SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Evidenčné číslo: FCHPT-5414-68764

Prediktívne riadenie flexibilného dopravného pása

Diplomová práca

2014 **Bc. Martin Franc**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Prediktívne riadenie flexibilného dopravného pása

Diplomová práca

FCHPT-5414-68764

Študijný program:	Automatizácia a informatizácia v chémii a potravinárstve
Číslo študijného odboru:	2621
Názov študijného odboru:	5.2.14 Automatizácia
Školiace pracovisko:	Ústav informatizácie, automatizácie a matematiky
Vedúci záverečnej práce:	doc. Ing. Michal Kvasnica, PhD.
Konzultant:	Ing. Bálint Takács

Bratislava 2014 **Bc. Martin Franc**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2013/2014
Evidenčné číslo: FCHPT-5414-68764

STU FCHPT

ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Martin Franc**
ID študenta: 68764
Študijný program: automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor: 5.2.14 automatizácia
Vedúci práce: doc. Ing. Michal Kvasnica, PhD.
Konzultant: Ing. Bálint Takács

Názov práce: **Prediktívne riadenie flexibilného dopravného pása**

Špecifikácia zadania:

Cieľom projektu je prediktívne riadenie flexibilného dopravného pásu. Pod pásom je umiestnených 6 piestov, ktorých vysúvaním vzniká na vlna, ktorá pohybuje transportovaným objektom. Úlohou riadenia je ovládať jednotlivé piesty tak, aby vznikla vlna požadovaného tvaru, ktorá presunie objekt na želané miesto.

Úlohy:
* Vytvoviť matematický model flexibilného dopravníka.
* Navrhnúť pozorovač stavu, ktorý na základe merania polohy určí rýchlosť objektu.
* Navrhnúť a implementovať LQR regulátor bez ohraničení.
* Navrhnúť a implementovať prediktívny regulátor s ohraničeniami.
* Navrhnúť spôsob implementácie v reálnom čase.
* Experimentálne overiť činnosť regulátorov a kvalitatívne ich porovnať.

Rozsah práce: 50

Zoznam odbornej literatúry:

1. Kvasnica, M. *Real-Time Model Predictive Control via Multi-Parametric Programming:Theory and Tools : Theory and Tools.* Saarbrücken: VDM Verlag Dr. Müller, 2009. 231 s. ISBN 978-3-639-20644-9.

Riešenie zadania práce od: 17. 02. 2014

Dátum odovzdania práce: 24. 05. 2014

L. S.

Bc. Martin Franc
študent

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

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

Poďakovanie

Ďakujem vedúcemu diplomovej práce doc. Ing. Michalovi Kvasnicovi, PhD. a konzultantovi Ing. Bálintovi Takácsovi za pomoc pri získavaní vedomostí z oblasti prediktívneho riadenia ako i celkovo za vedenie, cenné rady a pripomienky, ktoré mi poskytli počas štúdia a pri vypracovaní práce.

Abstrakt

Práca sa venuje problematike prediktívneho riadenia reálneho zariadenia, ktorým je flexibilný dopravný pás. Pod pásom sú na oboch koncoch umiestnené piesty, ktorých vysúvaním vzniká naklonenie roviny, ktoré pohybuje transportovaným objektom. Úlohou riadenia je zabezpečiť presúvanie objektu na žiadané miesto. Návrh pokročilejších stratégii riadenia zahrňuje vytvorenie modelu procesu, pozorovača v prípade nemerateľných stavov, a regulátora. Ciele práce sú vytvoriť matematický model pohybu gule na naklonenej rovine. Potom na základe modelu navrhnúť a implementovať optimálny pozorovač pre odhad nemeranej rýchlosti objektu. Po zabezpečení kvalitného odhadovania všetkých stavov navrhnúť a implementovať LQR regulátor bez ohraničení a prediktívny regulátor s ohraničeniami. A na záver experimentálne overiť činnosť regulátorov a kvalitatívne ich porovnať.

Abstract

The aim of the thesis is to implement model predictive control (MPC) technique on real device, which in our case is a flexible conveyor belt. The pistons are situated under the belt and by ejecting them we are able to change the position of the controlled object. The goal is to move the object to the desired position. To achieve this task at first we need to identify the systems dynamic behaviour, since not all states are measurable, state estimator is required as well. After the successfully tuning and implementation of the state estimator on the real device we design two types of controllers. The first one is linear-quadratic regulator (LQR) and the second one is MPC based controller with constraints. These controllers were tested under the same conditions and compared by qualitative indicators.

Obsah

Úvod

Prediktívne riadenie alebo MPC (*model predictive control*) patrí v súčasnosti medzi najpokročilejšie a najpoužívanejšie stratégie riadenia, hlavne v chemickom a petrochemickom priemysle. Základný princíp prediktívneho riadenia spočíva v predikcii budúceho správania riadeného procesu a formulácie účelovej funkcie, ktorá odráža cieľ riadenia, a zavádza prídavné ohraničenia na vstupy, stavy a výstupy. Často sa formuluje ako problém kvadratického programovania, ktorý sa rieši každú periódu vzorkovania pre získanie optimálneho, akčného zásahu. Pod pojmom MPC si môžeme tiež predstaviť LQR (*linear quadratic regulator*) riadenie doplnené o ohraničenia. Rovnako LQR aj MPC sú založené na stavovej spätnej väzbe.

Hlavným cieľom práce je poukázať na prednosti prediktívneho regulátora, ktorý umožňuje pracovať s ohraničeniami v porovnaní s LQR regulátorom a jednoduchým PID regulátorom pri riadení flexibilného dopravného pásu. Tento systém predstavuje dvojitý integrátor a má jeden vstup, uhol naklonenia pásu a jeden výstup, pozícia gule na páse. Uhol naklonenia pásu je zabezpečovaný vysúvaním a zasúvaním piestov na koncoch pásu. Celé modelové zariadenia je postavené zo stavebnice Lego Mindstrom a mozgom zariadenia je mikropočítač, ktorý komunikuje s osobným počítačom. Na dopravnom páse sa nachádza guľa, a pri zmene uhla dopravného pásu sa vplyvom gravitácie začne pohybovať smerom nadol. Rôzne typy regulátorov budú navrhnuté tak, aby bolo možné pozíciu gule na páse ovládať.

Práca je rozdelená na tri časti teoretickú, praktickú a záver. Teoretická časť sa začína opisom konštrukcie zariadenia, snímačov a akčných členov. Pokračuje manuálom pre pripojenie osobného počítača k dopravnému pásu prostredníctvom USB a popisom spôsobu práce robota v prostredí MATLABU. Po opise zariadenia a jeho súčastí nasleduje modelovanie procesu gule na naklonenej rovine a odvodenie prenosovej funkcie a stavového modelu. Nakoniec sa teoretická časť zaoberá problémami pozorovania stavu, LQR a MPC riadeniu. V úvode praktickej časti sa nachádza spôsob kalibrácie snímačov vzdialenosti a prepočet výstupu zo snímačov na konkrétnu pozíciu v centimetroch a pokračuje úpravou stavového modelu na základe vykonaných skokových zmien.

Potom nasleduje návrh Kalmanovho filtra pre odhad rýchlosti gule a návrhy a implementácie jednotlivých stratégii riadenia v MATLABE. Ako prvé je cieľom navrhnúť LQR stavovú reguláciu bez ohraničení a MPC stavovú reguláciu s ohraničeniami a porovnať ich priebehy riadenia. Nasleduje návrh a implementáciu LQR regulátora bez ohraničení a MPC regulátora s ohraničeniami, ktoré zabezpečia sledovania žiadanej referencie. V prípade sledovania je pre porovnanie použitý aj jednoduchý diskrétny PID regulátor. Činnosť regulátorov je experimentálne overená a kvalitatívne vyhodnotená. Záver obsahuje zhodnotenie dosiahnutých výsledkov riadenia.

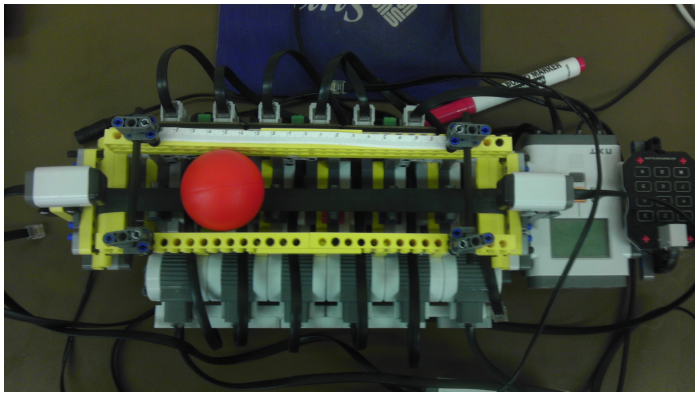
Kapitola 1

Teoretická časť

Teoretická časť je rozdelená na päť podkapitol a obsahuje všetky potrebné informácie k obsluhe dopravného flexibilného pása, a teoretický základ pre návrh viacerých typov riadení. Prvá podkapitola predstavuje jednoduchý návod od opisu zariadenia cez spojenie s PC až k spôsobu práce v prostredí MATLABU. Ďalšia sa zaoberá problematikou modelovania systému gule na naklonenej rovine. Ostatné podkapitoly popisujú teoretické odvodenia návrhu pozorovača, LQR a MPC riadenia.

