14: Porovnanie MHE s kalmanovým filtrom.

Z vyššie zobrazeného grafu Obr. 2.2.14 vyplýva, že MHE dosahuje v podstate rovnaké výsledky ako dobre nastavený kalmanov filter. U kalmanovho fitra sú však ladiacimi parametrami jeho póly, ktoré taktiež ovplyvňujú jeho stabilitu a tým aj stabilitu celého uzavretého regulačného obvodu. Pre nastavenie MHE sa primárne používajú váhové matice, ktorých hodnoty môžeme voliť nezávisle a majú skôr relatívny význam.

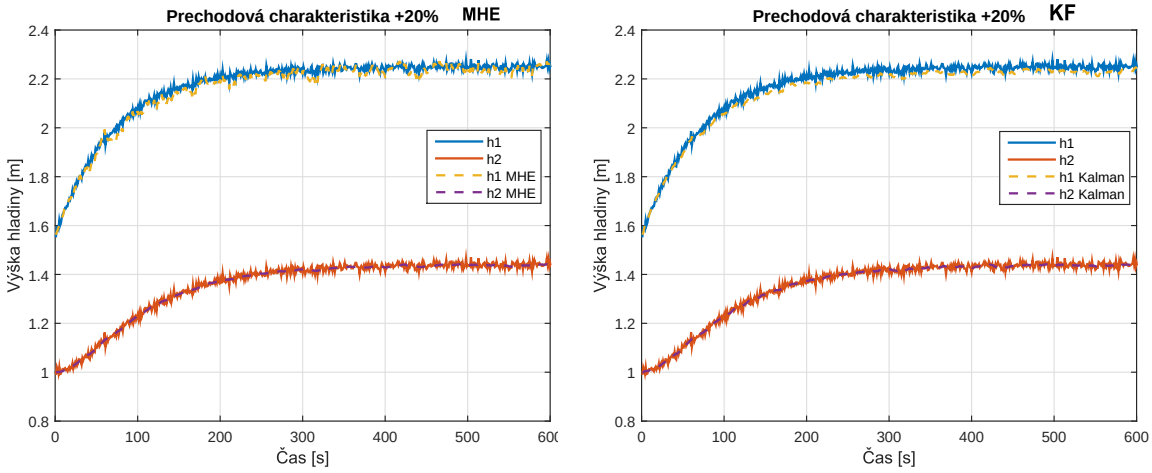
Ďalej uvažujme situáciu, že výšky hladín sú ohraničené výškou 1. zásobníka na 2,1 metra:



Obr. 2.2.15: Porovnanie MHE s KF pre systém s ohraničením.

Z vyššie uvedeného (Obr. 2.2.15) je zrejmé, že reálny systém sa saturuje na 2,1 metroch a MHE rešpektuje toto ohraničenie. Naopak kalmanov filter toto ohraničenie nezohľadní a preto nám poskytne výrazne chybný odhad stavu.

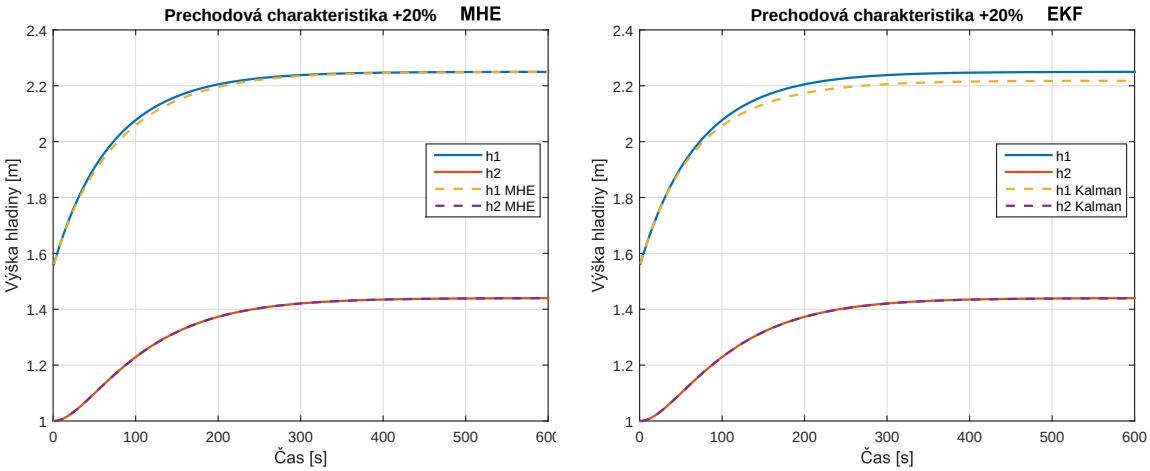
Ak je úlohou pozorovať reálny systém, tak je takmer isté, že výstup (y) bude mierne zašumený. Tento šum je spôsobovaný hlavne meracím členom, alebo napríklad aj vlnením hladiny v zásobníku, prípadne inými rušivými faktormi.



Obr. 2.2.16: Porovnanie MHE s KF pri zašumenom signále.

Znovu môžeme konštatovať veľmi podobné výsledky (Obr. 2.2.16) a výkon oboch pozorovačov, ale póly kalmanovho filtra sú nastavené už veľmi blízko nestabilnej oblasti, naopak MHE je možné ešte ďalej ladiť a tým zlepšiť jeho výkon. Ak sa bližšie pozrieme na výsledky, tak zistíme, že odhad MHE viac kmitá, no stredná hodnota lepšie kopíruje reálny stav v porovnaní s Kalmanovým filtrom.

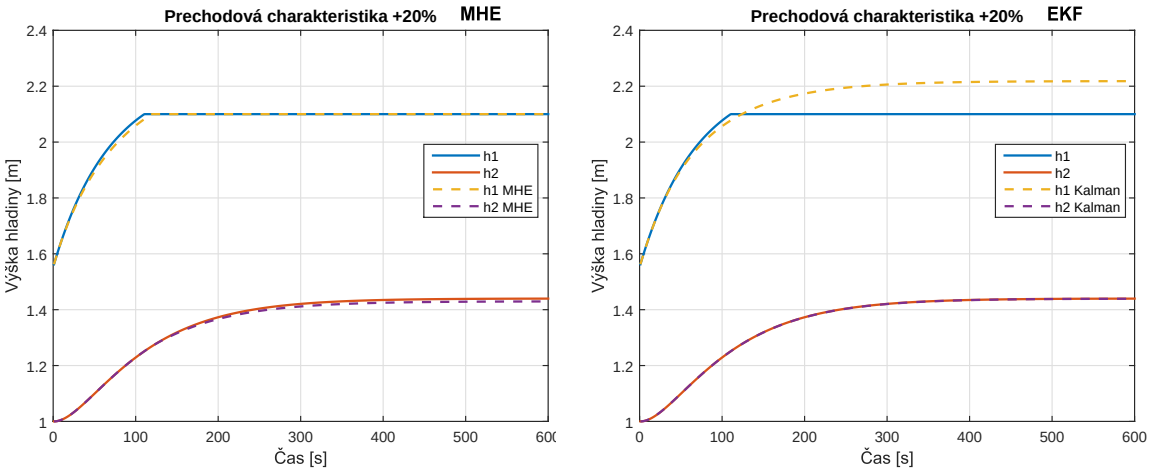
Podobný proces porovnania zopakujeme aj pre dynamický kalmanov filter.



Obr. 2.2.17: Porovnaie MHE s EKF.

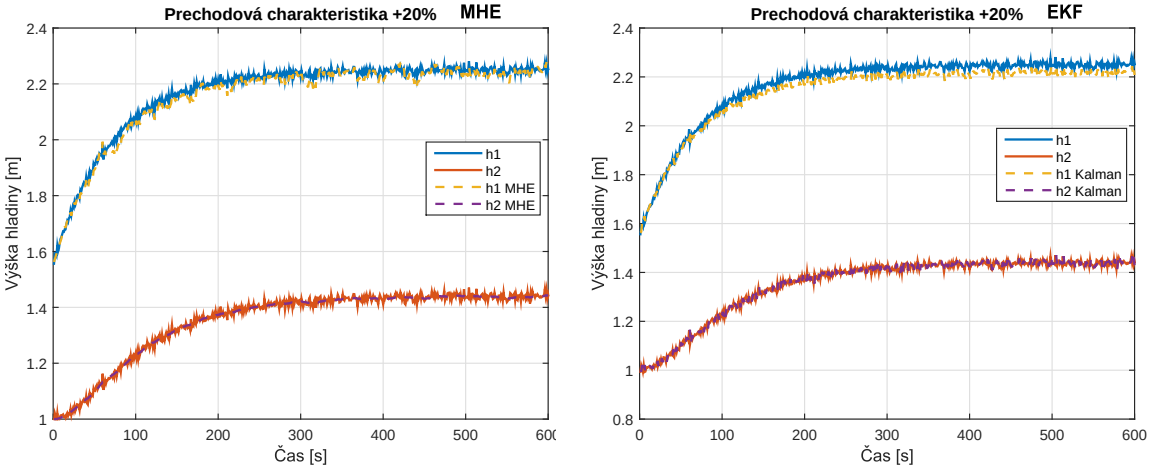
Pri štandardných podmienkach obstál MHE lepšie v porovnaní s dynamickým kalmanovým filtrom. Aj napriek veľkej snahe pri ladení kalmanovho filtra sa nepodarilo znížiť odchýlku odhadu h_2, pričom čas venovaný ladeniu bol porovnateľný ako pri ladení MHE.

Po aplikácii ohraničení získame podobné výsledky (Obr. 2.2.18) ako pri klasickom kalmanovom filtri, pretože ani dynamický kalmanov filter nedokáže zohľadniť ohraničenia.



Obr. 2.2.18: Porovnanie MHE s EKF pre systém s ohraničením.

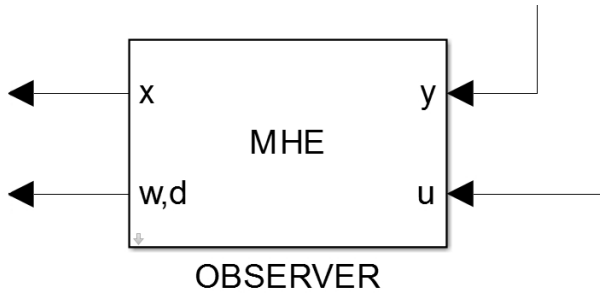
Ak použijeme signál y s vplyvom šumu, tak opäť môžeme konštatovať rovnaké ako pri štandardnom kalmanovom filtri.



Obr. 2.2.19: Porovnanie MHE s EKF pri zašumenom signále.

2.3 Blok MHE pre Simulink

Jedným z cieľov práce bolo vytvorenie kompaktného bloku pre odhad stavu na báze MHE, ktorý by bol použiteľný v Simulinku. Táto časť projektu bola azda najkomplikovanejšia, keďže bolo potrebné implementovať pomerne zložitý optimalizačný algoritmus v prostredí Simulink pričom cieľom je pracovať v reálnom čase a preto bolo potrebné zabezpečiť dostatočne nízku časovú náročnosť tohto algoritmu. Ďalšou výzvou bolo vytvorenie "user-friendly" prostredia pre zadávanie a spracovanie vstupných hodnôt a parametrov, ktoré by bolo čo možno najzrozumiteľnejšie a najintuitívnejšie. Samozrejme bolo tiež potrebné, aby bol blok univerzálny a dostatočne flexibilný pre rôzne aplikácie odhadu stavu.

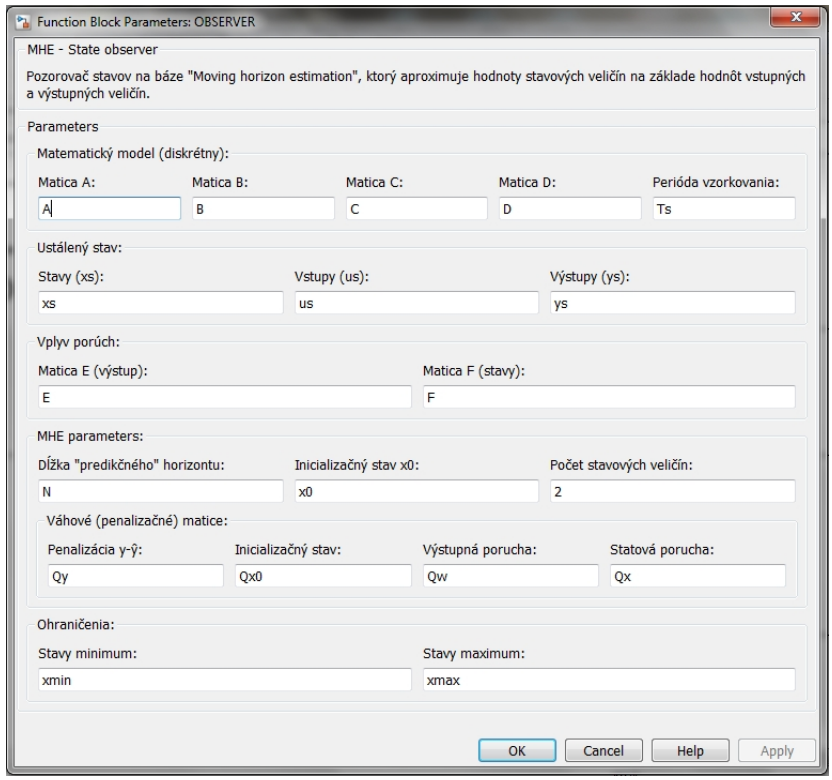


Obr. 2.3.1: Blok MHE vytvorený pre použitie v Simulinku.

2.3.1 Nastavenie parametrov a subsystém

Ako je z vyššie uvedeného obrázka 2.3.1 zrejmé, tak vstupom do pozorovača stavov je u - riadiaca veličina a y - riadená, respektíve meraná veličina. Primárnym výstupom sú odhadnuté stavy (x), ale používateľ môže taktiež sledovať vývoj stavovej a výstupnej poruchy (w, d). Ostatné (konštantné) parametre sú zadávané prostredníctvom masky (Obr. 2.3.2).

Parametre vrámci masky sú rozdelené do viacerých tematicky súvisiacich skupín, ktorých cieľom je uľahčiť orientáciu a zároveň znížiť priestorové požiadavky, keďže je potrebné zadať väčšie množstvo parametrov.



Obr. 2.3.2: Maska bloku MHE.

Ak používateľ klikne na tlačítko "Help", alebo si zobrazí parametre bloku, tak je k dispozícii nasledovná nápoveda, ktorá stručne opisuje význam a funkciu bloku a zároveň podrobnejšie opisuje jednotlivé vstupné parametre:

Tento blok plní funkciu pozorovača stavov systému, ktorého dynamiku je možné popísať pomocou matíc lineárneho stavového opisu.

Vstupom sú hodnoty riadiacich veličín (u) a výstupných veličín (y), ktoré sú v rámci bloku archivované pre celú dĺžku predikčného horizontu. Výstupom sú primárne aproximované - odhadnuté hodnoty stavových veličín (x), ale tiež hodnoty výstupnej (w) a stavovej poruchy (d). Výstupom je tiež nový vektor počiatočných stavov (x_0), ktorý je avšak vstupným parametrom funkcie MHE pre ďalšiu periódu vzorkovania, preto je využívaný len vrámci subsystému.

Všetky konštantné parametre potrebné pre správne fungovanie algoritmu je možné zadať prostredníctvom masky tohto bloku (MHE). Tu je krátky popis jednotlivých parametrov:

-Matematický model:

ktorý čo možno najspoľahlivejšie opisuje vlastnosti a dynamiku riadeného (pozorovaného) systému je nevyhnutnou súčasťou a jeho presnosť má zásadný vplyv na kvalitu odhadu stavov. Očakáva sa lineárny matematický model v maticovej podobe v diskrétnej časovej oblasti s periódou vzorkovania (Ts).

- Matica A: je matica popisujúca dynamiku pozorovaného systému. Očakáva sa štvorcová matica s rozmermi odpovedajúcimi počtu stavov - rádu systému.

- Matica B: opisuje vplyv riadiacej veličiny/ín (u) na hodnoty jednotlivých stavových veličín (x) systému. Počet riadkov musí byť zhodný s maticou A a počet stĺpcov reprezentuje počet vstupných veličín.

- Matica C: určuje vplyv jednotlivých stavov na hodnoty výstupných (riadených) veličín(y). Počet stĺpcov je zhodný s rozmerom matice A a počet riadkov je závislý na počte výstupných veličín. Táto matica má primárny vplyv na pozorovateľnosť daného systému.

- Matica D: udáva vplyv vstupných veličín (u) na výstup. Väčšinou je reprezentovaná nulovou maticou s tým, že počet riadkov je zhodný s počtom výstupov a počet stĺpcov zas s počtom vstupov.

- Ustálený stav:

zodpovedá hodnotám procesných veličín v linearizačnom bode - bode, pri ktorom bol odvodený lineárny matematický model.

- Stavy (xs): hodnoty stavových veličín v ustálenom stave. Sú reprezentované stĺpcovým vektorom s počtom prvkov rovným počtu stavov.

- Vstupy (us): hodnoty vstupných veličín v ustálenom stave. Sú reprezentované stĺpcovým vektorom s počtom prvkov rovným počtu vstupných veličín.

- Výstupy (ys): hodnoty výstupných veličín v ustálenom stave. Sú reprezentované stĺpcovým vektorom s počtom prvkov rovným počtu výstupných veličín.

- Vplyv porúch:

modelovanie porúch má svoje opodstatnenie vždy, keď matematický model plne nezodpovedá správaniu sa systému. Či už je rozdiel zapríčinený externou poruchovou veličinou (časovo variantná veličina vplývajúca na pozorovaný systém, ktorá nie je zadefinovaná ako vstupná veličina), alebo napríklad nelinearitou reálneho procesu, táto odchýlka sa prejaví v nepresnosti odhadu stavov. Principiálne rozlišujeme dve poruchy a to výstupnú a

stavovú. Pri modelovaní môžeme uvažovať, že na systém vplýva jedna z nich prípadne aj obe súčasne. Ich správna implementácia do modelu vedie na konzistentnejší odhad stavu.

- Matica E (výstup): je matica popisujúca vplyv poruchy na jednotlivé výstupy zo systému. Určujúce je umiestnenie nenulových prvkov v matici (ktorý výstup je v akom pomere ovplyvnený poruchou), veľkosť prvkov len nepriamoúmerne ovplyvní veľkosť výstupnej poruchy, keďže táto je tiež len odhadovaná. Počet riadkov matice E musí byť zhodný s počtom výstupných veličín. Ak neuvažujeme vplyv poruchy, tak zadáme nulovú maticu.

- Matica F (stavy): má obdobnú funkciu ako matica E s rozdielom, že porucha vplýva na stavy systému.

-Parametre MHE:

- Dĺžka "predikčného"horizontu: udáva "ako ďaleko do minulosti sa má MHE pozerať". Zvyšovanie po určitú hranicu sa prejaví zlepšením odhadu stavov, odporúčaná hodnota je 5 -10 (s vhodne zvolenou periódou vzorkovania), keďže tento parameter má zásadný vplyv na výpočtovú náročnosť algoritmu a tým aj na čas potrebný pre riešenie optimalizačného problému.