1.1 Opis zariadenia a jeho ovládanie

Model flexibilného dopravného pásu (obr. 1.1) je postavený zo stavebnice Lego Mindstrom a umožňuje študentom zoznámiť sa s programovaním reálneho zariadenia. Celý model pozostáva z dvoch snímačov vzdialenosti, šiestich piestov, kostry postavenej zo stavebnice, a mikropočítača. Mikropočítačom je 32-bitový NXT Brick (obr. 1.2) so 4Kbyte-ovou Flash pamäťou a 512 Byte-ovou pamäťou RAM.



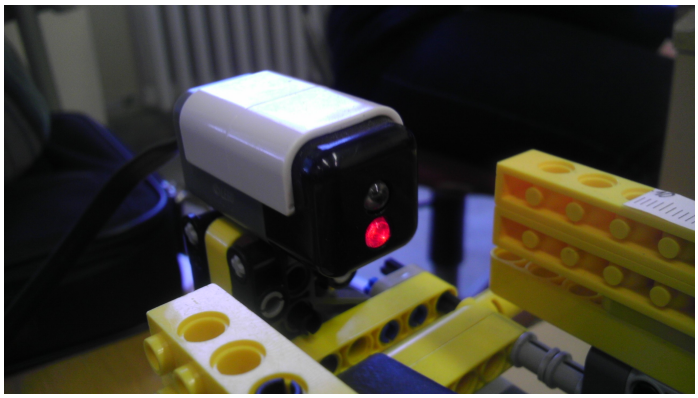
Obr. 1.1: Dopravný pás

Obr. 1.2: Mikropočítač

Základným rozhraním medzi užívateľom a riadiacou jednotkou je monochromatický maticový displej. Komunikáciu s počítačom zabezpečuje USB 2.0, prípadne Bluetooth technológia. Na spodnej strane kocky sa nachádzajú vstupné porty 1, 2, 3 a 4, ku ktorým sa pripájajú snímače. Na hornej strane sú výstupné porty A, B a C pre pripojenie akčných členov (servo motorčeky)[5]. Nad piestami sa nachádza natiahnutý gumený pás, po ktorom sa pohybuje guľa vplyvom naklonenia roviny. Toto naklonenie je spôsobené vysúvaním a zasúvaním jednotlivých piestov. Po celej dĺžke pásu sa nachádza pravítko, ktoré slúži k odčítaniu polohy gule.

Na výstavbu dopravného pása boli použité tieto snímače a akčné členy:

- **snímače vzdialenosti EOPD(*Electro-Optical Proximity Detector*):** na oboch koncoch dopravného pása sa nachádzajú senzory vzdialenosti (obr. 1.3), ktoré sú namierené proti sebe.

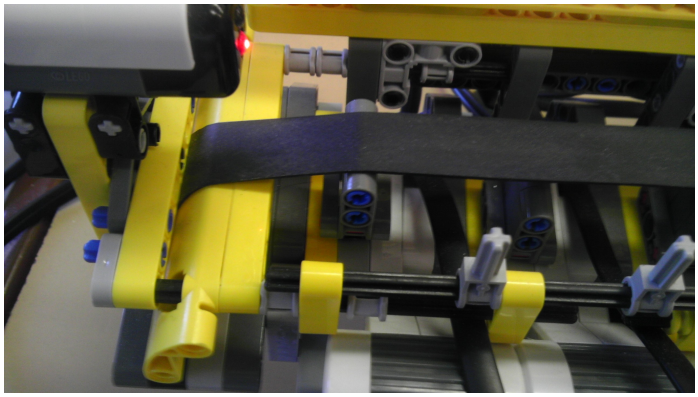


Obr. 1.3: Snímač vzdialenosti

Dôvod použitia dvoch snímačov je limitovaný maximálny dosah, ktorý je len okolo desiatich centimetrov, a dĺžka pásu je šestnásť centimetrov. Meranie vzdialenosti je nepriame a je založené na meraní odrazeného svetla od prekážky pred senzorom. Výstupom snímača

je celé číslo z rozsahu 0 až 1023. Kalibráciou je toto číslo prepočítané na vzdialenosť v centimetroch,

- **akčné členy (piesty):** medzi akčné členy patria dva servomotorčeky, ktoré sú pomocou prevodov prestavané na piesty (obr. 1.4). Vysúvaním a zasúvaním piestov dochádza k vytvoreniu naklonenej roviny vplyvom čoho sa guľa pohybuje.



Obr. 1.4: Piest

1.1.1 Spojenie zariadenia s PC pomocou USB

Na vytvorenie spojenie NXT robota a počítača s operačným systémom Windows je potrebné vykonať tieto operácie:

1. stiahnuť RWTH - Mindstorms NXT Toolbox for MATLAB verzia 4.07 a pridať cestu v MATLABE k tomuto súboru. Tento toolbox slúži na ovládanie NXT robota pomocou MATLABU,

2. stiahnuť lego USB ovládač Phantom driver verzia 1.03 a nainštalovať. Ovládač umožňuje rozpoznať robota prostredníctvom USB,

3. stiahnuť liveusb-win32 a nainštalovať podľa priloženého návodu. Doplní chýbajúci ovládač do Windowsu.

1.1.2 Spôsob práce s NXT robotom v MATLABE

Po nainštalovaní potrebných ovládačov a softvéru je robot schopný komunikovať s PC prostredníctvom MATLABU. Práca s robotom sa vykonáva prostredníctvom funkcií, ktoré sa nachádzajú

v Mindostorm NXT toolboxe [2]. Tento toolbox ponúka niekoľko základných funkcií pre komunikáciu so snímačmi, pohonmi a viaceré prídavné funkcie pre zjednodušenie vývoja vlastných aplikácií. Kód, ktorý vytvorí spojenie vyzerá takto

```
global BNB_DATA
COM_CloseNXT all
handle = COM_OpenNXT();
EOPDPorts = [SENSOR_1 SENSOR_2];
OpenEOPD(EOPDPorts(1), 'LONG', handle);
OpenEOPD(EOPDPorts(2), 'LONG', handle);
BNB_DATA.motors = [MOTOR_A, MOTOR_B];
BNB_DATA.EOPDs = EOPDPorts;
BNB_DATA.handle = handle;
```

Ako prvé sa vytvorí štruktúra BNB_DATA. Pred samotným vytvorením spojenia sa preventívne zatvoria všetky predchádzajúce komunikácie pomocou funkcie $COM\_CloseNXT$. Nové spojenie vytvorí funkcia $COM\_OpenNXT$ a uloží sa do premennej *handle*. Funkcia $OpenEOPD$ inicializuje EOPD senzory vzdialenosti. Prvý argument funkcie je konkrétny port na kocke. Argument $LONG$ sa zadáva v prípade, ak je vzdialenosť meraných objektov väčšia ako desať centimetrov. Posledný argument je spojenie *handle*. Na koniec sa do štruktúry BNB_DATA vloží ovládanie motorov ($MOTOR\_A$ vracia symbolickú 0, $MOTOR\_B$ vracia symbolickú 1), porty snímačov a spojenie *handle*. Medzi ďalšie využívané funkcie patrí $bnb\_resetPistonsToZero$, ktorá zasunie piesty do nulovej polohy. Funkcia $bnb\_movePistonsToPosition([50 \quad 50])$ vysunie piesty do polovice celého rozsahu a táto poloha pásu je aj štartovacia pre simulácie. Realizáciu vypočítaného akčného zásahu vykonáva funkcia $bnb\_createAngle$ a jej zápis je nasledovný

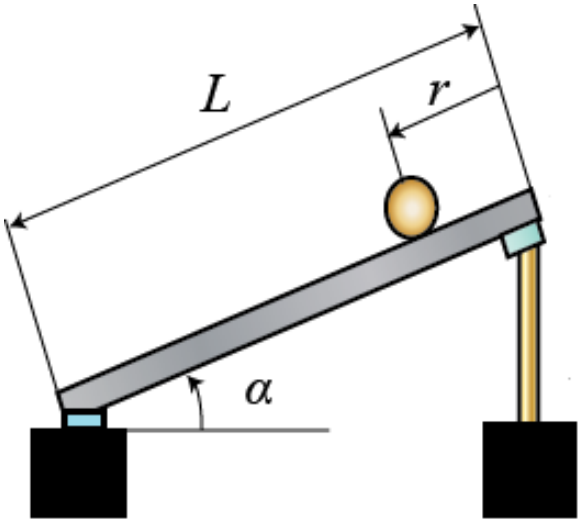
```
bnb_createAngle(angle, initial_pistons)
```

Prvým argumentom je uhol *angle*, ktorý je vypočítaný zvoleným regulátorom, a druhý argument je počiatočná poloha piestov. Každý piest je limitovaný svojou dĺžkou. Rozsah uhlov, ktoré môžu piesty vytvoriť, je od -5 po 5 stupňov.