- Inicializačný stav x0: reprezentuje aktuálny stav systému v čase keď je inicializovaný pozorovač stavov. Tento parameter sa použije iba v prvej perióde vzorkovania, ďalšie hodnoty si generuje algoritmus MHE sám. Očakáva sa stĺpcový vektor s dĺžkou rovnou počtu stavov. Ak informácia o počiatočnom stave nie je k dispozícii, zadajte približné hodnoty stavov, alebo nulový vektor.

- Váhové (penalizačné) matice:

vzhľadom na formuláciu MHE majú zásadný vplyv na fungovanie pozorovača stavov. Je potrebné myslieť na to, že hodnoty jednotlivých váhových matíc majú hlavne relatívny význam vzhľadom na ostatné váhové matice. Umiestnenie jednotlivých nenulových prvkov vo váhovej matici a ich hodnoty určujú ktorý zo stavov/výstupov/porúch má byť prioritne penalizovaný. Je vhodné použitie diagonálnych matíc, prípadne skalárov. Ak sa nevyžaduje rozlišovať medzi jednotlivými stavmi/výstupmi/poruchami je možné použiť jednotkové matice násobené skalárom.

- Penalizácia $y - \hat{y}$: rozdiel $y - \hat{y}$ (y - výstup z reálneho, pozorovaného systému, $\hat{y}$ - výstup z pozorovača stavov vypočítaný na základe matematického modelu) je hlavným kvalitatívnym ukazovateľom pozorovača stavov. Očakáva sa, že bude zadaná štvorcová matica s rozmerom rovným počtu výstupov, prípadne skalár. Ak je rozdiel medzi hodnotou

výstupnej veličiny zo systému a tou z pozorovača, tak je potrebné zvýšiť hodnotu tjeto váhovej matice.

- Inicializačný stav: je nevyhnutný pre inicializáciu MHE, pričom platí "lepší nepresný ako žiadny". V prípade, že stavy systému nepoznáme, alebo máme len veľmi skreslené údaje o ich hodnotách, tak v tomto prípade je vhodné zvoliť menšie hodnoty penalizačnej matice.

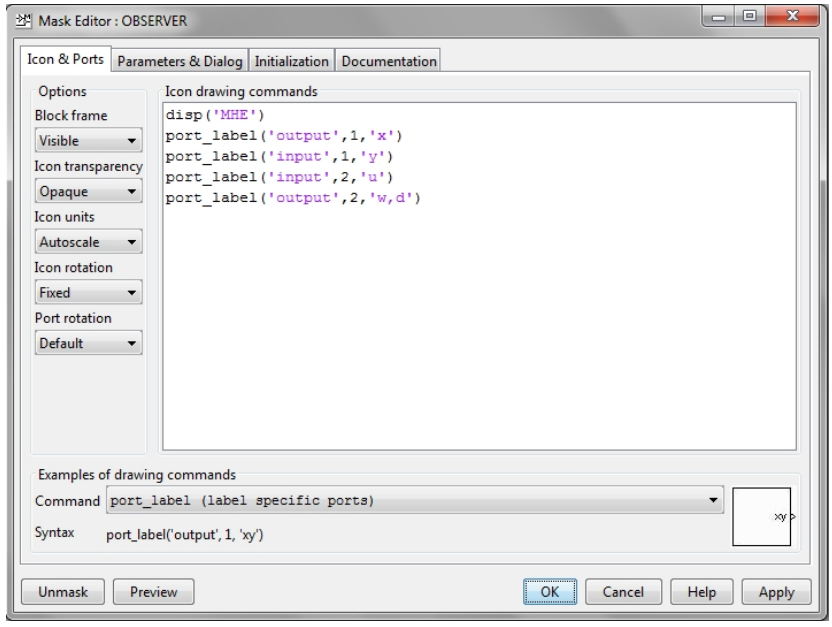
- Výstupná porucha: penalizuje kvadrát výstupnej poruchy (w), preto ak je hodnota výstupnej porucha príliž veľká, tak zvýšte hodnotu tejto váhovej matice.

- Stavová porucha: penalizuje kvadrát stavovej poruchy (d), preto ak je hodnota stavovej porucha príliž veľká, tak zvýšte hodnotu tejto váhovej matice.

- Ohraničenia:

sú hlavnou prednosťou MHE. Pozorovač plne rešpektuje dané ohraničenia a hodnota odhadnutých stavov bude vždy v rámci zadaného intervalu.

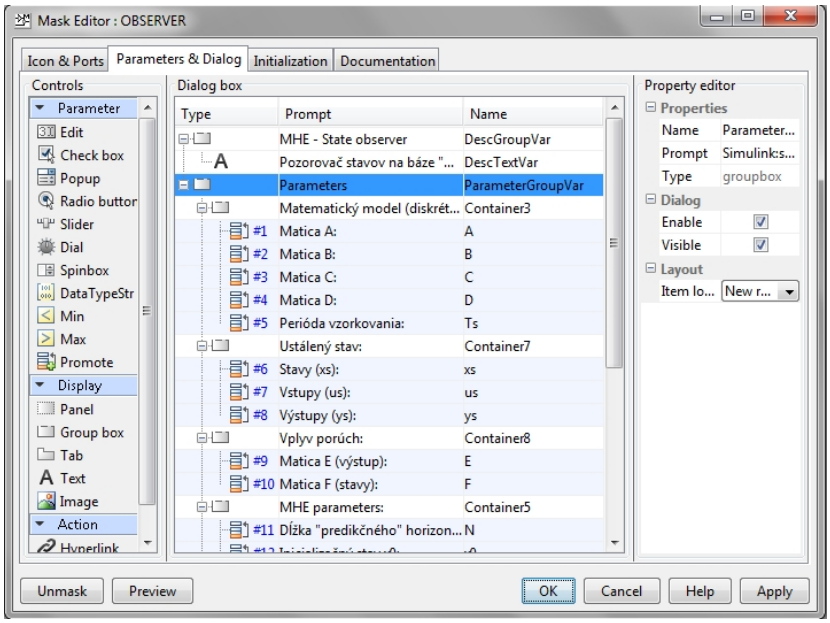
Tvorba masky a samotného bloku prebiehala zhruba nasledovne. V parametroch masky boli zadefinované a nastavené potrebné vstupy a výstupy z bloku podľa Obr. 2.3.3.



Obr. 2.3.3: Nastavenie vstupov a výstupov z masky.

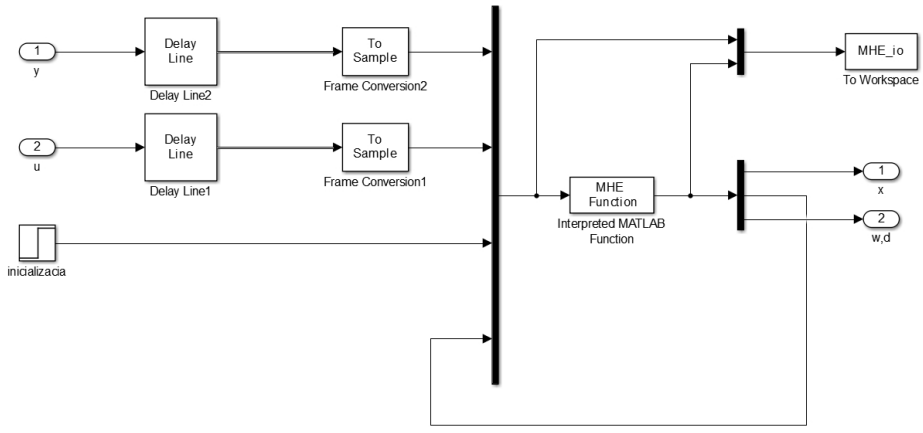
Následne boli nastavené jednotlivé polia pre vkladanie a editáciu konštantných para-

metrov systému a stručný popis bloku. Programovacie, respektíve editačné okno masky je vyobrazené na Obr. 2.3.4.



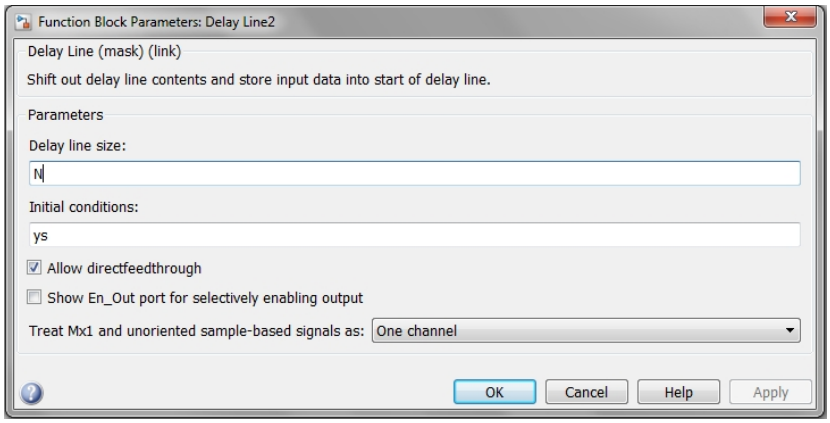
Obr. 2.3.4: Nastavenie parametrov masky.

Pod maskou bloku sa nachádza prvotné spracovanie signálov do vhodnej formy pre výpočtový algoritmus. Schému môžeme vidieť na nasledujúcom obrázku 2.3.5.



Obr. 2.3.5: Pod maskou bloku MHE.

Ak pôjdeme chronologicky, tak najprv hodnoty u a y vstupujú do bloku `Delay Line`, ktorého úlohou je zhromažďovať posledných N hodnôt, kde N je dĺžka horizontu zadaná v maske. Tento blok v každej perióde vzorkovania načíta novú hodnotu, pridá ju do vektora a následne odstráni najstaršiu hodnotu, potom celý nový vektor posunie ďalej, v podstate plní úlohu zásobníka hodnôt. Parametre tohto bloku zobrazuje Obr. 2.3.6.



Obr. 2.3.6: Line Delay.

Ako inicializačné hodnoty sú pre celý vektor nastavené hodnoty v ustálenom stave, ktoré sa po spustení postupne prepisujú novými hodnotami. Označená možnosť "Allow directfeedtrough" spôsobuje, že v aktuálnej perióde vzorkovania posunie tento blok vektor už aj s najaktuálnejšou hodnotou. Defaultné nastavenie bolo bez tohto parametra, čo spôsobovalo posun, respektíve oneskorenie odhadu stavu o jednu periódu vzorkovania, čo malo negatívny vplyv na kvalitu.

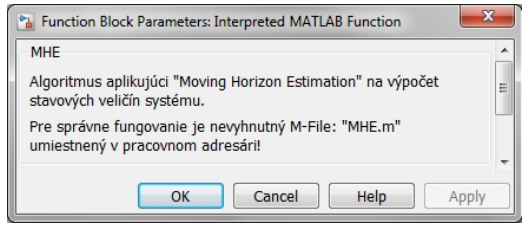
Problémom však je, že tento blok vytvára vektor, ktorý má štruktúru tzv. "frame", ktorý nie je povoleným vstupom do bloku s funkciou, preto tento signál musíme najskôr konvertovať v bloku `Frame Conversion` na štandardný vektor.

Blok s funkciou MHE má iba jeden vstup, pričom počet vstupov nie je možné zvýšiť, preto bol použitý `Mux`, ktorý všetky potrebné vstupné veličiny združí do jedného signálu. Ako vyplýva z obrázka 2.3.5, tak máme štyri vstupy. Prvým je vektor výstupných veličín, druhým vektor vstupov do pozorovaného systému, tretí vstup slúži len ako prepínač, ktorý informuje o prvej perióde vzorkovania a posledným je nový inicializačný stav x_0, ktorý je zároveň výstupom z funkcie MHE.

Funkcia MHE umožňuje opäť iba jeden výstup, preto všetky potrebné výstupné veličiny sú vo vnútri funkcie poskladané do jedného vektora, ktorý je pomocou `Demux`-u opäť

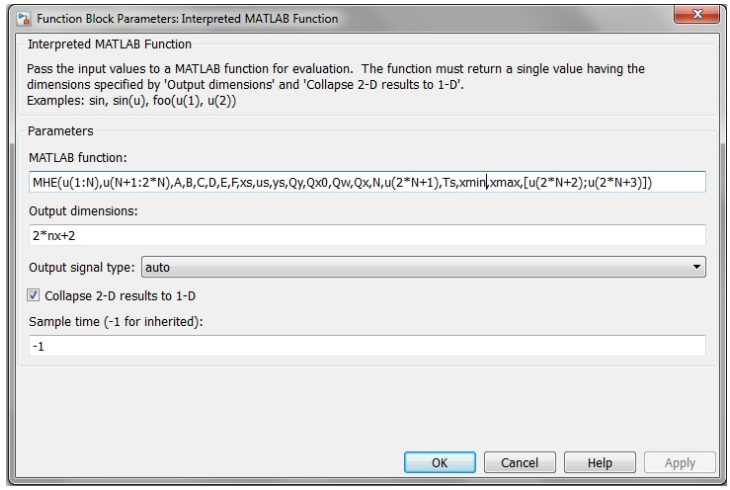
rozseparovaný na jednotlivé zložky. Prvým výstupom sú stavy, druhým nový inicializačný stav, ktorý sa len vracia späť na vstup vrámci masky a posledným sú poruchy.

Pre implementáciu algoritmu MHE bol použitý blok `Interpreted MATLAB Function`, ktorý ako jediný umožňoval použitie zvoleného algoritmu. Jeho maska preberá parametre z nadradenej masky, a preto jeho maska obsahuje len upozornenie, že je nevyhnutné, aby sa súbor s funkciou nachádzal v pracovnom adresári (Obr. 2.3.7).



Obr. 2.3.7: Maska funkcie.

Parametre tejto funkcie (Obr. 2.3.8) sú nastavené tak, aby ako vstupné parametre prevzala parametre z masky a taktiež je v nej rozdelený vektor u, ktorý označuje celý vstupný signál na jednotlivé veličiny, ktoré boli doň združené pomocou `Mux`-u. Veľkosť jednotlivých častí vektora u je variabilná a je daná hlavne dĺžkou predikčného horizontu, preto je zapísaná ako funkcia N, pre možnosť zmeny dĺžky predikčného horizontu.



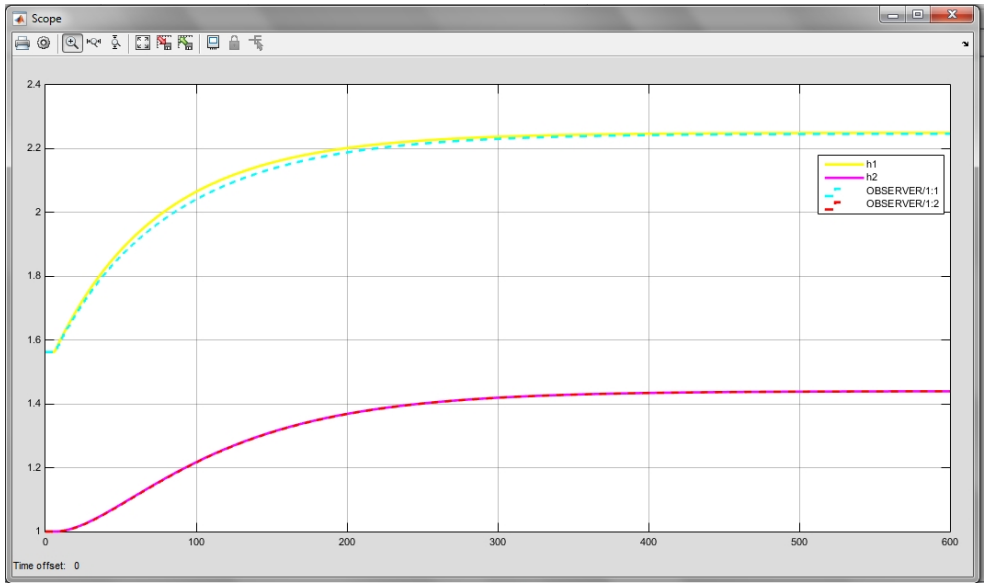
Obr. 2.3.8: Parametre funkcie.

Samotný algoritmus MHE implementovaný v rámci bloku je spolu s blokom dostupný

na internete a voľne stiahnuteľný [1]. Časť premenných je zadefinovaná ako typ "persistent", čo znamená, že hodnota týchto premenných sa zachová do ďalšej periódy vzorkovania. Celý algoritmus je rozdelený do dvoch častí, pričom prvá je inicializačná a je spustená len v prvej perióde vzorkovania. Na identifikáciu prvej periódy vzorkovania slúži `Step`, ktorý je umiestnený v subsystéme a po prvej perióde vzorkovania sa prepne z 0 na hodnotu 1, v algoritme ho nájdeme pod premennou "cas". Druhá časť programu sa vykoná v každej perióde vzorkovania, pričom prvá časť sa vynecháva a preberajú sa len hodnoty premenných typu "persistent", čím dochádza k výraznému zníženiu výpočtovej a tým pádom aj časovej náročnosti behu programu. Časová náročnosť bola jednou z hlavných nepriamych úloh, ktorej bola venovaná nemalá pozornosť vrámci tejto práce.

2.3.2 Overenie funkčnosti bloku

Nasledujúci graf (Obr. 2.3.9) prebratý priamo zo Simulinku prezentuje výkon bloku pozorovača stavov:

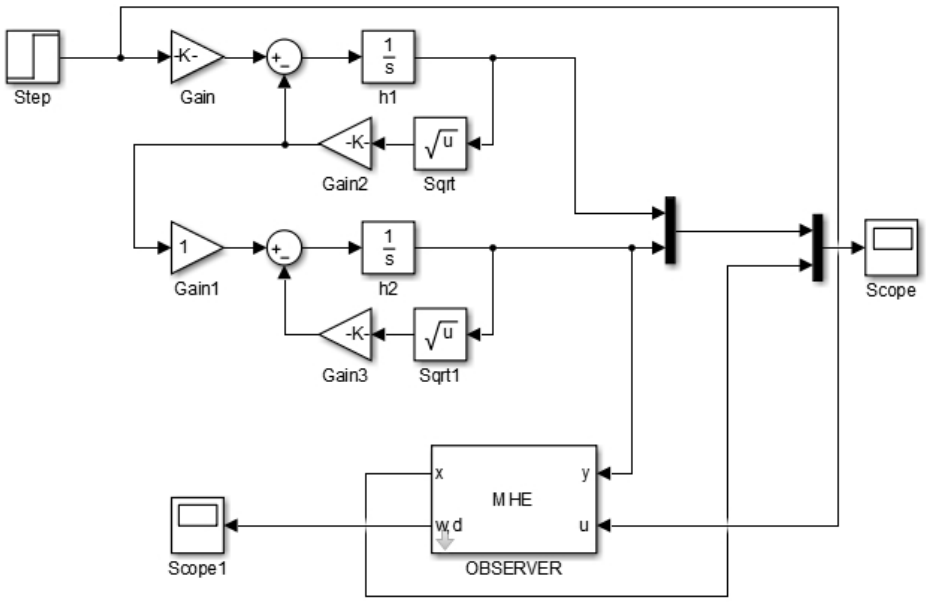


Obr. 2.3.9: Graf výkonu MHE v Simulinku.