1.2 Modelovanie systému gule na naklonenej rovine [1]

Pohyb gule po naklonenej rovine je schematický nakreslený na obrázku 1.5. Tento systém disponuje jedným vstupom, ktorým je uhol naklonenia roviny α, a jedným výstupom, pozíciou na dopravnom páse r. Systém sa chová diskrétne, pretože poloha gule aj uhol naklonenia piestov

sú snímané diskrétne v čase. Pri zmene uhla z horizontálnej polohy sa guľa vplyvom gravitácie začne pohybovať po dopravnom páse.



Obr. 1.5: Guľa na naklonenej rovine

Lagrangeová rovnica popisujúca pohyb gule na naklonenej ploche vyzerá nasledovne

$$0 = \Big(\frac{J}{R^2} + m\Big)\ddot{r} + mgsin\alpha - mr\dot{\alpha}^2, \tag{1.1}$$

(m) *hmotnosť gule* $[kg]$,

(R) *polomer gule* $[m]$,

(g) *gravitačné zrýchlenie* $\left[\frac{m}{s^2}\right]$,

(J) *moment inercie gule* (*v prípade dutej gule* $J = \frac{2mR^2}{3}$) $[kg.m^2]$,

(r) *pozícia gule* $[m]$,

(α) *uhol naklonenia* $[°]$.

Linearizáciou rovnice pre veľmi malé uhly ($\alpha \approx 0$) dostaneme rovnicu (1.2), lebo posledný člen rovnice (1.1) bude nulový a sínus malého uhla sa rovná priamo hodnote toho uhla.

$$\Big(\frac{J}{R^2} + m\Big)\ddot{r} = -mg\alpha \tag{1.2}$$

1.2.1 Prenosová funkcia

Na odvodenie prenosovej funkcie použijeme Laplacovú transformáciu. Prenosová funkcia sa rovná podielu Laplacovho obrazu vstupnej veličiny k Laplacovmu obrazu výstupnej veličiny.

$$\Big(\frac{J}{R^2} + m\Big)R(s)s^2 = -mg\Theta(s) \tag{1.3}$$

$$G(s) = \frac{R(s)}{\Theta(s)} = -\frac{mg}{(\frac{J}{R^2} + m)} \frac{1}{s^2} \tag{1.4}$$

Prenosovou funkciou (1.4) je dvojitý integrátor, ktorý je okrajovo stabilný, čo spôsobuje problémy pri riadení.

1.2.2 Stavová reprezentácia systému

Prenosová funkcia môže byť pretransformovaná do stavovej reprezentácie. Stavové premenné sú pozícia r a rýchlosť $\dot{r}$ a vstupom je uhol naklonenia α.

$$\underbrace{\begin{bmatrix} \dot{r}(t) \\ \ddot{r}(t) \end{bmatrix}}_{\dot{x}(t)} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r(t) \\ \dot{r}(t) \end{bmatrix}}_{Ax(t)} + \underbrace{\begin{bmatrix} 0 \\ \frac{mg}{(\frac{J}{R^2}+m)} \frac{2\pi}{180} \end{bmatrix} \alpha(t)}_{Bu(t)} \tag{1.5a}$$

$$y(t) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} r(t) \\ \dot{r}(t) \end{bmatrix}}_{Cx(t)} + \underbrace{\begin{bmatrix} 0 \end{bmatrix} \alpha(t)}_{Du(t)} \tag{1.5b}$$

Ak sa v matici A nachádza 1, neuvažuje sa vplyv trenia. Čím je táto hodnota menšia ako 1, tým je vplyv trenia väčší.

1.3 Pozorovač stavu

Na základe modelu reálneho zariadenia môžeme pokračovať v návrhu riadenia. V prípade návrhu pokročilejších systémov riadenia je potrebné merať všetky stavy. V praxi sa najčastejšie nedá zabezpečiť, že poznáme všetky prvky stavového vektora, a preto musíme neznáme stavy odhadovať. V prípade môjho zariadenia taktiež meriame len vzdialenosť a rýchlosť musíme odhadovať. Najvhodnejší spôsob ako stavy odhadovať, je použiť pozorovač stavu. Pozorovač je dynamický systém, ktorý na základe vstupov a výstupov zo systému vie odhadnúť stavové veličiny [4].

Majme riadený systém (1.6) v stavovej reprezentácii

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1.6a}$$

$$y(t) = Cx(t), \tag{1.6b}$$

potom odhad stavu $\hat{x}(t)$ sa bude rovnať

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L(y(t) - C\hat{x}(t)), \tag{1.7}$$

kde L predstavuje proporcionálnu spätnú väzbu, ktorá minimalizuje rozdiel medzi skutočným a odhadovaným výstupom. Chybu medzi skutočným a odhadovaným stavom vyjadruje rovnica (1.8)

$$e(t) = x(t) - \hat{x}(t), \tag{1.8}$$

a jej derivácia je definovaná rovnicou (1.9)

$$\begin{aligned} \dot{e}(t) = \dot{x}(t) - \dot{\hat{x}}(t) &= Ax(t) + Bu(t) - A\hat{x}(t) - Bu(t) - LCx(t) - LC\hat{x}(t) \\ &= A(x(t) - \hat{x}(t)) - LC(x(t) - \hat{x}(t)) \\ &= (A - LC)e(t). \end{aligned} \tag{1.9}$$

Aby sme dosiahli stabilný systém, tak zosilnenie L volíme tak, aby vlastné čísla matice $(A - LC)$ boli stabilné, čiže menšie ako nula. Takto navrhnutý pozorovač sa volá Luenbergerov.

Kalmnanov filter

Kalmanov filter je špeciálnym typom Luenbergerovho pozorovača,v ktorom účelová funkcia určitým vážením minimalizuje vplyv chyby počiatočného odhadu a chyby šumov, ktoré pôsobia na stavové rovnice a rovnice výstupu.
Majme systém (1.10) zaťažený nejakým náhodným šumom,

$$\dot{x}(t) = Ax(t) + \xi_x(t) \qquad \xi_x(t) \backsim N(0, Q) \tag{1.10a}$$

$$x(0) = \overline{x}_0 + \xi_{x_0} \qquad \xi_{x_0} \backsim N(0, P_0) \tag{1.10b}$$

$$y(t) = Cx(t) + \xi(t) \qquad \xi(t) \backsim N(0, R), \tag{1.10c}$$

kde šumy ξ_x, ξ_{x_0}, ξ majú normálne rozdelenie, nulovú strednú hodnotu a kovariančné matice sú Q, P_0, R. Ak poznáme charakteristiky týchto šumov, tak Kalmanov filter nám navrhne najlepší pozorovač, ktorý minimalizuje vplyv týchto chýb. Účelová funkcia vyzerá takto

$$\begin{aligned} J = \underbrace{\frac{1}{2}(x(0) - \overline{x}_0)^T P_0^{-1}(x(0) - \overline{x}_0)}_{G} + \frac{1}{2}\int_0^{t_f} \Big(&(\dot{x}(t) - Ax(t))^T Q^{-1}(\dot{x}(t) - Ax(t)) + \\ &+ (y(t) - Cx(t))^T R^{-1}(y(t) - Cx(t)) \Big) dt. \end{aligned} \tag{1.11}$$

Ak si zvolíme

$$u(t) = \dot{x}(t) - Ax(t) \iff \dot{x}(t) = Ax(t) + u(t), \tag{1.12}$$

Hamiltonián tejto účelovej funkcie má podobu,

$$H = \frac{1}{2}u(t)^T Q^{-1} u(t) + \frac{1}{2}(y(t) - Cx(t))^T R^{-1}(y(t) - Cx(t)) + \lambda(t)^T(Ax(t) + u(t)), \tag{1.13}$$

kde λ je Langrangeov multiplikátor, cez ktorý sú pridané obmedzenia. Následne sa aplikuje podmienka pre optimálne riadenie

$$\frac{\partial H}{\partial u} = Q^{-1}u(t) + \lambda(t) = 0 \implies u(t) = -Q\lambda(t), \tag{1.14}$$

pre $\dot{\lambda}$ vyhovuje rovnica (1.15)

$$\dot{\lambda}(t) = -\frac{\partial H}{\partial x} = C^T R^{-1} y(t) - C^T R^{-1} C x(t) - A^T \lambda(t), \tag{1.15}$$

koncový stav je voľný a platí

$$\lambda(t_f) = \frac{\partial G}{\partial x_f} = 0, \tag{1.16}$$

a počiatočný stav je

$$\lambda(t_0) = -\frac{\partial G}{\partial x_0} = -P_0^{-1} x(0) + P_0^{-1}\overline{x}_0 \implies x(0) = \overline{x}_0 - P_0\lambda_0. \tag{1.17}$$

Príjmeme kvalifikovaný odhad (predpoklad), že odhad stavu $\hat{x}$ v čase t_f na základe údajov do času t sa rovná

$$\hat{x}(t|t_f) = z(t) - P(t)\lambda(t). \tag{1.18}$$

Derivácia odhadu stavu (1.18) je vyjadrená v rovnici (1.19).

$$\dot{\hat{x}}(t) = \dot{z}(t) - \dot{P}(t)\lambda(t) - P(t)\dot{\lambda}(t) = A\hat{x}(t) - Q\lambda(t) \tag{1.19}$$

Po dosadení všetkých veličín a odseparovaní všetkých z na jednu stranu, zvyšku na druhú a ich následným položením rovným nule dostávame dve diferenciálne rovnice (1.20) a (1.21).