Ako je z vyššie uvedeného grafu 2.3.9 zrejmé, tak simulinkový blok pre odhad stavu dosahuje porovnateľnú kvalitu odhadu stavu ako algoritmus implementovaný v matlabe v rámci M-Filu.

[1] https://Simon_Koniar@bitbucket.org/Simon_Koniar/podporne-riadiace-systemy-na-baze-mpc

Pre testovanie bloku MHE bola zostavená nasledovná simulinková schéma (Obr. 2.3.10), ktorá reprezentuje zásobníky kvapalín (ich nelineárny model):



Obr. 2.3.10: Simulinková schéma.

2.3.3 Výpočtová náročnosť

Pre simuláciu s dĺžkou 600 períód vzorkovania (600 sekundová simulácia s periódou vzorkovania 1 sekunda) bol výpočtový čas 135 sekúnd, čo predstavuje v priemere 225 ms na periódu vzorkovania. Výpovedná hodnota tohto údaju je vysoká - tento pozorovač je možné použiť pre riadenie v reálnom čase pri perióde vzorkovania 1 sekunda, s dostatočnou časovou rezervou. V prípade, že by bolo potrebné riadiť systém s rýchlejšou dynamikou sú tu aj ďalšie možnosti znižovania výpočtovej náročnosti ako napríklad zavedenie 1-normy do účelovej funkcie prípadne skrátenie predikčného horizontu, ktorý bol pri testovaní nastavený na hodnotu 10, alebo použitím výkonnejšieho solvera - napr. Gurobi. Samozrejme, výpočtová náročnosť môže aj vzrásť a to v prípade, že bude potrebné aproximovať stavové veličiny zložitejšieho (viacstavového) systému.

Pred optimalizáciou výpočtovej náročnosti trvala simulácia s dĺžkou 100 sekúnd a periódou vzorkovania 3 sekundy (v podstate 34 períód vzorkovania) viac ako 1000 se-

kúnd, čo predstavuje v priemere takmer 30 sekúnd na periódu. Modifikáciou algoritmu a parametrov bloku sa dosiahlo viac ako 100-násobné zníženie výpočtového času.

Vzhľadom na formuláciu MHE (2.1.1) je možné konštatovať, že optimalizačný problém je konvexný, keďže účelová funkcia je kvadratická a ohraničenia lineárne v optimalizačných premenných. Principiálne sa jedná o kvadratické programovanie (QP) s ohraničeniami v tvare rovností a nerovností, pre ktoré je vyvinutá široká škála viac, či menej efektívnych solverov, ktoré dokážu daný optimalizačný problém riešiť. Problém MHE je možné pretransformovať do štandardnej formy pre QP:

$$\min \quad {}^1/_2\, U^\top H U + x_0 F^\top U \tag{2.3.1a}$$

$$\text{s.t.} \quad GU \leq w + E x_0 \tag{2.3.1b}$$

kde optimalizovanými premennými sú $U = [u_0^\top, \cdots, u_{N-1}^\top]^\top$. Tento problém sa typicky rieši buď použitím active-set metódy, prípadne pomocou bariérových metód. Tieto metódy sú implementované vo voľne šíriteľných ako i komerčných softvérových balíkoch, ako sú napr. GUROBI, CPLEX, MOSEK, CLP, atď.

V rámci tejto práce bol použitý YALMIP, ktorý dokáže efektívne pretransformovať daný optimalizačný problém do formy vhodnej pre ten-ktorý solver (v tomto prípade Quadprog). Jednou z možností ako znížiť výpočtovú náročnosť by bolo vynechanie YALMIP-u, čo by však vyžadovalo prevedenie formulácie do čisto maticovej formy, ktorú požaduje solver a tým by sa stratila časť flexibility a univerzálnosti, ktorý poskytuje vytvorený blok.

2.4 Zhrnutie

Ďalším z rodiny podporných riadiacich systémov, ktorý vznikol na báze MPC je Moving horizon estimation. MHE predstavuje pozorovač stavov, ktorý na základe riešenia optimalizačného problému dokáže poskytnúť informácie o vývoji stavových veličín pozorovaného systému. Tomuto konceptu bola v rámci práce venovaná väčšia pozornosť, keďže sa jedná o menej známu metódu aproximácie stavu, no zároveň disponujúcu výrazným potenciálom v podobe ľahko implementovateľných ohraničení. Vytvorený algoritmus MHE bol testovaný na lineárnom aj nelineárnom modeli sústavy zásobníkov kvapalín a zároveň bol jeho výkon porovnaný s bežne používeným Luenbergerovým pozorovačom, ako aj rôznymi verziami Kalmanovho filtra. Porovnanie výkonu bolo realizované jednak pri štandardných simulačných podmienkach, ale testovala sa aj odolnosť na vplyv šumu a výkon pri ohraničených hodnotách procesných veličín. Na základe získaných výsledkov je možné konštatovať, že MHE je plnohodnotnou alternatívou k zaužívaným pozorovačom stavu. Do budúcnosti je potrebné ešte ďalej pracovať na znížení jeho výpočtovej náročnosti a spôsoboch ladenia váhových matíc. Aj v tomto prípade bol vytvorený blok do Simulink-ovej knižnice, ktorý umožňuje pomerne jednoduchú implementáciu, ktorá zahŕňa pripojenie signálov (u a y) a zadanie konštantných parametrov vrátane modelu do masky bloku. Blok okrem aproximácie stavu vracia aj odhad hodnôt porúch, čo uľahčuje ladenie váhových matíc, ale taktiež ich možno použiť ako vstup napr. do MPC. V rámci tejto kapitoly bol bližšie popísaný aj postup tvorby masky, ako aj celého bloku v simulinkovom prostredí.

Kapitola 3

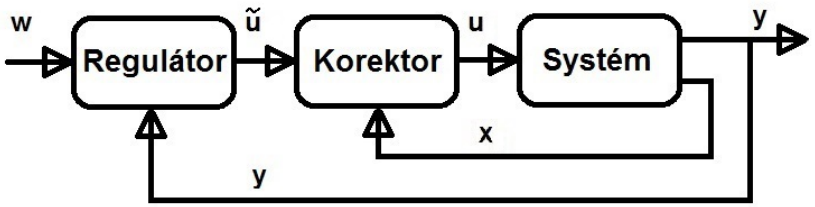
Korektor

Existujú aplikácie, pri ktorých je zaužívaná určitá forma a spôsob riadenia, či už sa jedná o manuálne alebo automatické riadenie a z určitých dôvodov je problematické túto štruktúru modifikovať alebo do nej zasahovať. Avšak aj overené a zaužívané riadiace systémy môžu zlyhať, a práve v tých "konzervatívnych" aplikáciách ako je napríklad medicína, alebo vojenský sektor majú zlyhania veľmi vysokú cenu. Z tohto, ale aj množstva iných dôvodov a podnetov sme sa rozhodli vytvoriť podporný riadiaci systém - "Korektor" na báze MPC, ktorý by zabezpečoval primárnu úlohu riadenia - stabilitu a bezpečnú prevádzku riadeného systému a to za predpokladu, že budú v maximálnej možnej miere rešpektované akčné zásahy pôvodného regulátora. Korektor bude pasívnym prvkom riadiaceho obvodu do momentu, kým nenastane, respektíve aktuálny stav a riadenie nebude viesť na nebezpečnú situáciu. Takýto prístup by mohol otvoriť cestu pokročilým metódam riadenia (MPC) do ďalších aplikácii, v ktorých by výrazným spôsobom pomohli zvýšiť bezpečnosť, alebo sekundárne zlepšiť kvalitu riadenia. Eliminovali by sa pochybenia a havárie zapríčinené zlyhaním ľudského faktoru, výpadkom komunikácie, alebo úmyselným narušením prípadne zásahom do riadiaceho systému, keďže sa predpokladá, že korektor bude inštalovaný v bezprostrednej blízkosti riadeného systému.

Ďalším konceptom rozvíjaným v rámci tejto práce sú invariantné množiny, ktoré sú zároveň implementované v korektore ako ohraničenia optimalizačného problému. Význam formulácie ohraničení v tvare invariantnej množiny spočíva v tom, že ak sa všetky ohraničenia MPC (stavová a výstupná rovnica, ohraničenia na jednotlivé veličiny) pretransformujú do jednej invariantnej množiny, potom je možné znížiť predikčný horizont MPC na jednu periódu vzorkovania, čo však vzhľadom na ohraničenie (invariantnú množinu) je ekvi-

valentné nekonečne dlhému predikčnému horizontu. Jednoducho povedané, invariantné množiny umožňujú zvýšiť kvalitu riadenia tým, že predĺžia predikčný horizont a to za súčasného zníženia výpočtovej náročnosti, keďže redukujú počet optimalizovaných premenných. Jediným problémom ostáva ich konštrukcia, ktorá je pri mnohorozmerných systémoch pomerne zložitá. V práci sa pokúsime priblížiť túto problematiku a preskúmať rôzne prístupy a postupy pri implementácii invariantných množín.

Praktické overenie fungovania či už korektora, alebo invariantných množín je v práci prezentované simulačne na matematickom modeli dvojitého integrátora, respektíve na modeli kvadrokoptéry. Jedným z cieľov práce bolo vyvinúť univerzálny blok pre Simulink, ktorý by pracoval na báze korektora a bol by ľahko implementovateľný pre ľubovolný systém.



Obr. 3.1: Schéma zapojenia Korektora v URO s regulátorom.

Korektor v našom ponímaní predstavuje prvok uzavretého regulačného obvodu (Obr. 3.1) korigujúci akčné zásahy regulátora, ktorý je pasívny do momentu, kedy akčný zásah regulátora vedie na porušenie ohraničení. Primárnou úlohou korektora je zabezpečenie stability URO pri súčasnom rešpektovaní používateľom definovaných ohraničení na jednotlivé procesné veličiny. Samotný algoritmus je založený na báze MPC so štandardnými ohraničeniami a modifikovanou účelovou funkciou.

3.1 Formulácia

Korektor je derivátom prediktívneho riadenia (MPC), preto je jeho formulácia do značnej miery podobná formulácii MPC. Principiálne sa jedná o optimalizačný problém s ohraničeniami v tvare rovností a nerovností, pričom optimalizovanou premennou je u_k - riadiaca veličina po korekcii v každej perióde vzorkovania na celom predikčnom horizonte N:

$$\min \sum_{k=0}^{N-1} \|Q_{\mathrm{u}}(u_k - \tilde{u})\|_p \tag{3.1.1a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{3.1.1b}$$

$$y_k = Cx_k + Du_k, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{3.1.1c}$$

$$x_0 = x(t) \tag{3.1.1d}$$

$$x_k \in \mathcal{X}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{3.1.1e}$$

$$u_k \in \mathcal{U}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{3.1.1f}$$

$$y_k \in \mathcal{Y}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{3.1.1g}$$

V účelovej funkcii (3.1.1a) penalizujeme rozdiel medzi u_k a vstupom do korektora $\tilde{u}$, ktorý reprezentuje akčný zásah nadradeného regulátora. Minimalizáciou tohto rozdielu sa zabezpečí, aby akčný zásah na výstupe z korektora v maximálnej možnej miere kopíroval akčný zásah na vstupe a tým si zachovával svoju pasivitu v rámci regulačného obvodu. Tomuto členu je v účelovej funkcii priradená váha pomocou váhovej matice Q_{u}, čo však nie je nevyhnutné keďže účelová funkcia štandardnej formulácie obsahuje iba jeden člen. No na druhej strane, ak uvažujeme viac vstupov, ktoré chceme medzi sebou rozlíšiť (napríklad na základe ceny), alebo ak by sme chceli zaviesť mäkké ohraničenia, prípadne iným spôsobom modifikovať formuláciu, tak potom získa svoje opodstatnenie. Podľa konkrétnej aplikácie sa zvolí jedna, dva, alebo nekonečno norma, ktorá určuje penalizačnú funkciu rozdielu v účelovej funkcii. Najčastejšie používanou je 2-norma, respektíve kvadrát 2-normy, ktorý vedie na kvadratický optimalizačný problém, keďže ohraničenia sú lineárne. Riešenie kvadratických optimalizačných problémov je konvexný problém a existuje viacero solverov, ktoré dokážu takéto problémy pomerne efektívne riešiť (napr. Quadprog).

Ohraničenia sú v podstate rovnaké ako v MPC: rovnica (3.1.1b) reprezentuje stavovú rovnicu implementujúcu model systému, na základe ktorého sa predikujú budúce stavy systému v rámci predikčného horizontu. V prípade, že je v rámci uzavretého regulačného obvodu implementované výstupné riadenie, alebo je z iného dôvodu praktické pracovať s výstupmi a nie len so stavmi systému, tak potom je k dispozícii výstupná rovnica (3.1.1c), ktorej výsledkom sú hodnoty výstupných veličín (y_k) v danej perióde vzorkovania. Ohraničenia (3.1.1e - 3.1.1g) sú de facto ohraničenia v tvare nerovností, keďže $\mathcal{X}$, $\mathcal{U}$ a $\mathcal{Y}$ reprezentujú množiny prípustných hodnôt pre jednotlivé procesné veličiny. Je vhodné a odporúčané ohraničenia na stavové a výstupné veličiny zadefinovať ako mäkké ohraničenia, čo zabezpečí riešiteľnosť opt. problému.

3.2 Princíp fungovania

Základné princípy boli popísané aj v predchádzajúcich kapitolách, kde bolo prezentované, že vstupmi do korektora sú akčné zásahy regulátora, prípadne operátora a taktiež aktuálne hodnoty stavových veličín riadeného systému, no nebolo poznamenané, že stavy systému nemusia byť vždy merateľné a preto je potrebné ich odhadovať pomocou pozorovača stavov, čo môže do istej miery skomplikovať implementáciu prezentovaného korektora. Problematike odhadu stavu sa bližšie venuje kapitola 2, ktorá implementuje pozorovač stavu na báze MHE (Moving Horizon Estimation), no existuje viacero prístupov, z ktorých najzaužívanejším je asi kalmanov filter, ktorý dokáže túto prekážku zdolať.

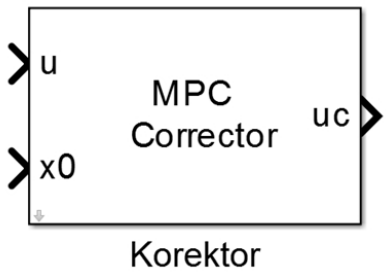
Po vstupe akčného zásahu regulátora do Korektora sa primárne vyhodnotí, či tento akčný zásah spĺňa ohraničenia na akčný zásah a s ohľadom na aktuálny stav systému sa na základe matematického modelu vypočítajú budúce stavy systému na celom predikčnom horizonte. Hodnoty týchto stavov a na základe nich odvodených výstupov systému sa taktiež vyhodnotia, či sú v rámci intervalov definovaných v ohraničeniach. Dôležité je poznamenať, že pri predikcii predpokladáme konštantný akčný zásah na celom predikčnom horizonte, čo nemusí korešpondovať s realitou, no na druhej strane nie je možné zistiť budúce akčné zásahy regulátora, ak uvažujeme ľubovoľnú štandardnú aplikáciu. Pri niektorých systémoch by bolo možno presnejšie a smerodajnejšie, ak by sme sa nepozerali len na samotnú hodnotu akčného zásahu, ale aj na jej deriváciu, respektíve diferenciu, ktorá zachytáva trend vývoja v čase. Ak Korektor vyhodnotí, že akčný zásah regulátora spĺňa ohraničenia, tak nezasahuje a pôvodný akčný zásah pustí ďalej - do procesu, respektíve akčného člena. Ak sa však ukáže, že akčný zásah regulátora vedie na porušenie ohraničení, tak potom Korektor daný akčný zásah modifikuje tak, aby splnil ohraničenia, ale zároveň aby sa čo možno najmenej odlišoval od pôvodného akčného zásahu. Aj keď Korektor optimalizuje akčný zásah na celom predikčnom horizonte, tak rovnako ako MPC, aplikuje iba prvý akčný zásah.

Ak uvažujeme v URO diskrétne riadenie, tak potom sa zaradením korektora do regulačného obvodu veľa nezmení, až na opozdenie riadenia o jednu periódu vzorkovania a prípadné predĺženie periódy vzorkovania na čas, ktorý je postačujúci pre riešenie optimalizačného problému v rámci korektora. V prípade, že zaradíme korektor do spojitého regulačného obvodu, potom je potrebné počítať s tým, že korektor spôsobí diskretizáciu akčného zásahu, čo sa môže prejaviť negatívne na kvalite riadenia a to hlavne pri riadení mnohorozmerných systémov a dlhom predikčnom horizonte, ktorý spôsobí vyššiu výpočtovú náročnosť optimalizačného problému. Znižovaniu zložitosti optimalizačného problému je venovaná aj kapitola 3.4 Invariantné množiny, ktorá opisuje jeden zo spôsobov

znižovania zložitosti a tým aj času potrebného na výpočet.

3.3 Blok pre Simulink

Praktickým výstupom tejto práce je univerzálny blok implementujúci korektor v rámci prostredia Simulink pre riadenie ľubovolného systému v kombinácii s ľubovolným regulátorom.

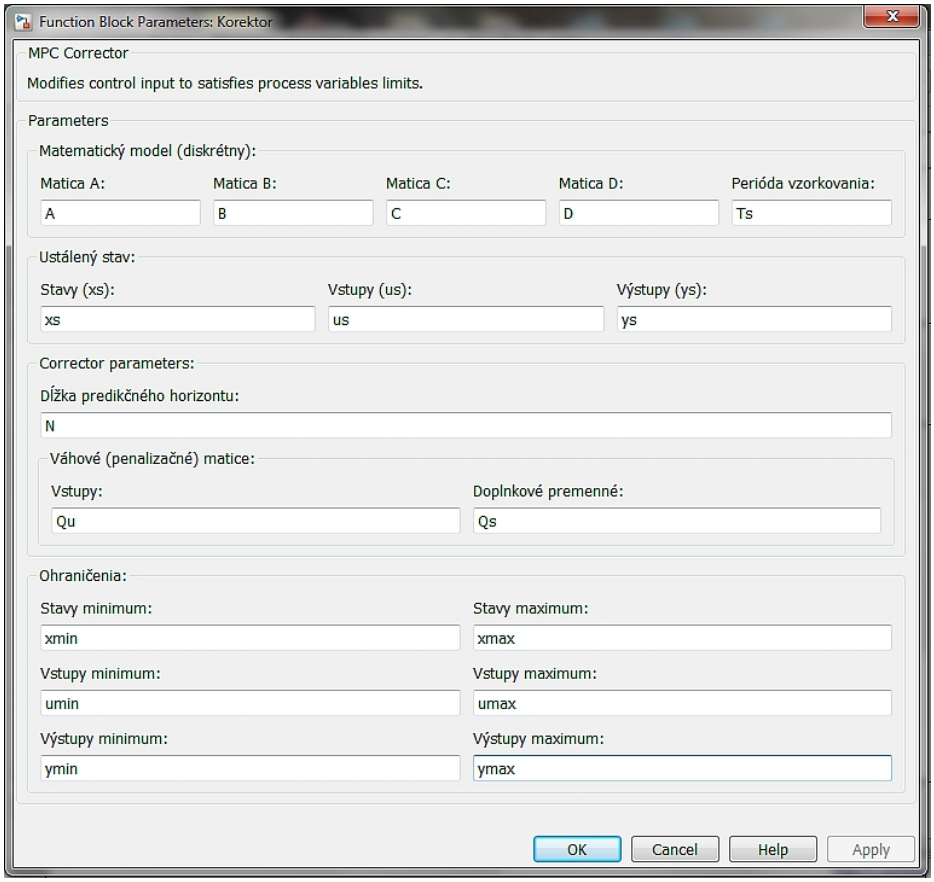


Obr. 3.3.1: Blok korektora v Simulinku.