$$\dot{z}(t) = P(t)C^T R^{-1} y(t) - P(t)C^T R^{-1} C z(t) + Az(t) \qquad z(0) = \overline{x}_0 \tag{1.20}$$

$$\dot{P}(t) = AP(t) + P(t)A^T + Q - P(t)C^T R^{-1} C P(t) \qquad P(0) = P_0 \tag{1.21}$$

Rovnica (1.21) je Riccatiho rovnica. Pre $t = t_f$ môžeme napísať

$$\hat{x}(t|t_f) = A\hat{x}(t|t_f) - Q\lambda(t) \implies \hat{x}(t_f|t_f) = A\hat{x}(t_f|t_f) - Q\lambda(t). \tag{1.22}$$

Po zohľadnení podmienky (1.16) potom platí

$$\hat{x}(t_f|t_f) = z(t_f). \tag{1.23}$$

Odhad stavu priebežne sa realizuje podľa vzťahu (1.24),

$$\dot{z}(t) = Az(t) + \underbrace{P(t)C^T R^{-1}}_{L(t)}(y(t) - Cz(t)) \iff \dot{\hat{x}}(t) = A\hat{x}(t) + L(t)(y(t) - Cz(t))$$

$$L(t) = P(t)C^T R^{-1}, \tag{1.24}$$

kde L predstavuje časovo premenlivé zosilnenie Kalmanovho filtra. P dostaneme riešením Riccatiho rovnice (1.21).

1.4 LQR riadenie

LQR riadenie znamená, že náš riadený systém je lineárny a účelová funkcia kvadratická. V stavovom opise systém vyzerá takto,

$$\dot{x(t)} = Ax(t) + Bu(t) \qquad x(0) = x_0, \tag{1.25}$$

kde A je matica dynamiky a vo všeobecnosti uvažujeme n stavov a m riadení a poznáme počiatočné podmienky, ktoré sú pevné. A chceli by sme minimalizovať kvadratické kritérium (1.26a),

$$J = \frac{1}{2}x_{t_f}^T Q_{t_f} x_{t_f} + \int_0^{t_f} (x(t)^T Q x(t) + u(t)^T R u(t))dt \tag{1.26a}$$
$$Q, Q_{t_f} \geq 0, R > 0, t_f \textit{známe},$$

ktoré bude nejakým spôsobom penalizovať stavy, nejakou spotrebou penalizovať energiu, ktorú budeme chcieť dať do systému, aby sme stavy dostali niekam k nule. A zároveň bude penalizovať nejakým spôsobom koncový stav. Úlohou LQR bude nájsť taký zákon riadenia u, ktorý bude funkciou stavu $u(t) = f(x(t), t) \quad x(t_f) \to 0$.
Ak chceme aby x v koncovom stave išlo k nule, tak hovoríme o LQR regulácii. V prípade LQR sledovania sa snažíme sledovať nejakú žiadanú hodnotu [4].

1.4.1 LQR regulácia

V prípade regulácie ideme z nejakého nenulového počiatočného stavu do nuly. Hamiltonova funkcia má tvar

$$H = \frac{1}{2}x(t)^T Q x(t) + \frac{1}{2}u(t)^T R u(t) + \lambda(t)^T (Ax(t) + Bu(t)). \tag{1.27}$$

Optimálne riadenie bez obmedzení získame

$$\frac{\partial H}{\partial u} = Ru(t) + B^T\lambda(t) = 0 \implies u(t) = -R^{-1}B^T\lambda(t) \quad Q_{t_f}, Q, R = \textit{symetrické}. \tag{1.28}$$

Pre $\dot{\lambda}$ vyhovuje rovnica (1.29).

$$\dot{\lambda}(t) = -\frac{\partial H}{\partial x} = -Qx(t) - A^T\lambda(t) \tag{1.29}$$

Pre koncový stav platí

$$\lambda(t_f) = \frac{\partial G}{\partial x_{t_f}} = Q_{t_f} x(t_f). \tag{1.30}$$

Rovnicu (1.25), po dosadení za optimálne u spolu s rovnicou (1.29) môžeme prepísať do vektorového tvaru ako

$$\begin{bmatrix} \dot{x} \\ \dot{\lambda} \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} x \\ r \end{bmatrix}. \tag{1.31}$$

U takéhoto systému rovníc sa dá riešenie napísať ako riešenie, ktoré závisí od x v čase t a λ v čase t takto

$$\begin{bmatrix} x(t_f) \\ \lambda(t_f) \end{bmatrix} = \begin{bmatrix} \Phi_{11}(t,t_f) & \Phi_{12}(t,t_f) \\ \Phi_{21}(t,t_f) & \Phi_{22}(t,t_f) \end{bmatrix} \begin{bmatrix} x(t) \\ \lambda(t) \end{bmatrix}. \tag{1.32}$$

Z vektorového tvaru (1.32) vieme odvodiť $\lambda(t)$ po dosadení za $\lambda(t_f)$ z rovnice (1.30) ako (1.33)

$$\lambda(t) = [\Phi_{22}(t,t_f) - Q_{t_f}\Phi_{12}(t,t_f)]^{-1} - [Q_{t_f}\Phi_{11}(t,t_f) - \Phi_{21}(t,t_f)]x(t). \tag{1.33}$$

Z riešenia tohto homogénneho systému rovníc vyplýva, že ak máme lineárny systém a kvadratickú účelovú funkciu, tak λ je nejakou lineárnou funkciou stavov (1.34).

$$\lambda(t) = P(t)x(t) \tag{1.34}$$

Derivácia funkcie (1.34) podľa času je

$$\dot{\lambda}(t) = \dot{P}(t)x(t) + P(t)\dot{x}(t). \tag{1.35}$$

Po dosadení rovníc (1.29), (1.25), (1.34) dostaneme

$$-Qx(t) - A^TPx(t) = \dot{P}(t)x(t) + P(Ax(t) + Bu(t)), \tag{1.36}$$

$u(t)$ sa dosadí z rovnice (1.28)

$$-Qx(t) - A^TPx(t) = \dot{P}(t)x(t) + PAx(t) + PBR^{-1}B^TPx(t). \tag{1.37}$$

Po vyňatí $x(t)$ pred zátvorku dostaneme

$$(-Q - A^TP - \dot{P}(t) + PA + PBR^{-1}B^TP)x(t) = 0. \tag{1.38}$$

A ak má táto rovnica platiť pre ľubovoľný stav, tak celá zátvorka sa musí rovnať nule. Čím dostaneme kvadratickú maticovú diferenciálnu rovnicu (1.39a), ktorá sa nazýva Riccatiho rovnica.

$$\dot{P}(t) + A^TP + PA - PBR^{-1}B^TP = -Q \tag{1.39a}$$

$$P(t) = P(t)^T > 0 \tag{1.39b}$$

$$P(t_f) = Q_{t_f} \tag{1.39c}$$

Koncovú podmienku (1.39c) dostaneme porovnaním rovníc (1.30) a (1.34). Dosadením do rovnice (1.28) za $\lambda(t)$ z rovnice (1.34) je optimálne riadenie vyjadrené ako spätná väzba k stavu (1.40)

$$u(t) = -\underbrace{R^{-1}B^T P(t)}_{K(t)} x(t) = -K(t)x(t). \tag{1.40}$$

$K(t)$ je časovo premenné, ktoré získame riešením Riccatiho rovnice (1.39a), ktorá je tiež časovo premenná. Ak čas t_f pôjde do nekonečna, tak náš stav v nekonečne pôjde do nuly a diferenciálna Riccatiho rovnica sa zmení na algebraickú (1.41)

$$A^T P + PA - PBR^{-1}B^T P = -Q. \tag{1.41}$$

1.4.2 LQR sledovanie

LQR riadenie vedie k proporcionálnej stavovej spätnej väzbe. Pri použití proporcionálnych regulátorov v spätnej väzbe dochádza pri zmenách žiadaných veličín a pri výskyte porúch na ľubovoľnom mieste uzavretého regulačného obvodu k vzniku trvalej regulačnej odchýlky. Spôsob ako zabezpečiť odstránenie trvalej regulačnej odchýlky je LQR s integračnou činnosťou [4]. Majme lineárny riaditeľný a pozorovateľný systém (1.42).

$$\dot{x}(t) = Ax(t) + Bu(t) \quad x(0) = x_0 \tag{1.42a}$$

$$y(t) = Cx(t) \tag{1.42b}$$

Nech $w(t)$ je vektor žiadaných veličín. Úlohou sledovania je zabezpečiť, aby sa odchýlka $e(t)$ (1.43) medzi žiadanou a výstupnou veličinou asymptoticky blížila k nule.

$$e(t) = w(t) - y(t) \tag{1.43}$$

Majme účelovú funkciu (1.44), ktorú chceme minimalizovať,

$$J = \underbrace{\frac{1}{2}e_{t_f}^T Q_{yt_f} e_{t_f}}_{G} + \int_{t_0}^{t_f} \Big(e(t)^T Q_y e(t) + u(t)^T R u(t)\Big) dt, \tag{1.44}$$

kde čas t_f je známy, Q_{yt_f} a Q_y sú reálne symetrické kladne semidefinitné matice a R je reálna symetrická kladne definitná matica. Po dosadení za odchýlku (1.43) do účelovej funkcie (1.44) získame

$$\begin{aligned} J =& \frac{1}{2}(w_{t_f} - y_{t_f})^T Q_{yt_f}(w_{t_f} - y_{t_f}) + \frac{1}{2}\int_{t_0}^{t_f} \Big((w(t) - y(t))^T Q_y (w(t) - y(t)) + \\ &+ u(t)^T R u(t)\Big) dt. \end{aligned} \tag{1.45}$$

Hamiltonova funkcia má tvar

$$H = \frac{1}{2}(w(t) - Cx(t))^T Q_{yt_f}(w(t) - Cx(t)) + \frac{1}{2}u(t)^T Ru + (t)\lambda(t)^T(Ax + Bu). \tag{1.46}$$

Optimálne riadenie získame opäť ako

$$\frac{\partial H}{\partial u} = Ru(t) + B^T\lambda(t) = 0 \implies u(t) = -R^{-1}B^T\lambda(t). \tag{1.47}$$

Pre $\dot{\lambda}$ vyhovuje rovnica (1.48),

$$\dot{\lambda}(t) = -\frac{\partial H}{\partial x} = C^T Q_y w(t) - C^T Q_y Cx(t) - A^T\lambda(t), \tag{1.48}$$

pre koncový stav platí

$$\lambda(t_f) = \frac{\partial G}{\partial x_{t_f}} = -C^T Q_{yt_f} w(t_f) + C^T Q_{yt_f} Cx(t_f). \tag{1.49}$$

Podobne ako v prípade regulácie rovnica (1.34) môžeme odvodiť λ ako funkciu stavu mínus nejaký vektor γ vzhľadom k tomu, že tam máme už aj žiadanú veličinu

$$\lambda(t) = P(t)x(t) - \gamma(t). \tag{1.50}$$

Deriváciou rovnice (1.50) podľa času dostaneme (1.51)

$$\dot{\lambda}(t) = \dot{P}(t)x(t) + P(t)\dot{x}(t) - \dot{\gamma}(t). \tag{1.51}$$

Výsledok spojenia rovníc (1.42b), (1.48), (1.50), (1.51) je

$$C^T Q_y w(t) - C^T Q_y Cx(t) - A^T\lambda(t) = \dot{P}(t)x(t) + P(t)(Ax(t) + Bu(t)) - \dot{\gamma}(t). \tag{1.52}$$

Následne do (1.52) aplikujeme optimálne u z rovnice (1.47)

$$\begin{aligned} &C^T Q_y w(t) - C^T Q_y Cx(t) - A^T P(t)x(t) + A^T\gamma(t) = \dot{P}(t)x(t) + P(t)Ax(t) - \\ &- P(t)BR^{-1}B^T P(t)x(t) + P(t)BR^{-1}B^T\gamma(t) - \dot{\gamma}(t). \end{aligned} \tag{1.53}$$

Aby rovnica platila pre ľubovoľné $x(t)$ a $w(t)$ musí $P(t)$ vyhovovať rovnici (1.54)

$$\dot{P}(t) = -P(t)A - A^T P(t) + P(t)BR^{-1}B^T P(t) - C^T Q_y C, \tag{1.54}$$

a $\gamma(t)$ rovnici (1.55)

$$\dot{\gamma(t)} = [P(t)BR^{-1}B^T - A^T]\gamma - C^T Q_y w. \tag{1.55}$$

Koncové podmienky (1.56) a (1.57) vychádzajú z porovnania rovníc (1.49) a (1.50).

$$P(t_f) = C^T Q_{yt_f} C \tag{1.56}$$

$$\gamma(t_f) = C^T Q_{yt_f} w(t_f) \tag{1.57}$$

Optimálnym riešením problému sledovania je riadenie (1.58)

$$u(t) = -R^{-1}B^T\lambda(t) = -\underbrace{R^{-1}B^T P(t)}_{K_1} x(t) + \underbrace{R^{-1}B^T P(t)\gamma(t)}_{K_2}. \tag{1.58}$$

Optimálne riadenie je regulátor s dvomi stupňami voľnosti, kde K_1 zabezpečí stavy do nuly a K_2 sledovanie

$$u(t) = -K_1 x(t) - K_2 f(w). \tag{1.59}$$

1.5 Prediktívne riadenie

Prediktívne riadenie nazývané tiež MPC, alebo APC (*Advanced Process Control*) je v súčasnosti najviac využívaná nová metóda v priemysle. Je prirodzeným pokračovaním LQR riadenia. MPC je založené na diskrétnych modeloch procesov, a preto všetky odvodenia a vzťahy budú v diskrétnej oblasti.

Prediktívne riadenia sa vyznačuje viacerými črtami:

- používame model procesu pri návrhu regulátora (regulátor musí poznať model procesu),
- známa budúca trajektória žiadanej hodnoty (poznáme ako sa bude vyvíjať trajektória žiadanej hodnoty určitý čas dopredu),
- postupnosť budúcich akčných zásahov je vypočítaná na základe minimalizáciu účelovej funkcie vzhľadom na budúce trajektórie prírastkov riadenia a regulačnej odchýlky,
- realizovaný je len prvý akčný zásah a v ďalšej perióde vzorkovania sa celý proces minimalizácie opakuje.

Najdôležitejšia vlastnosť, prečo sa MPC používa, je možnosť pracovať s obmedzeniami na vstupné a výstupné veličiny. Ďalej dokáže spracovávať poruchy, tým že v každej perióde vzorkovania robí optimalizáciu. Prediktívne riadenie sa dá dobré nastavovať a môžeme používať rozličné typy modelov. K nevýhodám patrí, že potrebuje výkonnejší procesor a pri zle navrhnutom MPC môžu byť problémy so stabilitou [4].

1.5.1 Modely

Budeme uvažovať modely s prírastkom riadenia, čím sa zabezpečí nulová trvalá regulačná odchýlka. Je to spôsobené tým, že model procesu je doplnený o integrátor. Prírastok riadenia Δu je definovaný

$$\Delta u(k) = u(k) - u(k-1) \implies u(k) = \Delta u(k) + u(k-1). \tag{1.60}$$

Prechodovú funkciu (1.61a) môžeme do tvaru s prírastkom riadenia (1.60) upraviť prenásobením celej rovnice deltou (1.61b) a dostaneme (1.61c).

$$A(z^{-1})y(k) = B(z^{-1})u(k) \tag{1.61a}$$

$$\Delta = 1 - z^{-1} \tag{1.61b}$$

$$(1 - z^{-1})A(z^{-1})y(k) = B(z^{-1})(1 - z^{-1})u(k) \tag{1.61c}$$

Po úprave dostaneme žiadaný tvar prenosovej funkcie (1.62)

$$\tilde{A}(z^{-1})y(k) = B(z^{-1})\Delta u(k). \tag{1.62}$$

V prípade stavovej reprezentácie (1.63) úprava na tvar s prírastkom riadenia vyzerá nasledovne.

$$x(k+1) = Ax(k) + Bu(t) \tag{1.63a}$$

$$y(k) = Cx(k) \tag{1.63b}$$

Po dosadení za $u(k)$ z rovnice (1.60) dostaneme

$$x(k+1) = Ax(k) + Bu(t) = Ax(k) + B\Delta u(t) + Bu(k-1). \tag{1.64}$$

Teraz vytvoríme nový vektor stavov rozšírený o minulé hodnoty riadenia (1.65)

$$\bar{x}(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \quad \bar{x}(k+1) = \begin{bmatrix} x(k+1) \\ u(k) \end{bmatrix}. \tag{1.65}$$

A nový tvar stavovej reprezentácie s prírastkom riadenia je

$$\bar{x}(k+1) = \underbrace{\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}}_{\bar{A}} \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} + \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_{\bar{B}} \Delta u(k) \tag{1.66a}$$

$$y(k) = \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_{\bar{B}} \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}. \tag{1.66b}$$

1.5.2 Účelová funkcia

Účelová funkcia pri prediktívnom riadení bude mať tvar (1.67).

$$J = \sum_{i=N_1}^{N_2} [P\hat{y}(k+i) - w(k+i)]^2 + \lambda \sum_{i=1}^{N_u} [\Delta u(k+i-1)]^2 \tag{1.67}$$

Kde N_1 je minimálny horizont, $N2$ maximálny horizont a N_u je horizont riadenia, ktorý znižuje výpočtovú náročnosť metódy (1.71), keď predpokladáme, že prírastky riadenia medzi N_u a N_2 sú nulové.

$$\Delta u(k+i-1) = 0, \quad N_u < i \leq N_2 \tag{1.68}$$

Pomocou P určitým spôsobom vážime y do budúcnosti.