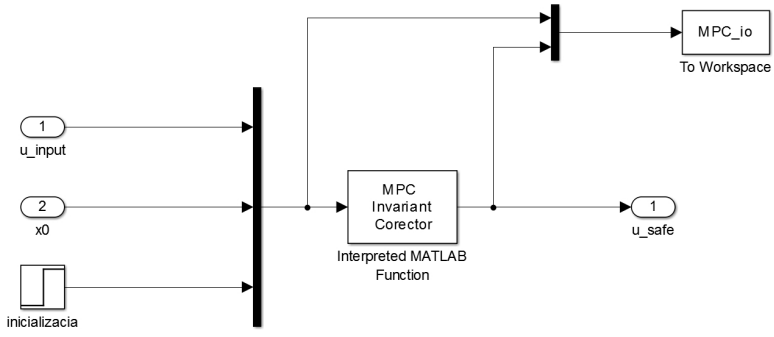
Interface bol navrhnutý tak, aby bol pre užívateľa čo možno najjednoduchší a najintuitívnejší a tým mu umožnil implementovať korektor aj bez znalostí o samotnom algoritme. Pre nezainteresovaného používateľa sa môže korektor javiť ako "inteligentný" blok saturácie, do ktorého zadá požadované ohraničenia, matice stavového modelu a pripojí potrebné signály. Bežný používateľ sa dostane na úroveň masky, ktorú je potrebné ako jedinú vyplniť pre správne fungovanie korektora, bez zásahov do nižších úrovní.

Ak si však konkrétna aplikácia vyžaduje modifikáciu algoritmu, ten je taktiež prístupný, keďže korektor je implementovaný ako `Interpreted Matlab Function`"(Obr. 3.3.3), takže sa dá pomerne jednoducho upraviť v rámci daného M-Filu. Štandardná formulácia algoritmu, ktorá už zároveň implementuje ohraničenia v tvare invariantnej množiny je spolu s blokom dostupná a voľne stiahnuteľná na internete [1].

[1] https://Simon_Koniar@bitbucket.org/Simon_Koniar/podporne-riadiace-systemy-na-baze-mpc



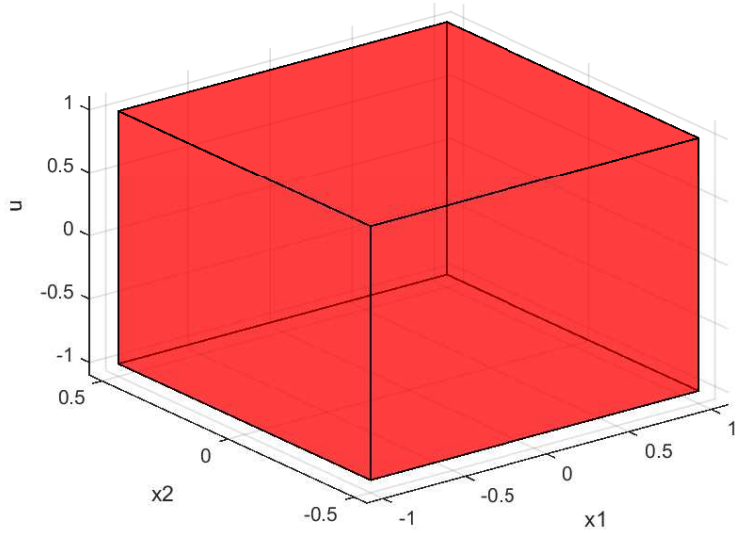
Obr. 3.3.2: Maska korektora.



Obr. 3.3.3: Subsystém korektora.

3.4 Invariantné množiny

Invariantné množiny predstavujú nástroj na zníženie zložitosti optimalizačného problému za predpokladu, že uvažujeme optimalizačný problém s ohraničeniami, ktoré sú reprezentované práve invariantnou množnou. Invariantnú množinu si môžeme predstaviť ako polytop v n-rozmernom priestore, ktorý zahŕňa stavy systému zlúčiteľné s ohraničeniami. Uvažujme jednoduchý príklad: majme systém s dvomi stavmi, pričom jeden z nich reprezentuje polohu a druhý rýchlosť, riadením vieme ovplyvňovať rýchlosť a tým pádom aj polohu. Ďalej uvažujme ohraničenia na oba stavy ako aj na akčný zásah. Ak predpokladáme, že veličiny sú ohraničené zdola ako aj zhora, potom ich môžeme zapísať ako šesť nerovností, ktoré rozdeľujú priestor na polpriestory. Ak vykreslíme prienik týchto polpriestorov do jedného grafu (Obr. 3.4.1), tak dostávame polytop (v tomto prípade kváder).



Obr. 3.4.1: Polytop definovaný ohraničeniami.

Práve na tomto fakte je založená teória, že každú množinu môžeme zapísať ako n nerovností prípadne ako jednu nerovnosť, kde ľavá strana je reprezentovaná maticou a pravou stranou je vektor, čo zjednoduší zápis do tvaru (3.4.1):

$$Hz \leq h \tag{3.4.1}$$

3.4.1 Konštrukcia invariantných množín

Uvažujme, že systém sa aktuálne nachádza v stave, ktorý je definovaný bodom, ktorý patrí do invariantnej množiny (Obr. 3.4.1) (daný stav spĺňa ohraničenia) a zároveň systém má určitú dynamiku popísanú matematickým modelom. Takže ak vykonáme akčný zásah, alebo necháme plynúť čas, tak stav systému sa zmení a bude definovaný iným bodom, ktorý môže byť buď vo vnútri, alebo mimo množiny. Ak si spomenieme na konštrukciu množiny, z ktorej vyplýva, že vnútro množiny zodpovedá stavom zlúčiteľným s ohraničeniami, tak potom môžeme konštatovať: ak sa nový stav nachádza v rámci množiny, tak potom sú ohraničenia splnené aj v tejto perióde vzorkovania. Avšak ak aplikujeme do systému iný akčný zásah, stavy sa môžu dostať mimo množiny. Hovorili sme, že akčný zásah má tiež svoje pôvodné ohraničenia a tým pádom mu zodpovedá jedna dimenzia v priestore definovaná nad rovinou reprezentujúcou dva stavy systému. Preto môžeme povedať, že ak sa nachádzame v bode definovanom súradnicami $[x_1; x_2]$, ktoré zodpovedajú stavu systému, tak nad týmto bodom je definovaný interval akčných zásahov, ktoré spĺňajú pôvodné ohraničenie na akčné zásahy. Ako sme už uvažovali, niektoré z týchto akčných zásahov dovedú systém mimo množiny, preto tieto akčné zásahy pre daný stav systému vylúčime z množiny, čím získame garanciu, že ak je systém v stave $[x_1; x_2]$ a aplikujeme akčný zásah z množiny definovanej nad daným stavom, potom v ďalšej perióde vzorkovania stav ostane v množine - ohraničenia budú splnené. Podobne orežeme akčné zásahy pre každý stav z množiny, čo zabezpečí, že ak sa stav systému nachádza v množine a realizujeme akčný zásah z množiny, potom sa bude stav systému v nasledujúcej perióde vzorkovania nachádzať taktiež vo vnútri množiny, čiže aj pôvodné ohraničenia budú splnené, keďže množina prípustných vstupov je podmnožinou pôvodnej, ktorá bola definovaná pôvodnými ohraničeniami. Takúto množinu (Ω) nazývame pozitívne-invariantnou množinou, pretože pre daný dynamický systém platí, že $\forall x_0 \in \Omega$, z ktorého vychádza trajektória stavu, zotrváva naďalej v množine Ω, t.j. $x_k \in \Omega$ pre $k = 1, 2, ..., \infty$. Túto podmienku však môže spĺňať viacero množín a preto je cieľom nájsť tzv. maximálnu pozitívne invariantnú množinu, ktorá zahŕňa všetky pozitívne-invariantné množiny.

V praxi využívame hlavne dve triedy invariantných množín - polyedrické a elipsoidálne a to z toho dôvodu, že tieto množiny zaraďujeme ku konvexným množinám, čo má významný vplyv na zložitosť optimalizačného problému, v ktorom sú implementované. V tejto práci sa budeme zaoberať iba polyedrickými množinami, ktoré dobre popisujú reálne fyzikálne ohraničenia, ktoré bývajú prirodzene definované práve formou nerovností. Výhodou polyedrických množín je, že dokážu lepšie aproximovať (vymedziť) stavový priestor zodpovedajúci invariantnej množine v porovnaní s elipsoidmi, no na druhej strane

majú aj vyššiu zložitosť, čo sa týka reprezentácie. Vo všeobecnosti polyéder chápeme ako n-rozmerné teleso, respektíve množinu. Ak je táto množina uzatvorená (čo je pri invariantných množinách pozitívna vlastnosť), potom nazývame túto množinu polytopom. Polytop môže vzniknúť principiálne dvoma spôsobmi a to: ako prienik konečného počtu polpriestorov, čím získame tzv. $\mathcal{H}$-polytop, alebo ako konvexná obálka konečného počtu bodov, kedy hovoríme o $\mathcal{V}$-polytope. Z uvedeného vyplývajú aj spôsoby reprezentácie polyédrov, ktoré môžu byť v rovinnej forme $\mathcal{H}$-reprezentácia:

$$\mathcal{H} = \{x \in \mathbb{R}^n : Fx \leq g\} \tag{3.4.2}$$

alebo vrchlovej $\mathcal{V}$-reprezentácia:

$$\mathcal{V} = \{x \mid x = Vu, \sum_{i}^{s} u_i = 1, u \geq 0\} \tag{3.4.3}$$

kde V je matica, ktorej jednotlivé stĺpce sú vrcholy polyédra.

Najväčšou praktickou prekážkou pri implementácii invariantných množín je pomerne vysoká náročnosť ich konštrukcie. V tomto projekte sme pre výpočet invariantných množín použili nasledovný zjednodušene popísaný postup:

1. Inicializujeme množinu pôvodnými ohraničeniami:

$$C^{(0)} = \mathcal{X} \tag{3.4.4}$$

2. Množinu v ďalšom kroku získame ako:

$$C^{(1)} = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U}, Ax + Bu \in C^{(0)}\} \tag{3.4.5}$$

3. Vo všeobecnosti v k-tom kroku:

$$C^{(k+1)} = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U}, Ax + Bu \in C^{(k)}\} \tag{3.4.6}$$

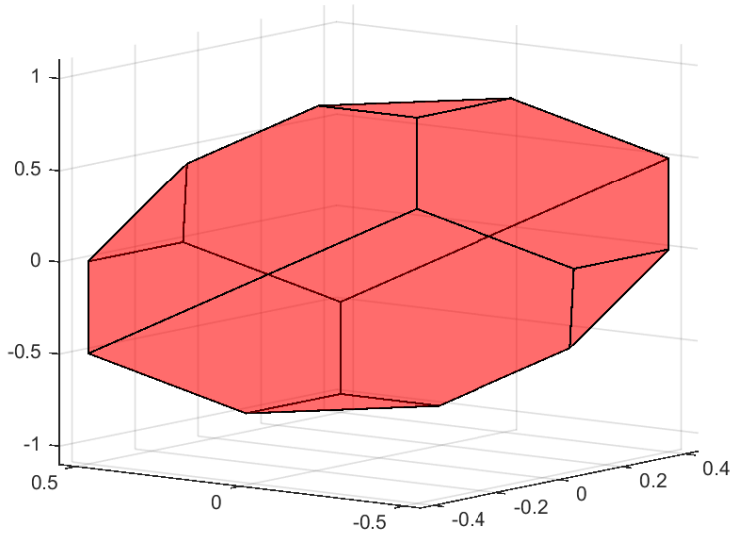
4. Iterácia sa ukončí, keď získame množinu, pre ktorú platí:

$$C^{(k+1)} = C^{(k)} \tag{3.4.7}$$

A túto množinu môžeme prehlásiť za invariantnú množinu:

$$C_\infty = C^{(k)} \tag{3.4.8}$$

Ako príklad môžeme uviesť invariantnú množinu dvojitého integrátora (Obr. 3.4.2), krotý bol základom aj pre simulácie v nasledujúcej časti práce.



Obr. 3.4.2: Invariantná množina.

3.4.2 Využitie v riadení a MPC

Okrem toho, že invariantné množiny nahrádzajú ohraničenia v tvare nerovnosti, tak predstavujú aj jeden z najnovších prístupov garancie stability MPC pri riadení systémov s obmedzeniami. Garancia stability vychádza v podstate zo samotnej definície invariantnej množiny, na základe ktorej môžeme tvrdiť, že ak je počiatočný stav systému prvkom invariantnej množiny, potom sa bude celá trajektória stavov nachádzať vo vnútri množiny, ktorá je uzatvorená a tým pádom stavy systému môžu nadobúdať len konečné hodnoty. Z uvedeného vyplýva významný fakt: stačí posúdiť, či je počiatočný stav prvkom množiny, čo nám zároveň poskytne informáciu o celom budúcom vývoji stavov. Pre MPC to znamená, že bude postačujúci predikčný horizont s dĺžkou jednej periódy vzorkovania, čo však bude zodpovedať nekonečne dlhému predikčnému horizontu, za predpokladu, že implementujeme ohraničenia vo forme invariantnej množiny do formulácie MPC. Táto modifikácia výrazne zjednoduší daný optimalizačný problém, keďže sa zníži počet optimalizovaných premenných, no na druhej strane ak uvažujeme tvorbu invariantnej množiny ako súčasť MPC, tak to spôsobí zvýšenie výpočtovej náročnosti v inicializačnej fáze, úmerné zložitosti

konštruovanej invariantnej množiny. Ak predpokladáme, že daná invariantná množina zodpovedá maximálnej pozitívne invariantnej množine a zároveň jej zložitosť je primeraná, tak okrem garancie stability je vysoký predpoklad zvýšenia kvality riadenia zapríčinený nekonečným predikčným horizontom daným invariantnou množinou. Formuláciu Korektora s ohraničeniami v tvare invariantnej množiny môžeme zapísať napríklad takto:

$$\begin{aligned} \min \quad & \|Q_{\mathrm{u}}(u_0 - \tilde{u})\|_p && \text{(3.4.9a)} \\ \text{s.t.} \quad & x_1 = Ax_0 + Bu_0 && \text{(3.4.9b)} \\ & x_0 = x(t) && \text{(3.4.9c)} \\ & \{x_1, u_0\} \in \mathcal{S} && \text{(3.4.9d)} \end{aligned}$$

kde $\mathcal{S}$ reprezentuje práve invariantnú množinu a z tohto dôvodu je aj pri predikčnom horizonte $N = 1$ optimalizačný problém (3.4.9) ekvivalentný k pôvodnému optimalizačnému problému (3.1.1). Zníženie predikčného horizontu má veľmi priaznivý vplyv na výpočtovú náročnosť, keďže s jeho skracovaním sa znižuje počet optimalizovaných premenných a tým dochádza k zrýchleniu jednotlivých elementárnych maticových operácií pri hľadaní riešenia.

Za predpokladu, že invariantná množina $\mathcal{S}$ je reprezentovaná polyhedronom, ktorý je možné definovať konečným počtom nerovností, ohraničenia si zachovajú lineárny charakter. Typ daného optimalizačného problému závisí tým pádom od typu normy v účelovej funkcii. Ak uvažujeme kvadrát 2-normy získame QP (kvadratické programovanie), prípadne LP (lineárny program) pri použití jedna, alebo nekonečno normy. V oboch prípadoch sa jedná o konvexné optimalizačné problémy, ktoré je možné efektívne riešiť pomocou množstva dostupných solverov (Quadprog, Linprog, GUROBI, atď.) a to s garanciou nájdenia globálneho optima.

3.5 Simulačné overenie

Cieľom tejto kapitoly je demonštrovať funkciu korektora na konkrétnom praktickom príklade. Korektor bol však konštruovaný univerzálne, aby dokázal pokryť široké spektrum aplikácii, preto je pomerne jednoduché zmeniť parametre a riadiť iný systém.

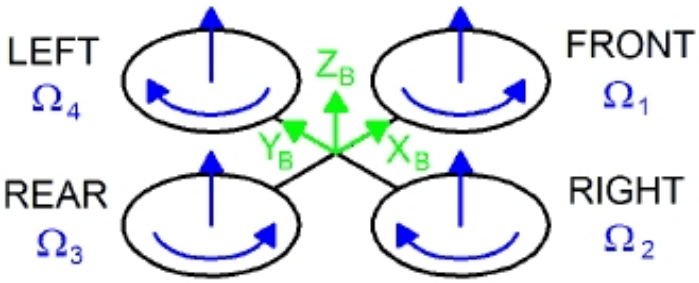
Verifikácia funkcionality bola overená implementáciou Korektora pri riadení nestabilného systému - kvadrokoptéry, konkrétne bol využitý model AR.Drone 2.0. Jedným z dôvodov výberu práve tohto hardvéru je fakt, že predmetné technológie sú na výraznom vzostupe a neustále sa rozširuje oblasť ich využitia, no na druhej strane ich riadenie nie je

elementárne, keďže sa jedná o prirodzene nestabilný systém s vysokým počtom stupňov voľnosti. Principiálne by sme mohli spotrebiteľský trh rozčleniť na dva hlavné segmenty: hobby a profesionálny. V hobby segmente môžeme očakávať veľké množstvo používateľov s nízkymi zručnosťami, ktorým drony slúžia hlavne pre zábavu. Získavanie počiatočných zručností pre ovládanie drona sa zväčša nezaobíde bez kolízií a škôd na hardvéri a práve z tohto dôvodu by zrejme používatelia ocenili funkciu Korektora, ktorý by dokázal predísť takýmto nežiadúcim situáciám a neskúseného operátora by posunul bližšie k profesionálovi. Čo sa týka profesionálnych aplikácií kvadrokoptér, tak tie sa vo väčšine prípadov práve len rozvíjajú, no už aj teraz môžeme sledovať nasadenie dronov napríklad pri zhotovovaní fotografií, skenovaní, mapovaní, doručovaní, alebo prieskume a to aj pre armádne účely. Pre profi segment je charakteristická vysoká úroveň schopností operátora, no na druhej strane tiež vysoká cena hardvéru, preto aj malé zaváhanie alebo nepozornosť môže viesť k vysokým škodám, ktorým by mohol Korektor predísť. Konkrétne AR.Drone 2.0 od spoločnosti Parrot bol pre účely testovania Korektora vybraný okrem iného preto, že umožňuje modifikovať softvér, ktorý výrobca poskytuje vo forme SDK (Piskorski, 2011).