1.5.3 Odvodenie prediktora

Pri odvodení prediktora používame stavový model (1.66). Predikcia vektora stavov a výstupov je definovaná rovnicami (1.69) a (1.70).

$$\begin{aligned}
\bar{x}(k+1) &= \bar{A}\bar{x}(k) + \bar{B}\Delta u(k) && \text{(1.69a)}\\
\bar{x}(k+2) &= \bar{A}\bar{x}(k+1) + \bar{B}\Delta u(k+1) = \bar{A}^2\bar{x}(k) + \bar{A}\bar{B}\Delta u(k) + \bar{B}\Delta u(k+1) && \text{(1.69b)}\\
&\vdots \\
\bar{x}(k+N_2) &= \bar{A}^{N_2}\bar{x}(k) + \bar{A}^{N_2-1}\bar{B}\Delta u(k) + \ldots + \bar{B}\Delta u(k+N_2-1) && \text{(1.69c)}
\end{aligned}$$

$$\begin{aligned}
\bar{y}(k+1) &= \bar{C}\bar{x}(k+1) && \text{(1.70a)}\\
\bar{y}(k+2) &= \bar{C}\bar{x}(k+2) = \bar{C}\Big(\bar{A}\bar{x}(k+1) + \bar{B}\Delta u(k+1)\Big) && \text{(1.70b)}\\
&\vdots \\
\underbrace{\bar{y}(k+N_2)}_{Y} &= \bar{C}\bar{x}(k+N_2) = \bar{C}\Big(\bar{A}\bar{x}(k+N_2-1) + \bar{B}\Delta u(k+N_2-1)\Big) && \text{(1.70c)}
\end{aligned}$$

Spojením predikcií stavov a výstupov dostaneme maticový tvar (1.71),

$$Y = \underbrace{\begin{bmatrix} \bar{C}\bar{A} \\ \bar{C}\bar{A}^2 \\ \vdots \\ \bar{C}\bar{A}^{N_2} \end{bmatrix}}_{Y_0} x(k) + \underbrace{\begin{bmatrix} \bar{C}\bar{B} & 0 & \ldots & \ldots & 0 \\ \bar{C}\bar{A}\bar{B} & \bar{C}\bar{B} & 0 & \ldots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \bar{C}\bar{B} & 0 \\ \bar{C}\bar{A}^{\bar{N_2}-1}\bar{B} & \ldots & & \ldots & \bar{C}\bar{B} \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_2-1) \end{bmatrix}}_{\tilde{u}} \begin{matrix} N_u \\ \downarrow \end{matrix}, \tag{1.71}$$

$$Y = Y_0 + G\tilde{u}, \tag{1.72}$$

kde Y je vektor predikcii výstupov do budúcnosti, Y_0 je vektor takzvaných voľných odoziev, G je matica, ktorá závisí iba od vlastností procesu, a $\tilde{u}$ je vektor prírastkov riadenia do budúcnosti. Ak by sme predpokladali, že niektoré Δu sú od pozície N_u až do konca nulové, tak by sa počet stĺpcov matice G zmenšil o N_u a znížila by sa výpočtová náročnosť.

Odvodenie optimálneho zákona riadenia bez ohraničení

Kritérium, ktoré chceme minimalizovať môžeme prepísať do tvaru

$$\min_{\tilde{u}} J = \sum [\hat{y}(i) - w(i)]^2 + \lambda \sum \Delta u(i)^2 = (Y - W)^T (Y - W) + \lambda \tilde{u}^T \tilde{u}. \tag{1.73}$$

Po dosadení za Y z rovnice (1.72) dostaneme

$$\begin{aligned} \min_{\tilde{u}} J &= (G\tilde{u} + (Y_0 - W))^T (G\tilde{u} + (Y_0 - W)) + \lambda \tilde{u}^T \tilde{u} \\ &= \tilde{u}^T G^T (Y_0 - W) + (Y_0 - W)^T G\tilde{u} + (Y_0 - W)^T (Y_0 - W) + \\ &+ \tilde{u}^T (G^T G + \lambda I)\tilde{u}. \end{aligned} \tag{1.74}$$

Keď chceme minimalizovať účelovú funkciu, tak platí

$$\begin{aligned} \frac{\partial J}{\partial \tilde{u}} = 0 &= G^T (Y_0 - W) + G^T (Y_0 - W) + 2(G^T G + \lambda I)\tilde{u} \\ &= 2G^T (Y_0 - W) + 2(G^T G + \lambda I)\tilde{u}. \end{aligned} \tag{1.75}$$

Vektor postupností prírastkov riadenia z predchádzajúcej rovnice sa rovná

$$\tilde{u} = (G^T G + \lambda I)^{-1} G^T (W - Y_0) \tag{1.76}$$

a z neho v zmysle prediktívneho riadenia zoberieme prvý prírastok a pripočítame ho k poslednému riadeniu, ktoré sme mali, a dostaneme $u(k)$, ktoré aplikujeme do procesu a celý postup zopakujeme v každej ďalšej perióde vzorkovania. V rovnici (1.76) sa mení len Y_0, ktorý závisí od aktuálneho stavu, a budúce žiadané hodnoty W. MPC regulátor získava voľnú odozvu Y_0 v každej perióde vzorkovania tak, že simuluje proces dopredu, keď všetky prírastky riadenia do budúcnosti sú nulové. Maticu G vypočítava z rovnice (1.72) tak, že uvažuje že systém je v ustálenom stave (stavy sú nulové) a celá voľná odozva je rovná nule. Skoková zmena v čase k je rovná jednej a všetky ostatné prírastky riadenia sú nulové, čím získa prvý stĺpec matice G, a v ňom sú všetky potrebné hodnoty na vytvorenie celej matice (1.71).

1.5.4 Obmedzenia

Možnosť pracovať s obmedzeniami patrí k najväčšej prednosti prediktívneho riadenia. V praxi sa najčastejšie využívajú ohraničenia na vstupy, prípadne je ohraničená rýchlosť ich zmeny. Stretávame sa aj s ohraničeniami na výstupy z procesu a to hlavne z bezpečnostného hľadiska. Kvadratickú účelovú funkciu (1.74) môžeme pretransformovať do všeobecnej formulácie kvadratického programovania,

$$\min_{\tilde{u}} \quad 2g^T\tilde{u} + \tilde{u}^T H\tilde{u} \tag{1.77a}$$

$$g^T = G^T(Y_0 - W) \tag{1.77b}$$

$$H = G^T G + \lambda I, \tag{1.77c}$$

ku ktorej môžeme pridať obmedzenia vzhľadom na budúce prírastky riadenia

$$A\tilde{u} \leq b. \tag{1.78}$$

V prípade, že chceme mať ohraničenia na prírastky riadenia $\Delta u_{min} \leq \Delta u \leq \Delta u_{max}$ matice A a b sú definované,

$$A = \begin{bmatrix} I \\ -I \end{bmatrix} \quad b = \begin{bmatrix} 1 \otimes \Delta u_{max} \\ -1 \otimes \Delta u_{min} \end{bmatrix}, \tag{1.79}$$

kde $\otimes$ označuje Kroneckerov produkt dvoch vektorov. A ak ohraničenia na vstupy $u_{min} \leq u \leq u_{max}$ A, b sú

$$A = \begin{bmatrix} L \\ -L \end{bmatrix} \quad b = \begin{bmatrix} 1 \otimes u_{max} - 1 \otimes u(t-1) \\ -1 \otimes u_{min} + 1 \otimes u(t-1) \end{bmatrix}, \tag{1.80}$$

a v prípade ohraničení na výstupe $y_{min} \leq y \leq y_{max}$.

$$A = \begin{bmatrix} G \\ -G \end{bmatrix} \quad b = \begin{bmatrix} y_{max} - y_0 \\ -y_{min} + y_0 \end{bmatrix}. \tag{1.81}$$

1.5.5 MPC regulácia

Všeobecná formuláciu MPC problému stavovej regulácie s lineárnymi ohraničeniami je definovaná,

$$\min \quad \sum_{k=0}^{N-1} x_k^T Q_x x_k + u_k^T Q_u u_k \tag{1.82a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k \tag{1.82b}$$

$$x_k = x(t) \tag{1.82c}$$

$$x_k \in \mathcal{X} \tag{1.82d}$$

$$u_k \in \mathcal{U}, \tag{1.82e}$$

kde stavový model reprezentuje ohraničenia v tvare rovníc (1.82b). Druhé ohraničenia v tvare rovnosti (1.82c) hovorí, aká je počiatočná podmienka optimalizačného problému (terajšie stavové meranie). Ohraničenia na stavové a vstupné veličiny sú v tvare nerovníc (1.82d), (1.82e) [3]. N je predikčný horizont, $Q_x \in \mathbb{R}^{n_x \times n_x}$ a $Q_u \in \mathbb{R}^{n_u \times n_u}$ sú váhové matice. Takáto formulácia optimalizačného problému minimalizuje hodnotu rozdielu stavu a akčného zásahu od nulovej hodnoty. MPC stavovú reguláciu pre predikčný horizont N môžeme rozpísať pomocou rovníc (1.83).