3.5.1 Model - AR.Drone 2.0

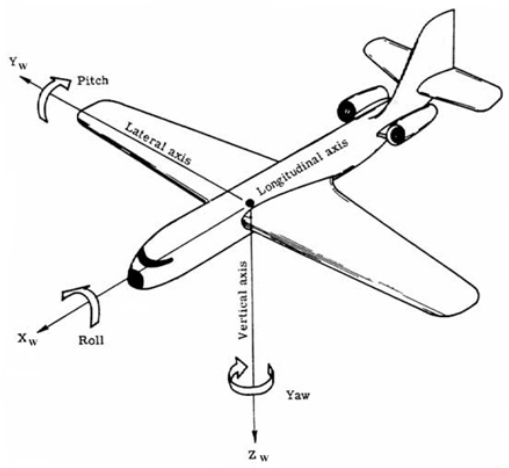
Obr. 3.5.1: AR.Drone 2.0.

AR.Drone 2.0 sa zaraďuje do kategórie UAV (Unmanned aerial vehicle) - bezpilotných lietadiel. Dron je poháňaný a zároveň riadený pomocou štyroch vrtúľ (rotorov), ktorých vzájomnými výkonovými pomermi je možné dron stabilizovať, alebo koordinovane ovládať. Každý pár protiľahlých rotorov sa otáča rovnakým smerom, ako to zobrazené na diagrame 3.5.2.



Obr. 3.5.2: Smer otáčania rotorov.

Pre ovládanie sú kľúčové tri rôzne uhly, ktoré udávajú jednotlivé odklony od referenčnej polohy a ich zmenou sa docieli pohyb drona: Roll(), Pitch() a Yaw(). Pitch ovplyvňuje pohyb dorna vpred a vzad, Roll charakterizuje bočný náklon, ktorého dôsledkom je pohyb vľavo respektíve vpravo. Posledný uhol Yaw udáva otočenie okolo vlastnej vertikálnej osi voči východzej polohe. Parameter ovplyvňujúci letovú výšku sa nazýva Throttle(). Prehľadnú schému s popisom jednotlivých uhlov môžete vidieť na nasledujúcom obrázku 3.5.3:

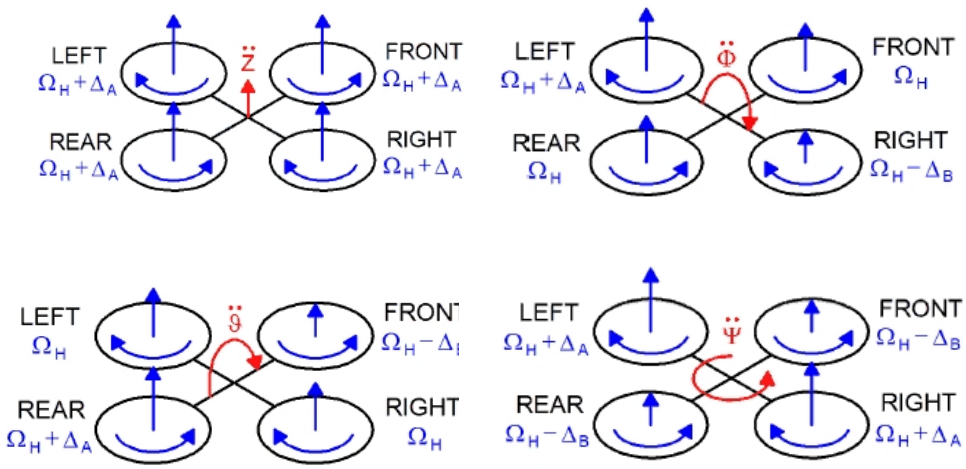


Obr. 3.5.3: Popis uhlov.

Architektúra riadenia je rozdelená na dve principiálne vrstvy: vonkajšiu a vnútornú. Vonkajšia vrstva pracuje práve s vyššie popísanými uhlami, ktorých hodnoty posiela vnútornej vrstve ako žiadané hodnoty (Set Pointy). Táto vrstva zároveň komunikuje

s operátorom, od ktorého prijíma povely a transformuje ich do formátu vhodného pre vnútorný riadiaci obvod. Ďalšími funkciami tejto vrstvy je nadviazanie a udržanie spojenia - komunikácie s dronom prostredníctvom Wifi a interpretácia stavových dát zaslaných dronom. Pozn. práve do tejto vrstvy bude implementovaný Korektor.

Vnútorná riadiaca vrstva je implementovaná priamo v drone a má na starosti hlavne riadenie napätí na jednotlivé motory rotorov na základe žiadaných hodnôt jednotlivých uhlov. Pri rôznych výkonových nastaveniach jednotlivých rotorov bude dron realizovať manévre, ktoré sú zobrazené na diagramoch 3.5.4.



Obr. 3.5.4: Pohyby drona pri určitom výkone rotorov.

Riadenie

Z pohľadu riadenia sa jedná o nestabilný spojitý mnohorozmerový systém s veľmi rýchlou dynamikou. Dron má štyri vstupy a zrovna tak aj štyri výstupy, preto ho môžeme zaradiť medzi MIMO systémy. Väčším problémom pri návrhu riadenia sú však výrazné vzájomné interakcie s nelineárnym charakterom medzi týmito veličinami. Ďalej môžeme dron považovať za systém so šiestimi stupňami voľnosti (6-DoF) , ktorými sú jednak priestorové súradnice polohy drona (x, y, z), respektíve (p_x, p_y, p_z) a taktiež eulerove uhly (ϕ, θ, ψ).

Vstupmi do systému sú tri uhly náklonu, ktorými sa primárne riadi pohyb vo vodorovnej rovine a štvrtým vstupom je rýchlosť vo vertikálnom smere, ktorým sa riadi letová výška dronu. Samozrejme dron má vlastný vnútorný riadiaci systém, ktorý riadi priamo napätia

na servomotory, pre ktorý sú referenciami práve vyššie spomenuté vstupy. Vektor vstupov môžeme zadefinovať ako $u = (u_\phi, u_\theta, u_{\dot{\psi}}, u_{\dot{z}})^T$, kde u_ϕ je referencia na Roll, u_θ na Pitch, $u_{\dot{\psi}}$ na Yaw (rýchlosť otáčania okolo vertikálnej osi) a $u_{\dot{z}}$ predstavuje žiadanú hodnotu pre vertikálnu rýchlosť (v smere z). Všetky tieto veličiny sú normované a preto ich očakávané hodnoty sú z intervalu $< -1,\ 1 >$ v prípade použitia modelu AR.Drone 2.0.

Stavové veličiny tohto systému majú nasledovnú fyzikálnu interpretáciu:

$$x = (p_x, p_y, p_z, \dot{p}_x, \dot{p}_y, \dot{p}_z, \phi, \theta, \psi, \dot{\psi})^{\mathrm{T}} \tag{3.5.1}$$

kde prvé tri stavy (p_x, p_y, p_z) reprezentujú polohu v priestore, druhá trojica stavov $(\dot{p}_x, \dot{p}_y, \dot{p}_z)$ udáva rýchlosti v jednotlivých osiach, tretia trojica (ϕ, θ, ψ) zachytáva uhly náklonov vzhľadom na horizontálnu a vertikálnu rovinu ako aj uhol otočenia vzhľadom na definovaný priamy smer. Posledný stav $(\dot{\psi})$ udáva uhlovú rýchlosť pri otáčaní okolo vertikálnej osi (z). Riadeniu drona typu AR.Drone 2.0 sa tiež venuje Števek (2016).

Komunikácia

Ako už bolo vyššie spomenuté, dron komunikuje prostredníctvom WiFi na základe UDP protokolu. Celkovo sa realizujú tri služby na samostatných portoch. Port 5556 je vyhradený pre riadenie - posielanie tzv. AT príkazov pre dron, ktoré sú preddefinované. Tieto príkazy sú odosielané na pravidelnej báze, zvyčajne tridsať krát za sekundu.

Aktuálne hodnoty jednotlivých stavových veličín, ako napríklad eulerove uhly, rýchlosť, výška a pod. je možné čítať z portu 5554, kam ich dron posiela. Tieto dáta sú nazývané Navdata a k dispozícii sú dva módy posielania týchto dát: demo a full. V režime full sú dáta odosielané dronom zhruba 200 krát za sekundu a balík dát obsahuje aj veľké množstvo pre užívateľa, respektíve pre vonkajší riadiaci obvod neužitočných dát, ktoré slúžia hlavne na diagnostiku porúch, prípadne sú predpripravené pre budúci rozširujúci hardvér a softvér. Praktický význam v bežnom letovom režime majú iba dáta, ktoré sú posielané v režime demo zhruba 15 krát za sekundu a vo výrazne menšom pakete.

Dron obsahuje dve integrované HD kamery (jednu vpredu a druhú na spodnej strane), takže je možné online sledovanie videa z týchto kamier. Dáta kódujúce video obraz sú prenášané pomocou portu 5555 na klientské zariadenie, kde je ich možné dekódovať a zobrazovať.

Štvrtý port číslo 5559 je vyhradený pre prenos kritických dát, hlavne konfiguračných príkazov, ktoré modifikujú nastavenia drona.

Pri implementácii korektora budeme využívať port 5556, pomocou ktorého budeme po-

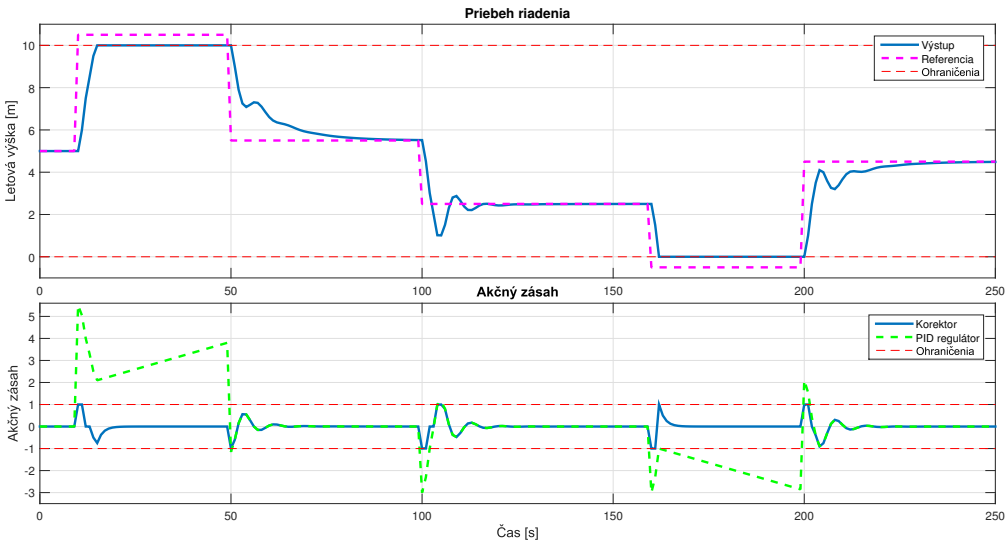
sielať akčné zásahy regulátora vo forme AT-commandu: $'AT*PCMD = seq, [Roll, Pitch, Altitude, Yaw]$ Tento príkaz obsahuje štyri hodnoty, ktoré reprezentujú žiadané hodnoty pre jednotlivé uhly náklonu, respektíve otočenia (Roll, Pitch, Yaw) a taktiež pre výšku. Opačným smerom budú dáta prúdiť portom 5554 a ich konverziou z binárneho formátu získame pre nás dôležité informácie o aktuálnych uhloch, výške a rýchlostiach v jednotlivých osiach. Nezískame však priestorové súradnice, preto bude potrebné navrhnúť pozorovač stavu, ktorý na základe dostupných veličín dopočíta, respektíve odhadne hodnoty týchto stavových veličín.

3.5.2 Riadenie výšky s použitím invariantných množín

Pre účely názornej prezentácie funkcie korektora s ohraničeniami vo forme invariantnej množiny sme sa rozhodli pre simuláciu riadenia jednej zo slučiek, a to riadenia vo vertikálnej osi, čiže letovej výšky dronu. Keď vyextrahujeme relevantnú časť modelu, tak získame dvojstavový model, kde prvým stavom je poloha v z-ovej osi (výška) a druhým je rýchlosť v smere tejto osi. Priamo je možné riadiť iba rýchlosť, ktorá však ovplyvňuje polohu, preto je tento systém plne riaditeľný.

Uvažujme ohraničenia na stavy ako aj na vstupy. Stavové ohraničenia vymedzujú pásmo povolenej letovej výšky - polohu podlahy a stropu v súradniciach z-ovej osi, za predpokladu uzavretého letového priestoru. Ohraničenia na vstupné veličiny zase udávajú maximálne zrýchlenie v kladnom, respektíve zápornom smere, ktoré je limitované výkonmi motorov. Práve z dôvodu existencie týchto ohraničení je výhodné do tohto obvodu zakomponovať korektor, ktorý zabezpečí, že dron nenarazí do stropu a ani sa nerozbije o podlahu, a to aj v takom prípade, že operátor, alebo regulátor by tak chcel učiniť. Názornejšia je však simulačná ukážka, pri ktorej predpokladáme, že s dronom lietame v desať metrov vysokej hale (na úrovni +0,0m sa nachádza podlaha a vo výške +10,0m je strop) a tým pádom je pásmo letovej výšky drona vymedzené intervalom <0, 10> m.

Ako môžeme vidieť v nasledujúcom grafe 3.5.5, tak operátor mení žiadanú hodnotu letovej výšky a implementovaný PI regulátor sa snaží túto referenciu dosiahnuť (alternatívne môžeme predpokladať, že operátor riadi priamo nastavením veľkosti akčného zásahu), no korektor obmedzí akčný zásah do takej miery, aby boli dodržané dané ohraničenia, o ktorých PI regulátor nevie a operátor nechce rešpektovať, keďže v niektorých časoch je referencia mimo ohraničení.



Obr. 3.5.5: Priebeh riadenia s ohraničeniami.

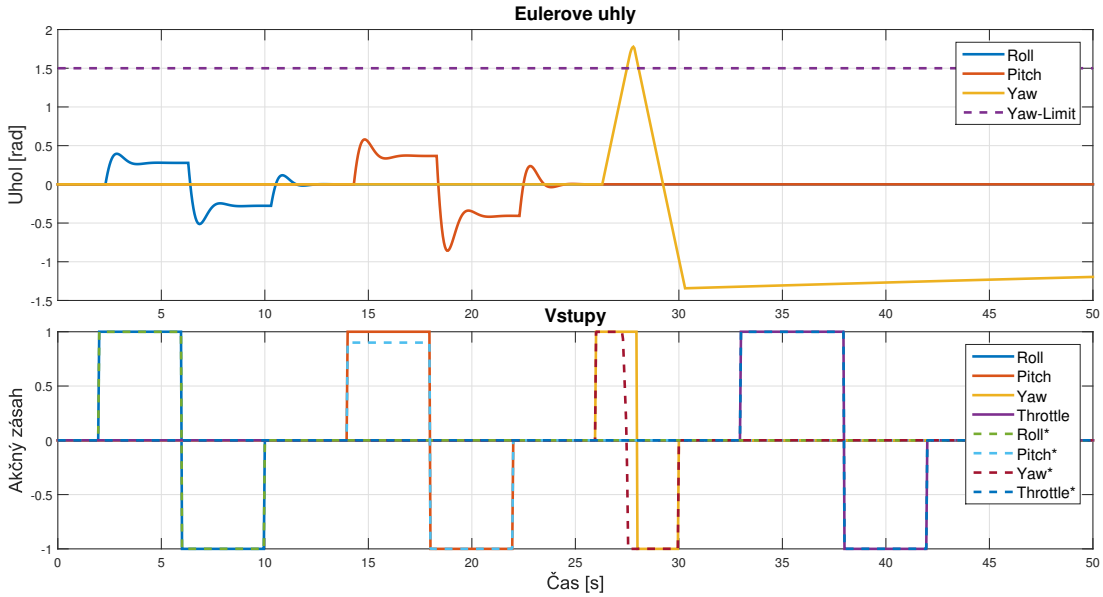
Ako je možné vidieť v grafe 3.5.5, tak korektor jednak oreže akčné zásahy, ktoré sú mimo ohraničení, ale taktiež prispôsobí akčný zásah tak, aby boli dodržané aj stavové ohraničenia. Veľmi dôležitým faktom je, že korektor rešpektuje akčný zásah regulátora, keď sú procesné veličiny v rámci ohraničení, čo môžeme vidieť v čase od 50 do času približne 160 sekúnd, ako aj v posledných päťdesiatich sekundách simulácie.

Na základe prezentovaných výsledkov môžeme konštatovať, že korektor je schopný udržať procesné veličiny v definovaných ohraničeniach a tým zaistiť bezpečnú prevádzku riadeného procesu a to pri maximálnom možnom rešpektovaní akčného zásahu hlavného regulátora. Ako je možné vidieť v grafe 3.5.5 v čase od 20 do 50 sekúnd, tak narastajúci akčný zásah regulátora nemá žiadny vplyv na výstup z korektora, keďže tento akčný zásah by viedol na porušenie stavových ohraničení. Korektor však vždy upraví akčný zásah iba do takej miery, aby boli dodržané ohraničenia a za žiadaných okolností neovplyvňuje kvalitu riadenia, ak sú procesné veličiny v rámci svojich ohraničení. Praktické môže byť vytvorenie "nárazníkového pásma"pomocou mäkkých ohraničení s primeranou penalizáciou doplnkových premenných, čo jednak spôsobí mierne plynulejší priebeh riadenia, zvýši bezpečnosť a v neposlednom rade vytvorí priestor na kompenzovanie neurčitostí pri riadení reálnych procesov,keďže korektor používa lineárny model a tým zvýši robustnosť riadenia, respektíve korekcie.

3.5.3 Riadenie modelu AR.Drone s Korektorom

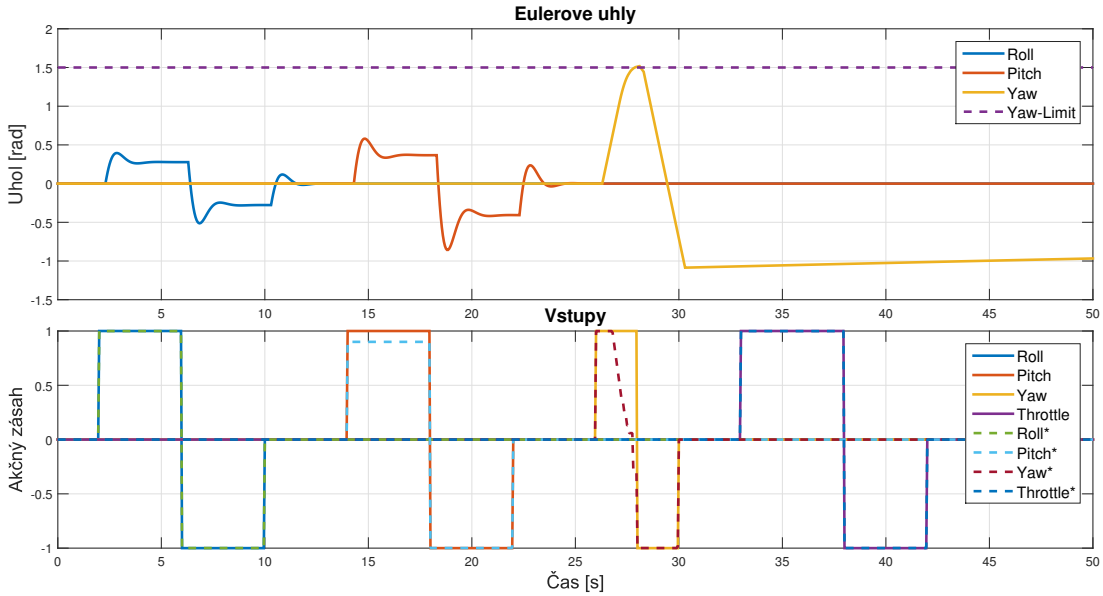
Pre overenie funkcionality a výkonnosti korektora bol v prostredí Simulink implementovaný matematický model drona s patričnými vstupmi, výstupmi a dynamikou (Sanabria, 2013). Keďže uvažujeme prípad, kedy je dron riadený operátorom, tak jednotlivé vstupy sú výsledkom sčítania skokových zmien v rôznych časoch, pričom hodnota výsledného signálu je -1,0,1, čo zodpovedá ovládaniu pomocou tlačidla na ovládači a dôležitá je šírka pulzu. Postupne budeme simulovať scenáre s rôznymi ohraničeniami a nastaveniami v rámci korektora, ktorý je zaradený medzi "operátorom" a procesom s cieľom modifikovať vstupy v prípade, že vedú na porušenie ohraničení. Dôležité je poznamenať, že boli implementované ohraničenia na vstupy a výstupy, pričom výstupné ohraničenia boli definované ako mäkké, s príslušnou penalizáciou doplnkových premenných v účelovej funkcii.