$$\begin{aligned} \min \quad & x_k^T Q_x x_k + u_k^T Q_u u_k + x_{k+1}^T Q_x x_{k+1} + u_{k+1}^T Q_u u_{k+1} + x_{k+2}^T Q_x x_{k+2} + \\ & + u_{k+2}^T Q_u u_{k+2} + \ldots + x_{k+N-1}^T Q_x x_{k+N-1} + u_{k+N-1}^T Q_u u_{k+N-1} \end{aligned} \tag{1.83a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k \tag{1.83b}$$

$$\vdots$$

$$x_{k+N-1} = Ax_{k+N-2} + Bu_{k+N-2} \tag{1.83c}$$

$$x_k = x(t) \tag{1.83d}$$

$$x_{min} \leq x_k \leq x_{max} \tag{1.83e}$$

$$\vdots$$

$$x_{min} \leq x_{k+N-1} \leq x_{max} \tag{1.83f}$$

$$u_{min} \leq u_k \leq u_{max} \tag{1.83g}$$

$$\vdots$$

$$u_{min} \leq u_{k+N-1} \leq u_{max} \tag{1.83h}$$

Ohraničenia na stavy a vstupy od (1.83e) do (1.83h) sa dajú upraviť do tvaru (1.84).

$$Hx_k \leq K \tag{1.84a}$$

$$\vdots$$

$$Hx_{k+N-1} \leq K \tag{1.84b}$$

$$Lu_k \leq M \tag{1.84c}$$

$$\vdots$$

$$Lu_{k+N-1} \leq M \tag{1.84d}$$

Formulácia MPC regulácie (1.83) s použitím úprav (1.84) nie je vhodná pre bežne používané solvery a je možná transformácia do tvaru všeobecnej formulácie kvadratického programovania (1.85). Prvým krokom je zadefinovanie vektora optimalizovaných premenných z a následne všetkých ostatných matíc.

$$\min \quad z^T P z + 2Q^T z + R \tag{1.85a}$$

$$\text{s.t.} \quad Az \leq B \tag{1.85b}$$

$$Gz = \tilde{H} \tag{1.85c}$$

$$z = \begin{bmatrix} u_k \\ \vdots \\ u_{k+N-1} \\ x_k \\ \vdots \\ x_{k+N-1} \end{bmatrix} \quad P = \begin{bmatrix} Q_u & \dots & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & \dots & 0 \\ 0 & \dots & Q_u & 0 & \dots & 0 \\ 0 & \dots & 0 & Q_x & \dots & 0 \\ 0 & \dots & 0 & 0 & \ddots & 0 \\ 0 & \dots & 0 & 0 & \dots & Q_x \end{bmatrix}$$

$$B = \begin{bmatrix} K \\ \vdots \\ K \\ M \\ \vdots \\ M \end{bmatrix} \quad A = \begin{bmatrix} 0 & \dots & 0 & H & \dots & 0 \\ 0 & \dots & 0 & 0 & \ddots & 0 \\ 0 & \dots & 0 & 0 & \dots & H \\ L & \dots & 0 & 0 & \dots & 0 \\ 0 & \ddots & 0 & 0 & \dots & 0 \\ 0 & \dots & L & 0 & \dots & 0 \end{bmatrix}$$

$$\tilde{H} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ xt \end{bmatrix} \quad G = \begin{bmatrix} -B & \dots & 0 & -A & I & \dots & 0 \\ 0 & \ddots & 0 & 0 & \ddots & \ddots & 0 \\ 0 & \dots & B & 0 & \dots & A & I \\ 0 & \dots & 0 & I & 0 & \dots & 0 \end{bmatrix}$$

Takáto formulácia MPC sa nazýva aj riedka, lebo jednotlivé matice obsahuje veľa núl. Výhodou je, že s takými maticami sa ľahko vykonávajú jednotlivé matematické operácie. Nevýhodou je väčší počet optimalizovaných premenných. Druhá často používaná formulácia sa volá hustá a dá sa odvodiť tak, že vektor stavov vyjadríme ako funkciu počiatočného stavu a vektora vstupov (1.86) a dosadíme do účelovej funkcie.

$$x_k = x(t) \tag{1.86a}$$

$$x_{k+1} = Ax_k + Bu_k \tag{1.86b}$$

$$
\begin{aligned}
x_{k+2} &= Ax_{k+1} + Bu_{k+1} && (1.86c)\\
&= A^2x_k + ABu_k + Bu_{k+1}\\
&\vdots\\
x_{k+N-1} &= Ax_{k+N-2} + Bu_{k+N-2} && (1.86d)\\
&= A^{N-1}x_k + A^{N-2}Bu_k + \ldots + ABu_{k+N-3} + Bu_{k+N-2}
\end{aligned}
$$

Rovnice (1.86) sa dajú prepísať do vektorového tvaru (1.87).

$$
X = \underbrace{\begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{N-1} \end{bmatrix}}_{\tilde{A}} x(t) + \underbrace{\begin{bmatrix} 0 & 0 & \ldots & 0 \\ B & 0 & \ldots & 0 \\ AB & B & \ldots & 0 \\ \vdots & \ddots & \ddots & 0 \\ A^{N-2}B & A^{N-3}B & \ldots & B \end{bmatrix}}_{\tilde{B}} \underbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-2} \end{bmatrix}}_{U} \tag{1.87}
$$

Podobným spôsobom sa pretransformujú aj ohraničenia v tvare rovníc. Účelová funkcie bude mať tvar (1.88), ktorý je ekvivalentný s tvarom (1.83a).

$$
\min \quad X^T\tilde{Q}_xX + U^T\tilde{Q}_uU \tag{1.88}
$$

X je vektor stavov, U vektor vstupov a $Q_x \in \mathbb{R}^{n_x \times n_x}$, $Q_u \in \mathbb{R}^{n_u \times n_u}$ sú váhové matice.

$$
X = \begin{bmatrix} x_k \\ x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N-1} \end{bmatrix} \quad U = \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N-1} \end{bmatrix} \tag{1.89}
$$

$$
\tilde{Q}_x = \begin{bmatrix} Q_x & 0 & 0 & 0 & 0 \\ 0 & Q_x & 0 & 0 & 0 \\ 0 & 0 & Q_x & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & Q_x \end{bmatrix} \quad \tilde{Q}_u = \begin{bmatrix} Q_u & 0 & 0 & 0 & 0 \\ 0 & Q_u & 0 & 0 & 0 \\ 0 & 0 & Q_u & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & Q_u \end{bmatrix} \tag{1.90}
$$

Po dosadení vektorového tvaru (1.87) do účelovej funkcie (1.88) dostaneme

$$\begin{aligned}&(\tilde{A}x(t)+\tilde{B}U)^T\tilde{Q}_x(\tilde{A}x(t)+\tilde{B}U)+U^T\tilde{Q}_uU=(x(t)^T\tilde{A}^T+U^T\tilde{B}^T)\tilde{Q}_x(\tilde{A}x(t)+\\&+\tilde{B}U)+U^T\tilde{Q}_uU=x(t)^T\tilde{A}^T\tilde{Q}_x\tilde{A}x(t)+x(t)^T\tilde{A}^T\tilde{Q}_x\tilde{B}U+U^T\tilde{B}^T\tilde{Q}_x\tilde{A}x(t)+\\&+U^T\tilde{B}^T\tilde{Q}_x\tilde{B}U+U^T\tilde{Q}_uU=x(t)^T\tilde{A}^T\tilde{Q}_x\tilde{A}x(t)+2x(t)^T\tilde{A}^T\tilde{Q}_x\tilde{B}U+\\&+U^T(\tilde{B}^T\tilde{Q}_x\tilde{B}+\tilde{Q}_u)U\end{aligned} \tag{1.91}$$

a výsledná hustá formulácia účelovej funkcie v tvare kvadratického programovania je

$$\min \quad U^T\underbrace{(\tilde{B}^T\tilde{Q}_x\tilde{B}+\tilde{Q}_u)}_{P}U+2\underbrace{x(t)^T\tilde{A}^T\tilde{Q}_x\tilde{B}}_{Q^T}U+\underbrace{x(t)^T\tilde{A}^T\tilde{Q}_x\tilde{A}x(t)}_{R}. \tag{1.92}$$

Úprava ohraničení v tvare rovnosti (rovnice od (1.83b) do (1.83d)) sa vykoná tak, že do ohraničení na stavy v tvare nerovníc (rovnice od (1.84a) až (1.84b)) sa dosadia rovnice (1.86).

$$Hx_k \leq K \tag{1.93a}$$

$$H(Ax_k+Bu_k) \leq K \tag{1.93b}$$

$$H(A^2x_k+ABu_k+Bu_{k+1}) \leq K \tag{1.93c}$$

$$\vdots$$

$$H(A^{N-1}x_k+A^{N-2}Bu_k+\ldots+ABu_{k+N-3}+Bu_{k+N-2}) \leq K \tag{1.93d}$$

Predchádzajúce úpravy zabezpečia, že všetky ohraničenia v tvare rovníc sa pretransformujú na ohraničenia na vstupné veličiny v tvare nerovníc. Vo vektorovom tvare sú definované

$$\begin{bmatrix}0&0&0&\ldots&0\\HB&0&0&\ldots&0\\HAB&HB&0&\ddots&0\\\vdots&\ddots&\ddots&\ddots&0\\HA^{N-2}B&HA^{N-3}B&\ldots&HB&0\\L&0&0&\ldots&0\\0&L&0&\ldots&0\\0&0&L&\ddots&0\\\vdots&\ddots&\ddots&\ddots&0\\0&0&\ldots&0&L\end{bmatrix}\begin{bmatrix}u_k\\u_{k+1}\\u_{k+2}\\\vdots\\u_{k+N-1}\end{bmatrix}\leq\begin{bmatrix}K\\K\\K\\\vdots\\K\\M\\M\\M\\\vdots\\M\end{bmatrix}-\begin{bmatrix}H\\HA\\HA^2\\\vdots\\HA^{N-1}\\0\\0\\0\\\vdots\\0\end{bmatrix}x(t).$$

Najväčšou prednosťou hustej formulácie je menší počet optimalizovaných premenných, čím sa výrazne znižuje výpočtová náročnosť. Nevýhodou sú štrukturálne matice v účelovej funkcii a v obmedzeniach, ktoré sa invertujú omnoho ťažšie ako v prípade riedkej formulácie.