Ako prvé uvažujme horné ohraničenie na uhol otočenia - Yaw na úrovni 1,5 rad. Zároveň sme zadefinovali ohraničenie pre žiadanú hodnotu Pitch, takže vstupná hodnota Pitch bude orezaná na hodnotu 0,9. Pri nastavení predikčného horizontu v korektore na dĺžku 5 získame výsledky prezentované v grafe 3.5.6:



Obr. 3.5.6: Ohraničenie Yaw.

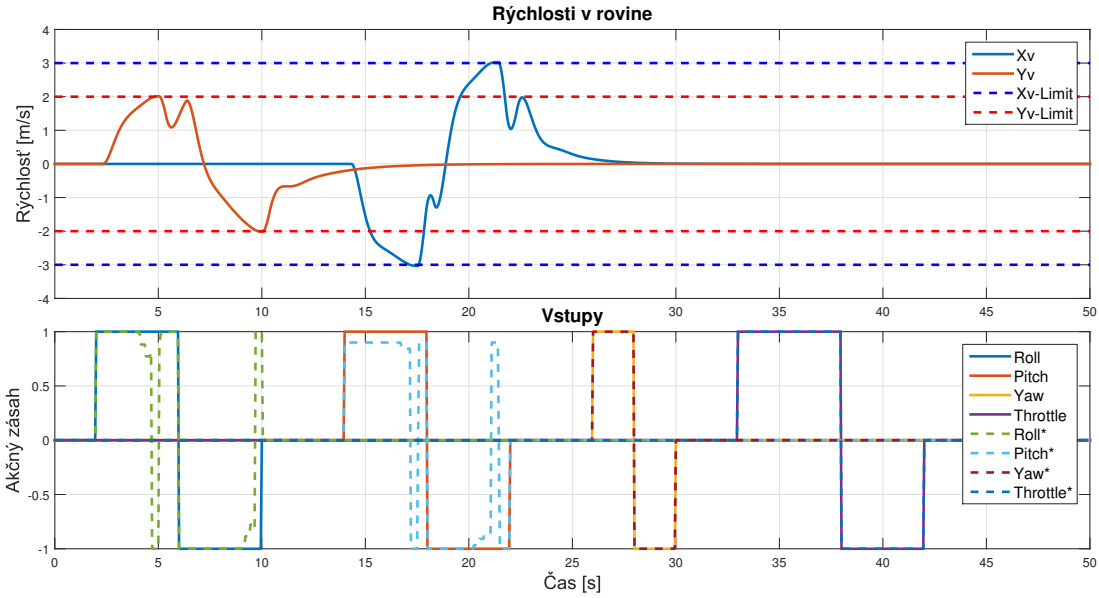
Z grafu 3.5.6 je zrejmé, že dané ohraničenie bolo mierne porušené, čo je prípustné vzhľadom k tomu, že sa jedná o mäkké ohraničenie, no na druhej strane akékoľvek porušenie ohraničení je nežiadúce. Graf vstupov 3.5.6 zobrazujúci okrem iných aj priebeh žiadanej hodnoty pre Yaw posielaný do vnútorného regulátora ukazuje, že pôvodný akčný zásah operátora bol korektorom modifikovaný s cieľom zmierniť deviáciu ohraničenia, no nedostatočne. Jednou z možností ako zredukovať porušenie mäkkých ohraničení je zvýšenie penalizácie doplnkových premenných, ale v tomto prípade by to nebolo účinné opatrenie, pretože porušenie je spôsobené rýchlou dynamikou systému, na ktorú korektor nestíha reagovať. Z tohto dôvodu zopakujeme simuláciu, ale predĺžime predikčný horizont na 15.



Obr. 3.5.7: Ohraničenie Yaw s N = 15.

Po predĺžení predikčného horizontu pozorujeme výrazné zlepšenie (Obr. 3.5.7) - výstup ostáva tesne pod ohraničením a akčný zásah sa mení plynulejšie, čo znamená, že korektor s hraničnou situáciou počítal v dostatočnom predstihu, preto mohol prijať potrebné opatrenia včas a s menšou razanciou.

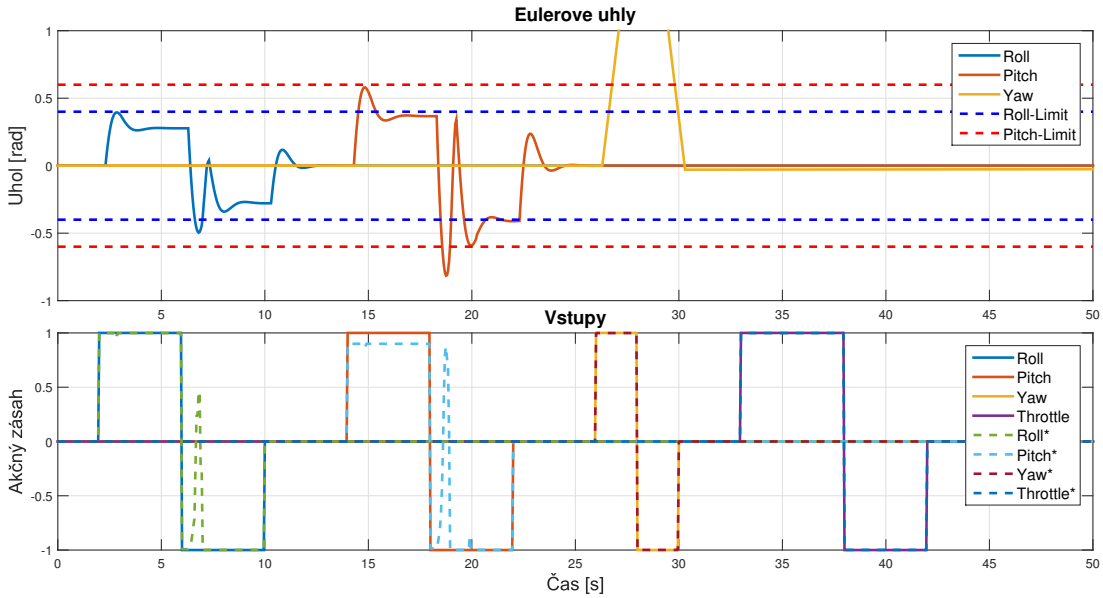
Ďalšími parametrami charakterizujúcimi stav drona sú jeho rýchlosti v horizontálnej rovine, t.j. v osiach X a Y. Tieto rýchlosti závisia od uhlov, hlavne od Roll a Pitch, ktoré sa zase vyvíjajú na základe referencie, ktorú ako vstup pošleme riadiacemu microcontrolleru v drone. Preto ich môžeme nepriamo ovplyvňovať pomocou korektora, ktorý o nich vie na základe matematického modelu. Uvažujme symetrické ohraničenia na rýchlosť v X-ovej a Y-ovej osi s hodnotou 2, respektíve 3, ako to je uvedené v grafe 3.5.8.



Obr. 3.5.8: Ohraničenie na vertikálne rýchlosti.

Ako môžeme vidieť v grafe 3.5.8, tak korektor si bez väčších problémov poradí aj s týmito ohraničeniami, no za cenu ostrejších zmien vstupov, ktoré však vyplývajú z optimalizačného problému, keďže sa snažíme minimalizovať odchýlku od pôvodného akčného zásahu. Ak by sa vyžadoval plynulejší priebeh, tak by pomohlo predĺženie predikčného horizontu v spojení s úpravou váhovania v účelovej funkcii. Zároveň si môžeme všimnúť, že ostatné vstupy, ktoré nemajú na ohraničené veličiny vplyv sú korektorom nedotknuté a sú plne rešpektované pokyny operátora.

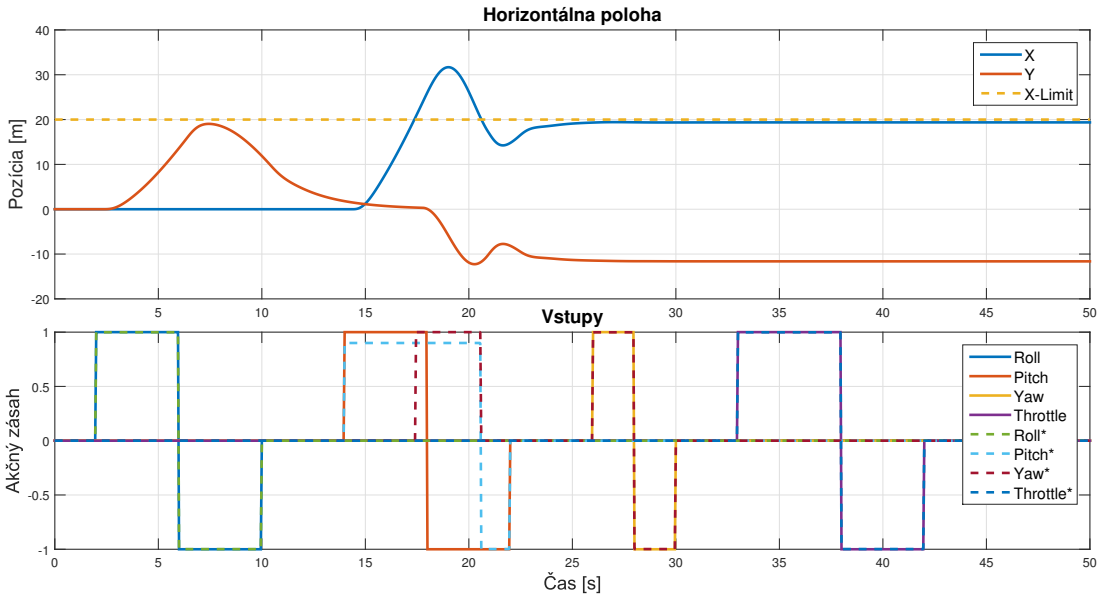
Uhly bočného a čelného náklonu - Roll a Pitch sú riadenými veličinami vnútorných riadiacich slučiek, no vhodnou modifikáciou ich referencie vieme ovplyvniť aj ich priebeh (Obr. 3.5.9).



Obr. 3.5.9: Ohraničenie na uhly náklonu.

Ani pri predikčnom horizonte dlhom 25 períód vzorkovania nedokážeme udržať výstup v ohraničeniach (Obr. 3.5.9). V kladnej oblasti sú ohraničenia dodržané, no v zápornej oblasti pozorujeme malé podregulovanie, na ktoré systém zareaguje s určitým oneskorením a to prudkou zmenou referencie. Zdrojom problému sú pravdepodobne pomerne agresívne naladené vnútorné regulátory s integračnou zložkou.

Poslednou, no nie menej významnou veličinou, ktorej sa budeme venovať je poloha drona v horizontálnej rovine. Ako ilustračný príklad budeme uvažovať ohraničenie na X-ovú súradnicu:



Obr. 3.5.10: Ohraničenie na polohu.

Ako môžeme vidieť v grafe 3.5.10, tak pomerne rýchla dynamika spôsobila kmitavý regulačný pochod, no aj po výraznejšom prekmite sa výstup reprezentujúci polohu vzhľadom na x-ovú súradnicu stabilizoval a udržal tesne pod ohraničením. Malá odchýlka v ustálenom stave ako aj zhoršená kvalita je pravdepodobne dôsledkom nepresnosti modelu, ktorý vznikol linearizáciou výrazne nelineárneho systému obsahujúceho okrem iného funkcie sínus a kosínus. Za povšimnutie stojí graf riadiacich veličín, v ktorom je možné vidieť, že Korektor použil dve rôzne vstupné veličiny v snahe udržať výstup v ohraničeniach. Práve tento fakt prameniaci z optimality Korekora zvyšuje jeho hodnotu, keďže v kritickej situácii nezabezpečí len jej predídeniu, ale urobí to optimálne - vždy nájde to najlepšie, najlacnejšie, prípadne najmenej bolestivé riešenie daného problému.

3.6 Zhrnutie

Korektor predstavuje nový koncept, ktorý má perspektívu uspieť v určitej oblasti aplikácií, keďže ide skôr o bezpečnostný ako riadiaci prvok regulačného obvodu. Tak ako aj ďalšie systémy prezentované v tejto práci, aj Korektor je derivátom MPC a pracuje na báze riešenia optimalizačného problému v každej perióde vzorkovania, no jeho primárnou

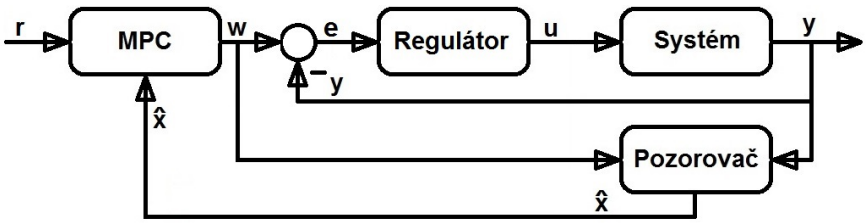
úlohou nie je riadiť, ale posudzovať, či daný akčný zásah regulátora je bezpečný vzhľadom na ohraničenia. Korektor predchádza nebezpečnej situácii - porušeniu ohraničení len v nevyhnutnom rozsahu, preto ho je možné považovať za pasívny prvok obvodu do momentu, kedy sa schyľuje k porušeniu ohraničení. Očakávané správanie bolo overené a demonštrované na nelineárnom modeli kvadrokoptéry AR.Drone 2.0, ktorá predstavuje mnohorozmerový MIMO systém s rýchlou dynamikou. Prezentované výsledky potvrdzujú, že koncepcia Korektora je správna cesta, ktorá si môže v budúcnosti nájsť svoje miesto v oblasti automatizácie a riadenia procesov, no ešte stále tu je množstvo priestoru na jej zlepšenie a zdokonalenie. Dôležitým aspektom je výpočtová náročnosť, ktorej sme sa z časti venovali aj v kapitole 3.4 o invariantných množinách, keďže ich úlohou je zníženie výpočtovej náročnosti skrátením predikčného horizontu, čoho dôsledkom je redukcia počtu optimalizovaných premenných.

Kapitola 4

Tvarovač referencie - Reference governor

Existuje mnoho stratégií a prístupov k riadeniu technologických procesov, ktoré sa viac, či menej ujali v praxi, prípadne si našli svoje miesto v akademickej sfére a ich nástup do priemyslu len príde. Jedno z popredných miest si bezpochyby už niekoľko desaťročí držia PID regulátory, ktoré sú jasným favoritom pre riadenie procesných veličín so spojitým priebehom. Jednoduchosť, predvídateľnosť podložená rozsiahlou teóriou, ako aj pomerne jednoduché nastavenie ich predurčili do väčšiny súčasných priemyselných aplikácií s perspektívou aj do budúcnosti. S rozvojom technológií smerom ku komplexnejším procesom a vyšším nárokom na riadenie sa však čím ďalej viac prejavuje jedno z ich hlavných obmedzení - schopnosť riadiť iba jeden výstup. Tento nedostatok sa prejaví pri riadení MIMO systémov, kedy je potrebné použiť viac PID regulátorov. Problém však nastáva v prípade, že jednotlivé procesné veličiny medzi sebou výraznejšie interagujú a tým pádom pri zmene jednej zo vstupných veličín sa zmena prejaví vo viacerých výstupoch, ktoré sa preto odchýlia od svojich referencií. Na regulačnú odchýlku zareagujú príslušné regulátory zmenou im pridelených vstupov, čo vedie k ovplyvneniu ešte väčšieho množstva výstupov a tým k zníženiu kvality riadenia. A to nehovoriac o celkovej účinnosti (napríklad energetickej) daného zariadenia, alebo celej prevádzky, keďže jednotlivé PID regulátory navzájom o sebe "netušia" a nie to už o cene vstupov, ktoré riadia. Je zrejmé, že ich zásahy musia byť koordinované "zhora" niekým, kto má dostatočný prehľad o procese ako o celku - a tým je práve Reference governor.

Reference governor je "hlavný", respektíve nadradený regulátor, ktorého úlohou je nastavovať žiadané hodnoty pre jednotlivé podriadené (väčšinou PID) regulátory a tým riadiť danú prevádzku ako celok, s prihliadnutím na jej stav ako aj na definované kvalitatívne a ekonomické ukazovatele. V úlohe reference governora takmer spravidla nájdeme práve MPC, ktoré je predurčené pre riadenie mnohorozmerových systémov (Klaučo, 2015). V takomto usporiadaní dokonca nie je až taká problematická ani vysoká výpočtová náročnosť MPC, keďže od reference governora sa doslova vyžaduje aby bol niekoľkonásobne pomalší ako podriadené regulátory, vzhľadom k tomu, že im zadáva žiadané hodnoty. Koncepcia reference governora na báze MPC si už našla svoje miesto aj v priemysle, hlavne v rafinériách, no ešte stále je aktuálnou témou aj v akademickej sfére. Pozíciu Reference governora v rámci uzavretého regulačného odvodu znázorňuje nasledujúca schéma 4.1:



Obr. 4.1: MPC v úlohe Reference governora v URO.

4.1 Formulácia Reference governora

Reference governor (RG) je vo svojej podstate MPC aplikované na riadenie špecifického systému - uzavretej regulačnej slučky obsahujúcej vlastný regulátor, ktorým môže byť napr. PID regulátor, alebo aj MPC. Ako vyplýva zo schémy 4.1, RG generuje žiadanú hodnotu w (modulovanú referenciu) pre podriadený regulátor na základe informácie o užívateľskej referencii r a aktuálnom stave systému $\hat{x}$. V prípade, že stavové veličiny systému sú merateľné, t.j. sú výstupmi zo systému, bolo by možné vynechať pozorovač stavov. No zároveň je potrebné uvedomiť si, že MPC pracuje s modelom celého URO vrátane podriadeného regulátora, ktorý zvyšuje rád systému, respektíve počet stavových veličín, ktorých hodnotu musíme taktiež poznať. S cieľom uľahčiť implementáciu a predísť chybám bol preto blok RG prezentovaný v kapitole 4.2 navrhnutý tak, že vstupom je výstupná veličina y, respektíve vektor výstupných veličín a odhad stavu prebieha v rámci bloku.

Čo sa týka samotnej formulácie reference governora (4.1.1), tak tá je takmer identická

s formuláciou MPC, až na pár detailov:

$$\min \sum_{k=0}^{N-1} (r-y_k)^\top Q_\mathrm{e}(r-y_k) + \Delta w_k^\top Q_{\Delta\mathrm{w}} \Delta w_k + (r-w_k)^\top Q_\mathrm{rw}(r-w_k) + s_k^\top Q_\mathrm{s} s_k \tag{4.1.1a}$$

$$\text{s.t. } \tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}w_k, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1b}$$

$$y_k = \tilde{C}_\mathrm{y}\tilde{x}_k + \tilde{D}_\mathrm{y}w_k, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1c}$$

$$u_k = \tilde{C}_\mathrm{u}\tilde{x}_k + \tilde{D}_\mathrm{u}w_k, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1d}$$

$$\tilde{x}_0 = \tilde{x}(t) \tag{4.1.1e}$$

$$\Delta w_k = w_k - w_{k-1}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1f}$$

$$\tilde{x}_\mathrm{min} - s_{\mathrm{x},k} \leq \tilde{x}_k \leq \tilde{x}_\mathrm{max} + s_{\mathrm{x},k}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1g}$$

$$y_\mathrm{min} - s_{\mathrm{y},k} \leq y_k \leq y_\mathrm{max} + s_{\mathrm{y},k}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1h}$$

$$u_k \in \mathcal{U}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1i}$$

$$w_k \in \mathcal{W}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1j}$$

$$\Delta w_k \in \Delta\mathcal{W}, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1k}$$

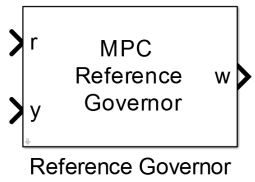
$$s_k \geq 0, \quad \text{pre} \quad k = 0, \ldots, N-1 \tag{4.1.1l}$$

Účelová funkcia implementovanej formulácie obsahuje štyri členy, ktoré sú penalizované ako kvadrát 2-normy s príslušnými váhovými maticami. Prvý člen $(r-y_k)^\top Q_\mathrm{e}(r-y_k)$ penalizuje regulačnú odchýlku, druhý $\Delta w_k^\top Q_{\Delta\mathrm{w}} \Delta w_k$ zmierňuje oscilácie a prispieva k plynulejšiemu priebehu modulovanej referencie (w). Úlohou tretieho člena $(r-w_k)^\top Q_\mathrm{rw}(r-w_k)$ je udržiavať vypočítanú žiadanú hodnotu v blízkosti užívateľskej referencie (r) a posledný člen penalizuje vektor doplnkových premenných s_k, ktorý obsahuje slacky z mäkkých ohraničení (4.1.1g) a (4.1.1h).

Vo formulácii je zámerne použité označenie vektora stavov $\tilde{x}$, aby sme zdôraznili, že ide o rozšírený stavový vektor obsahujúci stavy systému, ako aj stavy PID regulátora. Použitý stavový opis má dve výstupné rovnice (4.1.1c) a (4.1.1d), aby bolo možné ohraničiť aj akčný zásah PID regulátora (4.1.1i).

4.2 Blok pre Simulink

Posledným blokom do knižnice podporných riadiacich systémov na báze MPC je univerzálny blok Reference governora:



Obr. 4.2.1: Blok Reference governora vytvorený pre použitie v Simulinku.

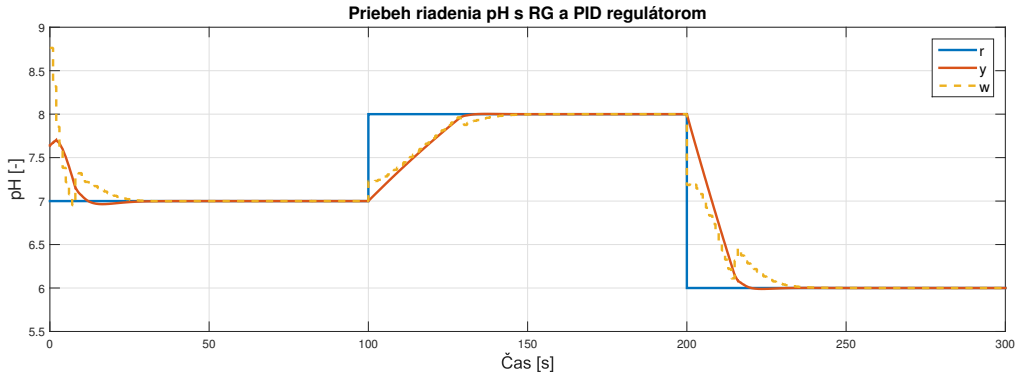
Ako už bolo povedané aj v kapitole 4.1 výstupom z bloku je modulovaná referencia w, ktorá je zároveň žiadanou hodnotou pre PID regulátor. Blok RG v tejto konfigurácii predpokladá PI, alebo PID regulátor v URO. Vstupom do bloku je užívateľská referencia r a taktiež aktuálny výstup zo systému, na základe ktorého sa vo vnútri bloku odhaduje stav systému a regulátora. Konštantné parametre sa do bloku zadávajú prostredníctvom masky 4.2.2:

Obr. 4.2.2: Maska bloku Reference governora v Simulinku.

Hlavným rozdielom v porovnaní s predchádzajúcimi blokmi je, že do masky je potrebné zadať spojitý model procesu (matice A, B, C, D), nakoľko diskretizácia prebehne až po odvodení modelu celého URO vo vnútri bloku. Pre diskretizáciu je nevyhnutná perióda vzorkovania Ts, ktorá sa zadáva spolu s parametrami modelu, ako aj ustálené stavy, pri ktorých bol daný linearizovaný model odvodený. Keďže predpokladáme, že URO obsahuje aj PID regulátor, musíme zadať aj jeho parametre spolu s koeficientom filtra N_f pre derivačnú zložku. Zo skúsenosti odporúčame zadať malé číslo (napr. 0,1), aj keď v bloku PID regulátora bude vyššia hodnota.

Blok obsahuje aj diskrétny Luenbergerov pozorovač pre odhad hodnôt stavových veličín, takže je potrebné zadať póly pozorovača p vo forme riadkového vektora s príslušnou dĺžkou. Pozorovač je potrebné inicializovať určitou hodnotou stavových veličín, preto ak poznáme aspoň približnú hodnotu stavov pri spustení simulácie môžeme zlepšiť kvalitu riadenia, inak zadáme nulový stĺpcový vektor. Stavový vektor modelu URO v rámci bloku je usporiadaný tak, že prvé sú stavy pozorovača a za nimi nasledujú stavy systému podľa zadaného modelu. Samozrejmosťou je dĺžka predikčného horizontu N ako aj jednotlivé váhové matice účelovej funkcie popísané v kapitole 4.1. Poslednými parametrami masky pre RG sú krajné hodnoty intervalov ohraničení jednotlivých stavových veličín. V prípade väčšieho počtu premenných jedného typu (napr. výstupov) očakáva sa stĺpcový vektor limitných hodnôt, pričom poradie zodpovedá zadanému modelu.

Blok bol vyvíjaný a testovaný na lineárnom matematickom modeli neutralizačného reaktora, ktorý vznikol identifikáciou reálneho laboratórneho reaktora. Po vyladení parametrov sme získali regulačný pochod vynesený v grafe 4.2.3. Z grafu vyplýva, že dosahujeme pomerne dobrú kvalitu riadenia, až na prvých pár sekúnd simulácie, v ktorých prebieha inicializácia reference governora a konvergencia pozorovača stavu, čo má značne negatívny vplyv na kvalitu. Tento jav by bolo možné zmierniť, prípadne eliminovať zadaním správnych hodnôt stavových veličín pri inicializácii pozorovača a RG. Pomerne plynulý a mierny priebeh modulovanej referencie je dôsledkom dvoch faktorov: prvým je pomerne silný PID regulátor implementovaný v URO. Druhým sú ohraničenia na veľkosť akčného zásahu, ktoré sú obsiahnuté aj v reference governore, takže RG udržuje pomerne malú regulačnú odchýlku, aby agresívny PID regulátor tieto ohraničenia neporušil.



Obr. 4.2.3: Zmena akčného zásahu.

Algoritmus Reference governora spolu s prislúchajúcim blokom je dostupný a voľne stiahnuteľný na internete[1].

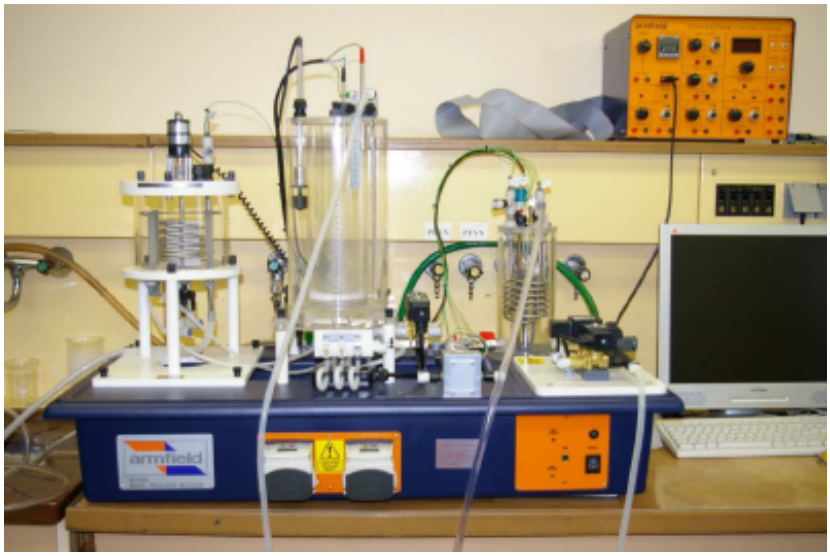
4.3 Experimentálne overenie

Simulink umožňuje aj priame riadenie rôznych "skutočných" technologických procesov, ak je počítač dovybavený vstupno-výstupnou kartou od spoločnosti Humusoft, ktorá dokáže pracovať s digitálnymi ako aj analógovými vstupmi a výstupmi. Na druhej strane je zabezpečená komunikácia s Matlabom a Simulinkom, do ktorého prúdia aktuálne dáta z procesu a zároveň sa odosielajú príkazy pre akčné členy. Táto funkcia Simulinku tým pádom dáva blokom vytváraným v rámci tejto práce nový rozmer, keďže ich uplatnenie nekončí len pri simuláciách, ale umožňuje aj ich priamu a jednoduchú implementáciu pri riadení reálnych procesov.

V rámci tejto práce sme sa rozhodli implementovať Reference governor pri riadení laboratórneho neutralizačného reaktora značky Armfield PCT41 (Obr. 4.3.1). Ide o prietokový miešaný kotlový reaktor (CSTR), do ktorého sú privádzané dva reaktanty (kyselina octová a hydroxid sodný) pomocou peristaltických čerpadiel, ktorých výkon je možné plynulo regulovať. V našom prípade bolo čerpadlo pre dávkovanie kyseliny nastavené na konštantný výkon 50% a dávkovacie čerpadlo zásady bolo akčným členom URO. Boli použité zriedené roztoky reaktantov s koncentráciou $0,01 mol.dm^{-3}$, ktoré zmierňovali vysoké zosilnenie procesu. Výška hladiny v reaktore je samoregulujúca, nakoľko odvod produktu z reaktora

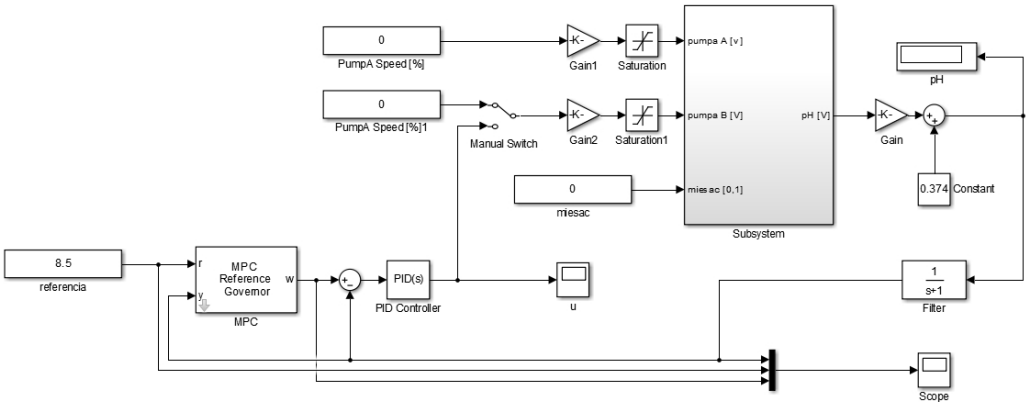
[1] https://Simon_Koniar@bitbucket.org/Simon_Koniar/podporne-riadiace-systemy-na-baze-mpc

je realizovaný cez prepad. Meranou a zároveň riadenou veličinou je pH v reaktore, ktoré je merané pomocou elektródy s napäťovým výstupom. Hodnota pH v reaktore sa mení v závislosti od pomeru dávkovaných reaktantov, keďže hydroxid sodný s koncentráciou $0,01 mol.dm^{-3}$ má pH približne 12 a kyselina octová (ako slabá kyselina) má pri danej koncentrácii $pH \doteq 3,4$. Problémom však je, že hodnota pH sa nemení lineárne, ale silno nelineárne podľa príslušnej titračnej krivky, čo je z pohľadu riadenia vážny problém, keďže zosilnenie procesu sa mení v závislosti od pracovného bodu. Ďalšou komplikáciou je nesymetrická dynamika procesu v smere znižovania a zvyšovania hodnoty pH. Samozrejme výkony čerpadiel sú obmedzené, takže je potrebné kalkulovať len s prietokom, ktorý dokážu poskytnúť. Vzhľadom na tieto okolnosti je riadenie neutralizačného reaktora výzvou.

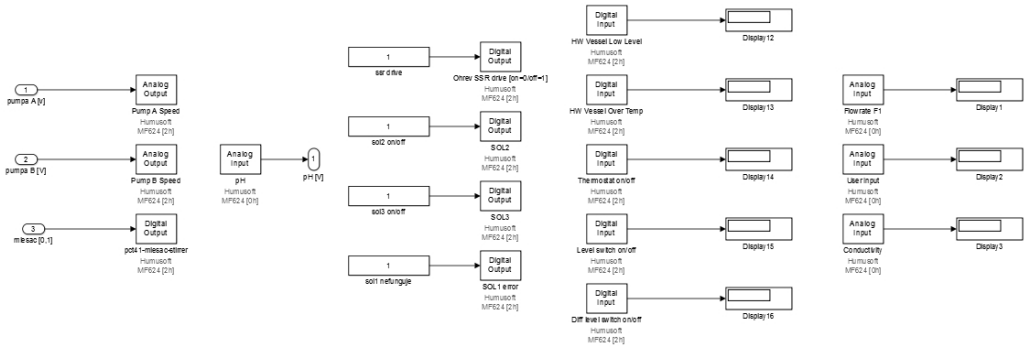


Obr. 4.3.1: Neutralizačný reaktor PCT 40.

Riadenie neutralizačného reaktora bolo realizované prostredníctvom Simulink-ovej schémy (Obr. 4.3.2), ktorá obsahovala blok `Subsystem` reprezentujúci reaktor. Vo vnútri tohto bloku (Obr. 4.3.3) sa nachádzajú jednotlivé analógové/digitálne vstupy a výstupy prepojené prostredníctvom karty Humusoft s reálnym procesom. V danej konfigurácii využívame iba tri vstupy: jeden digitálny, ktorý ovláda miešadlo a dva analógové pre ovládanie čerpadiel. Pri nábehu boli oba čerpadlá nastavené na konštantný výkon 50% až do ustálenia, kedy bolo čerpadlo pre dávkovanie zásady prepnuté na signál z PID regulátora. Výstupný signál bol z voltov prepočítaný pomocou kalibračnej krivky na pH a následne filtrovaný, keďže pH sonda vnášala do systému šum.

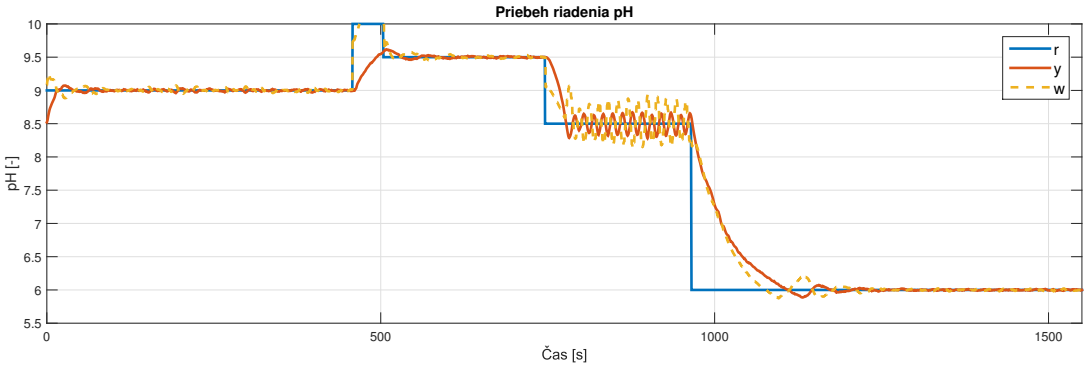


Obr. 4.3.2: Simulinková schéma pre riadenie reaktora.



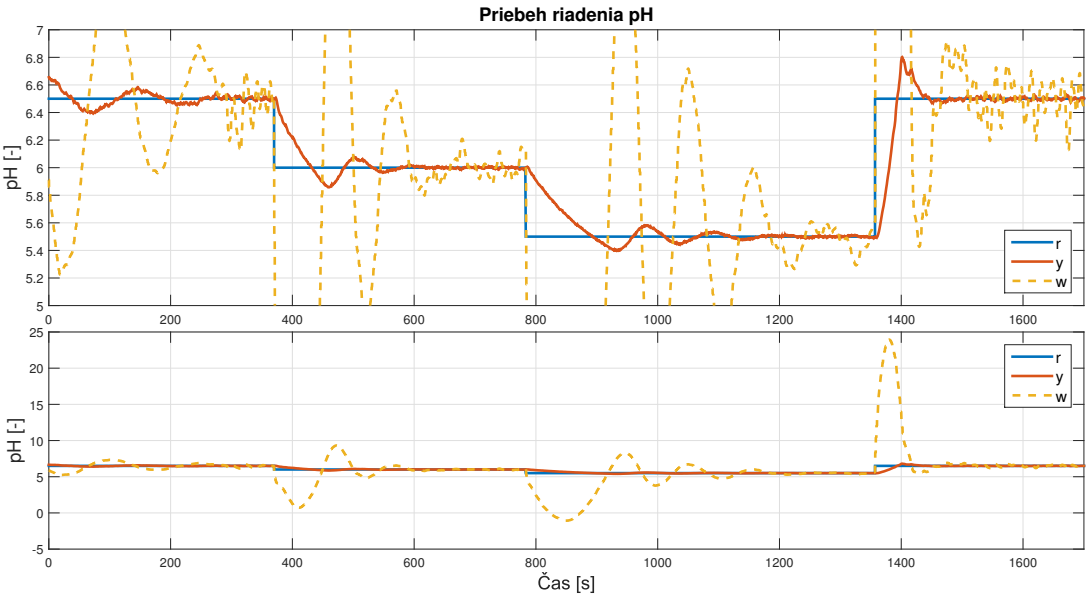
Obr. 4.3.3: Blok pH reaktora v Simulinku.

Prvé meranie (Obr. 4.3.4) bolo realizované po miernej úprave váhových matíc (penalizácia regulačnej odchýlky bola znížená o jeden rád) v porovnaní so simuláciou, čo viedlo na menej agresívny regulačný pochod. Kvalitu riadenia môžeme považovať vzhľadom na charakter procesu za uspokojivú, až na úsek so žiadanou hodnotou $ph = 8,5$, kedy výstup osciluje. Zhoršenie kvality riadenia súvisí s priblížením sa k neutrálnemu bodu, kedy je zosilnenie procesu najväčšie. Tesne pred časom 500 s došlo k zmene referencie, avšak na nedosiahnuteľnú hodnotu (pri danom nastavení procesu), preto bola operatívne znížená na správnu hodnotu.



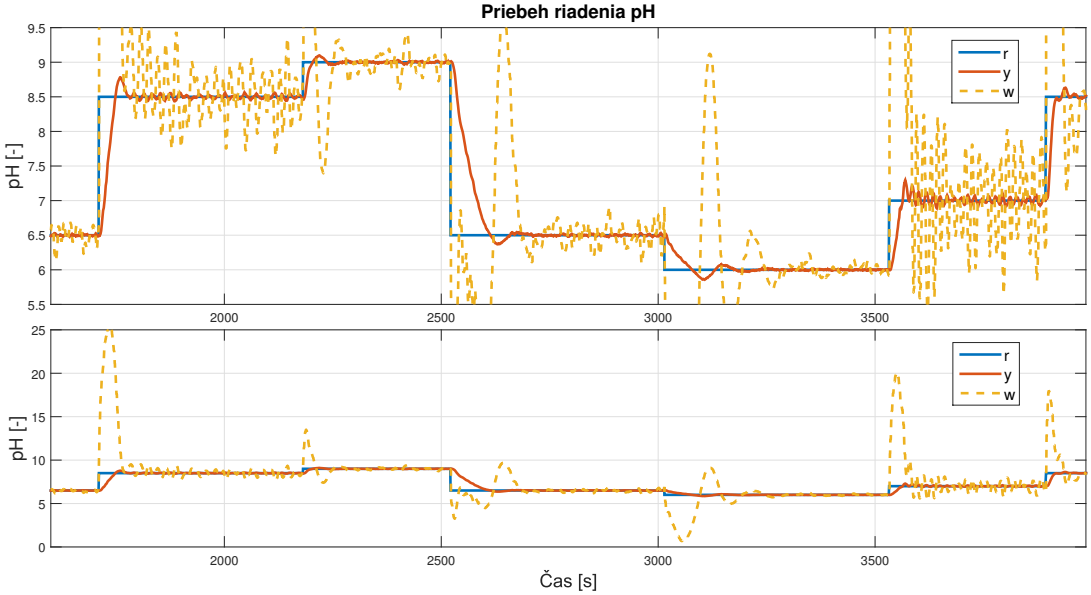
Obr. 4.3.4: Priebeh riadenia pH.

Pri druhom meraní (Obr. 4.3.5) bol použitý zhruba desaťnásobne "slabší" PID regulátor (proporcionálna zložka násobiaca integračnú aj derivačnú zložku bola desaťnásobne nižšia) a zároveň bola zvýšená penalizácia diferencie modulovanej referencie (Δw). Uvedené zmeny sa výrazne prejavili v kvalite riadenia. Čas regulácie sa predĺžil vplyvom výraznejších preregulovaní a následných tlmených oscilácií. Ďalej môžeme pozorovať podstatne výraznejšie zmeny modulovanej referencie (Obr. 4.3.5 - spodný graf), čo sa týka jej absolútnej hodnoty, ktoré sú zapríčinené slabším PID regulátorom - na jeho vybudenie je potrebné "nasimulovať" väčšiu regulačnú odchýlku.



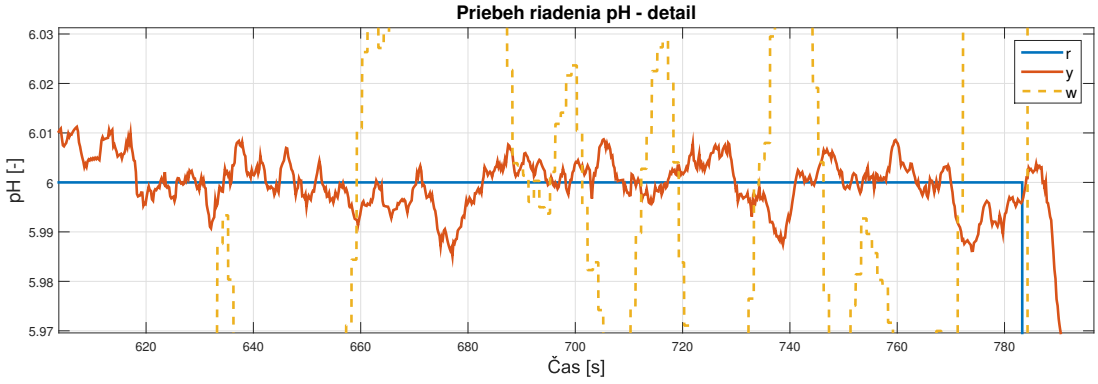
Obr. 4.3.5: Priebeh riadenia pH 1.časť.

Graf 4.3.6 je priamym pokračovaním merania 4.3.5, ale so zníženou penalizáciou diferencie modulovanej referencie s cieľom zrýchlenia dynamiky URO. V porovnaní s prvým meraním 4.3.4 môžeme aj pri hodnote 8,5 pozorovať priebeh bez výraznejších oscilácií. Dokonca aj pri hodnote $pH = 7$ (neutrálny roztok -> najväčšie zosilnenie) sa výstup drží v delta okolí referencie s veľkosťou $\pm 0,1$.



Obr. 4.3.6: Priebeh riadenia pH 2.časť.

Ak sa pozrieme na detail okolia referencie v ustálenom stave (Obr. 4.3.7), tak zistíme, že výstup sa nachádza v delta okolí referencie s veľkosťou $\pm 0,01$, čo je pravdepodobne technologický limit tohto procesu.



Obr. 4.3.7: Priebeh riadenia pH detail.

4.4 Zhrnutie

Reference governor, alebo tvarovač referencie predstavuje posledný príspevok tejto práce do knižnice podporných riadiacich systémov na báze MPC. Aj keď je posledným, neznamená to, že je menej dôležitým, práve naopak. Ako už bolo v rámci tejto kapitoly spomenuté, koncepcia reference governora vo forme MPC si našla svoje uplatnenie aj v priemysle, no zároveň je stále atraktívnou témou aj pre akademickú sféru, čo dáva predpoklad na jej ďalšie zdokonaľovanie a rozširovanie poľa jej pôsobnosti. Výsledkom tejto časti práce je algoritmus pre výpočet optimálnej žiadanej hodnoty, ktorý vznikol modifikáciou prediktívneho riadenia. Zároveň bol vytvorený univerzálny blok pre Simulink, ktorý umožňuje pomerne jednoduchú a intuitívnu implementáciu reference governora, čo výrazne rozširuje skupinu používateľov. Programová časť je voľne prístupná a modifikovateľná, takže skúsenejší používatelia môžu jednoducho prispôsobiť algoritmus svojim potrebám, alebo ho ďalej zdokonaľovať. Ako aj pri ostatných, tak aj pri tomto bloku existuje množstvo priestoru pre zdokonaľovanie a rozširovanie funkcií, ktoré je možne zakomponovať a tým zvýšiť jeho úžitkové vlastnosti. No zároveň treba mať na pamäti výpočtovú náročnosť, ktorá s každým rozšírením vzrastá. Preto je potrebné sa venovať aj tejto oblasti a podnikať kroky vedúce k znižovaniu výpočtovej náročnosti. Druhá časť kapitoly 4 bola venovaná overeniu a demonštrácii funkčnosti bloku. Samozrejmosťou je simulačné overenie na matematickom modeli, na ktorom prebiehal vývoj bloku. Po dosiahnutí výsledkov s požadovanou kvalitou v simulačnej oblasti sme pristúpili k experimentálnemu overeniu. Riadeným procesom bol laboratórny neutralizačný reaktor, ktorý bol ovládaný priamo zo Simulinku, v ktorom bol implementovaný reference governor. Hlavným cieľom tohto kroku bolo poukázať na fakt, že využitie vytvorených blokov zďaleka nekončí pri simuláciách s modelmi, ale je pomerne jednoduché ich implementovať aj pri riadení skutočných procesov. Samozrejme riadenie reálnych procesov prináša so sebou množstvo komplikácií, ale ako bolo prezentované v práci, náš blok reference governora nemal s prechodom žiadny problém a nevyžadoval žiadne dodatočné úpravy. Je pravdou, že kvalita riadenia sa mierne zhoršila, no s prihliadnutím na charakter procesu sa aj tak jedná o výsledok, ktorý vysoko prekonal naše očakávania. Bolo realizovaných viacero meraní, s rôznymi nastaveniami váhových matíc a parametrov PID regulátora za účelom demonštrovania vplyvu týchto ladiacich parametrov na priebeh riadenia. Určite je možné dosiahnuť ešte lepšiu kvalitu riadenia od tej prezentovanej, no tá nie je primárnym cieľom tejto práce.

Záver

Táto práca prezentuje a analyzuje návrh ako aj implementáciu rôznych derivátov prediktívneho riadenia, tak ako to je deklarované v cieľoch práce. Výsledkom práce je taktiež knižnica univerzálnych blokov pre Simulink, ktorá umožňuje pomerne jednoduchú implementáciu jednotlivých podporných riadiacich systémov vyvinutých v rámci tohto diplomového projektu. Práca bola rozčlenená do štyroch logických celkov, z ktorých prvý sa zaoberá prediktívnym riadením, ktoré tvorí základ pre všetky vyvíjané systémy. Keďže existuje mnoho rôznych formulácií MPC, vytvorili sme dva samostatné bloky: "Basic", implementujúci základnú formuláciu a "Advanced", ktorý obsahuje rozšírenu a modifikovanú formuláciu zohľadňujúcu niektoré praktické aspekty implementácie MPC. Overenie bolo realizované na nelineárnom MIMO modeli štyroch zásobníkov kvapalín, čím bola potvrdená funkčnosť vyvinutého bloku.

Druhá časť práce je venovaná návrhu a implementácii pozorovača stavov na báze Moving Horizon Estimation (MHE), ktorý nie je veľmi rozšírený. Najprv prebehol návrh samotného algoritmu pre výpočet stavu systému, založenom na riešení optimalizačného problému v každej perióde vzorkovania. Tento algoritmus bol implementovaný, otestovaný a doladený na matematickom modeli sústavy dvoch zásobníkov kvapalín, avšak výsledkom je univerzálny algoritmus, ktorý umožňuje odhadovať stavy rôznych dynamických systémov. Ďalšou významnou časťou práce je porovnanie výkonu MHE s inými v praxi zaužívanými pozorovačmi stavov pri rôznych podmienkach, čoho cieľom je preukázanie, že aj MHE je plnohodnotnou alternatívou, ba dokonca v niektorých prípadoch optimálnou cestou pre odhad stavu systému. Vo väčšine pípadov dosahoval MHE minimálne porovnateľné výseldky s inými pozorovačmi, no pri systémoch s ohraničeniami mal výrazne navrch a preto práve pri ohraničených systémoch je MHE rozumnou voľbou. V súlade s cieľmi práce bol vytvorený univerzálny blok pre odhad stavu na báze MHE pre Simulink. Tento blok umožňuje pomerne jednoduchú implementáciu MHE pre pozorovanie systémov v prostredí Simulink už pri základných znalostiach danej problematiky, ktoré sú potrebné

na voľbu vhodných parametrov. Interface pre zadávanie parametrov bol konštuovaný tak, aby bol pre užívateľa čo najprehľadnejší a uľahčil mu používanie tohto bloku. Bola taktiež vytvorená dokumentácia k danému bloku (rovnako ako u všetkých ostatných blokoch) pozostávajúca hlavne z nápovedy pre zadávanie jednotlivých parametrov a ich účinku na výsledok - odhadnutý stav. Poslednou významnou úlohou bolo znižovanie výpočtového času, ktorý bol pri pôvodnej formulácii neakceptovateľne vysoký. Po modifikácii algoritmu sa však podarilo priemernú časovú náročnosť výpočtu na periódu vzorkovania znížiť viac ako 100-násobne, čo viedlo k efektívnemu fungovaniu bloku a možnosti jeho uplatnenia pri pozorovaní systémov v reálnom čase. Tento koncept a poznatky z vývoja bloku boli aplikované aj pri tvorbe ostatných blokov.

Tretím tematickým celkom tejto práce bol návrh a rozvoj konceptu korektora ako derivátu MPC, ktorého primárnou úlohou nie je riadenie, ale korekcia akčných zásahov regulátora, tak aby boli dodržané ohraničenia na procesné veličiny. Hlavnou myšlienkou bolo vytvorenie prvku regulačného obvodu, ktorý by bol pasívny až do momentu, kedy generovaný akčný zásah regulátora vedie na porušenie ohraničení, čomu zamedzí práve korektor, ktorý v nevyhnutnej miere upraví akčný zásah. Podobne ako MPC aj korektor je možné pomerne ľahko modifikovať a tým ho prispôsobiť pre konkrétnu aplikáciu. Výsledkom práce je aj univerzálny blok pre Simulink, ktorý umožní používateľovi jednoducho implementovať korektor pre jeho vlatnú aplikáciu. Výkon korektora bol prezentovaný v závere kapitoly na komplexnom, ale názornom praktickom príklade, kde riadeným systémom je dron (kvadrokoptéra). Jedná sa o pomerne zložitý nelineárny MIMO systém s výraznými interakciami medzi veličinami, ktorého riadenie je výzvou. Boli prezentované rôzne simulačné scenáre, na ktorých sme demonštrovali princíp fungovania a výkonnosť Korektora. Toto overenie potvrdilo teoretické princípy, z ktorých sme vychádzali a zároveň preukázalo opodstatnenosť rozvoja tohto konceptu. Možno práve tento koncept otvorí dvere optimálnemu riadeniu a tým pádom aj vyššej bezpečnosti aj do konzervatívnejších aplikácií, v ktorých sú zaužívané určité prístupy k riadeniu a nie je tu dostatočná motivácia ich meniť. V práci sú prezentované simulačné výsledky, no cieľom do budúcnosti je implementovať korektor pri riadení reálneho zariadenia, čím by sa ešte názornejšie preukázala jeho opodstatnenosť. Zároveň by sa overila jeho robustnosť a odolnosť, napríklad na vplyv vonkajších porúch, čo by viedlo na jeho ďalšie zdokonalenie, alebo modifikáciu. Chyby sa stávajú a vždy budú, no práve korektor môže zabezpečiť aby sa pochybenia zaobišli bez vážnych následkov.

Nad rámec pôvode definovaných cieľov práce bol v rámci poslednej kapitoly vyvinutý blok spolu s algoritmom implementujúci Reference governor. Tento krok bol logickým pokračovaním začatej práce, keďže reference governor je takisto koncepcia vychádzajúca

z MPC a zároveň sa mu podarilo vo výraznejšej miere preniknúť aj do priemyselných aplikácií. Pravdepodobne najvyššiu hodnotu v tejto časti práce má popri algoritme a bloku práve experimentálna verifikácia, ktorá poukazuje na fakt, že využitie vytvorených blokov zďaleka nekončí pri simuláciách s modelmi, ale je pomerne jednoduché ich implementovať aj pri riadení skutočných procesov. Zároveň za zmienku stojí, že prechod zo simulačného prostredia na reálny proces bol úplne bezproblémový a jedinou zmenou bolo zníženie hodnoty jednej z váhových matíc, čím sa dosiahla pomerne dobrá kvalita riadenia. Ďalej bolo preukázané, že zmenou ladiacich parametrov sa dajú dosiahnuť rôzne priebehy riadenia, takže je možné reference governor nastaviť podľa daných požiadaviek. Samozrejme, ako aj pri ostatných blokoch prezentovaných v rámci tejto práce existuje množstvo priestoru na zlepšovanie a zdokonaľovanie. Do budúcna by bolo prínosné ak by sa napríklad formulácia (spolu s blokom) rozšírila o možnosť použitia viacerých PID regulátorov súčasne, čím by sa oblasť aplikácií obohatila o MIMO systémy, ktoré sú cieľovou skupinou pre reference governory.

Literatúra

BESANCON, G. 2007. *Nonlinear Observers and Applications*. New York: Springer Verlag, 2007. 224 s. ISBN: 978-3-540-73502-1.

BORRELLI, F. et al. 2015. *Predictive Control for linear and hybrid systems*. 2015. 6 s.

HOLAZA, J. et al. 2015. *Safety Verification of Implicitly Defined MPC Feedback Laws*. Bratislava: Slovenská technická univerzita, 2015. 6 s.

KALMAN, R. E. 1960. *A New Approach to Linear Filtering and Prediction Problems*. Journal of Basic Engineering, March 1960, s. 35-45.

KARAS, A. et al. 2007. *Praktické aspekty prediktívneho riadenia*. Bratislava: Slovenská technická univerzita, 2007. 171 s. ISBN 978-80-89316-06-9.

KLAUČO, M., KVASNICA, M. 2015. *Control of a Boiler-Turbine Unit Using MPC-based Reference Governors*. Bratislava: Slovak University of Technology, 2015. 27 s.

KONIAR, S. 2015. *Prediktívne riadenie rektifikačnej kolóny*. Bratislava: Slovenská technická univerzita, 2015. 82 s.

LÖFBERG, J. 2004. *YALMIP : A Toolbox for Modeling and Optimization in MATLAB*. CACSD Conference, Taipei, Taiwan, 2004.

MACIEJOWSKI, J.M. 2002. *Predictive Control with Constraints*. Harlow: Pearson Education Limited, 2002. 331 s. ISBN 0-201-39823-0.

PISKORSKI, S., BRULEZ, N. 2011. *AR.Drone Developer Guide*. Parrot, 2011.

SANABRIA, E. et al. 2013. *ARDrone Simulink Development Kit*. University of Minnesota, 2013.

ŠTEVEK, J. et al. 2016. *Teaching Aids for Laboratory Experiments with AR.Drone2 Quadrotor*. Bratislava, 2016. 8s.

ZAVALA, V. et al. *A Fast Moving Horizon Estimation Algorithm Based on Nonlinear Programming Sensitivity*. Pittsburgh PA: Carnegie Mellon University, 2008. 19 s.