1.5.6 MPC sledovanie

Cieľom sledovanie je zabezpečiť, aby sa riadená veličina dostala na nejakú žiadanú hodnotu. V tomto prípade už nechceme minimalizovať rozdiel stavov od nejakej hodnoty ale výstupov. V účelovej funkcii (1.82a) sa preto nahradí vektor stavov x za vektor výstupov y a pridá sa ohraničenia na výstup v tvare rovnosti (1.94c).

$$\min \quad \sum_{k=0}^{N-1} y_k^T Q_y y_k + u_k^T Q_u u_k \tag{1.94a}$$

$$\begin{aligned} \text{s.t.} \quad x_{k+1} &= Ax_k + Bu_k && (1.94b)\\ y_k &= Cx_k + Du_k && (1.94c)\\ x_k &= x(t) && (1.94d)\\ x_k &\in \mathcal{X} && (1.94e)\\ u_k &\in \mathcal{U} && (1.94f) \end{aligned}$$

Tvar účelovej funkcie (1.94a) je ekvivalentný s tvarom (1.95),

$$\min \quad \sum_{k=0}^{N-1} \|(Q_y(y_k - 0)\|_p + \|Q_u(u_k - 0)\|_p, \tag{1.95}$$

ktorý vyjadruje snahu minimalizovať rozdiel výstupu a akčného zásahu od nulovej referencie. V prípade nenulovej referencie výstupnej veličiny sa účelová funkcia upraví do tvaru (1.96).

$$\min \quad \sum_{k=0}^{N-1} \|(Q_y(y_k - y_{ref})\|_p + \|Q_u(u_k - 0)\|_p \tag{1.96}$$

Takto definovaná účelová funkcia nie je však vhodná na dosiahnutie nulovej trvalej regulačnej odchýlky. Dôvodom je, že MPC regulátor sa snaží minimalizovať rozdiel výstupnej veličiny od nenulovej referencie, ale zároveň minimalizovať aj rozdiel akčného zásahu od nulovej referencie. Tu dochádza ku konfliktu, lebo nie je možné dostať výstup na nenulovú žiadanú hodnotu, ak je nulový akčný zásah. Jedným z riešení tohto problému je rozšírenie stavovej reprezentácie o integrátor (1.97). Integračná činnosť sa zabezpečí pridaním predchádzajúceho akčného zásahu do stavového vektora a tvar rozšíreného stavového modelu je definovaný rovnicami (1.98).

$$\Delta u_k = u_k - u_{k-1} \tag{1.97}$$

$$\underbrace{\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix}}_{\tilde{x}_{k+1}} = \underbrace{\begin{bmatrix} A & B \\ 0 & I \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix} B \\ I \end{bmatrix}}_{\tilde{B}} \underbrace{\Delta u_k}_{\tilde{u}_k} \tag{1.98a}$$

$$y_k = \underbrace{\begin{bmatrix} C & D \end{bmatrix}}_{\tilde{C}} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \underbrace{D}_{\tilde{D}} \Delta u_k \tag{1.98b}$$

Zavedením premenných s vlnovkou je stavový model (1.98) ekvivalentný modelu (1.99).

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k + \tilde{B}\tilde{u}_k \tag{1.99a}$$

$$y_t = \tilde{C}\tilde{x}_t + \tilde{D}\tilde{u}_t \tag{1.99b}$$

Použitím modelu rozšíreného o integrátor (1.99) dostaneme takzvanú Δu formuláciu prediktívneho riadenia v tvare (1.100),

$$\min \quad \sum_{k=0}^{N-1} \|(Q_y(y_k - y_{ref})\|_p + \|Q_{\Delta u}\Delta u_k\|_p \tag{1.100a}$$

$$\begin{aligned}
\text{s.t.} \quad \tilde{x}_{k+1} &= \tilde{A}\tilde{x}_k + \tilde{B}\Delta u_k && \text{(1.100b)} \\
y_k &= \tilde{C}\tilde{x}_k + \tilde{D}\tilde{u}_k && \text{(1.100c)} \\
\tilde{x}_k &= \begin{bmatrix} x(t) \\ u_{k-1} \end{bmatrix} && \text{(1.100d)} \\
\tilde{H}_{xu}\tilde{x}_k &\leq \tilde{K} && \text{(1.100e)} \\
H_y y_t &\leq K_y, && \text{(1.100f)}
\end{aligned}$$

ktorá patrí k najpoužívanejším, a dá sa upraviť do všeobecnej formulácie kvadratického programovania podobnými úpravami, ako bolo podrobne opísané v predchádzajúcej kapitole.

Kapitola 2

Praktická časť

Praktická časť sa začína opisom navrhnutého spôsobu kalibrácie a prepočtu výstupov zo snímačov na vzdialenosť v centimetroch a pokračuje úpravou stavového modelu. Čo možno najpresnejší model je potrebný pre návrh kvalitného riadenia. Zároveň pokročilejšie metódy riadenia potrebujú poznať všetky prvky stavového vektora. Tento problém je vyriešený pomocou Kalmanovho filtra. Hlavnou časťou je návrh viacerých typov riadenia na reálnom zariadení. Ako prvé a najjednoduchšie je riadenia pomocou diskrétneho PID regulátora. Pokročilejším riadením je potom LQR a LQR s integračnou činnosťou. Posledným a najpokročilejším je MPC, ktorého návrh je aj hlavným cieľom práce. Celé riadenie procesu bude prebiehať pomocou S-funkcií, ktoré budú generovať akčný zásah podľa príslušných stratégii riadenia. Vstupom do funkcie bude aktuálna pozícia gule na páse a v prípade sledovania aj žiadaná hodnota. V každej perióde vzorkovania sa daná funkcia zavolá a na základe vstupov a zvolenej stratégie riadenia vypočíta akčný zásah, ktorý sa pošle do procesu.

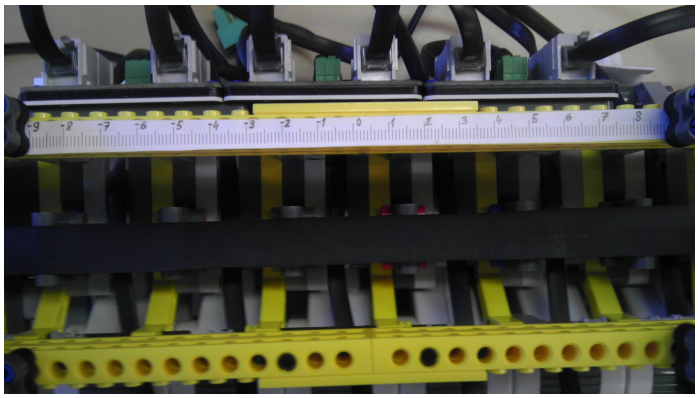
2.1 Kalibrácia snímačov vzdialenosti

Výstupovom EOPD snímača vzdialenosti je celé číslo z intervalu od 0 po 1023, ktoré pre používateľa nemá žiadnu výpovednú hodnotu. Funkcia, ktorá vracia toto celé číslo je *GetEOPD* a jej zápis vyzerá nasledujúco

```
[throwaway read1] = GetEOPD(port, handle)
```

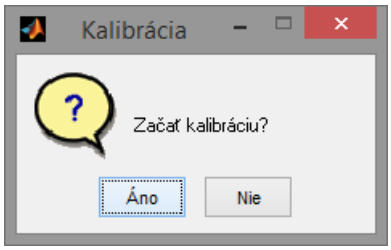
Vstupom do funkcie sú port, do ktorého je snímač zapojený, a premenná *handle*. Prvým výstupom je hodnota vzdialenosti kalibrovaná pomocou vstavanej funkcie *CalibrateEOPD*, ktorá nie je vhodná pre dané podmienky. A druhým výstupom je konkrétne celé číslo zo zadaného intervalu.

Toto celé číslo je nutné prepočítať na vzdialenosť podľa pravítka (obr. 2.1), ktoré sa nachádza na dopravnom páse.



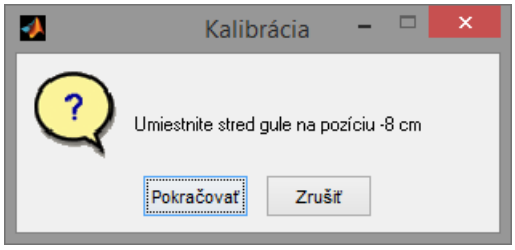
Obr. 2.1: Pravítko

V strede pravítka sa nachádza nula a na každú stranu je osem centimetrov. Kladný smer napravo od nuly a záporný smer na ľavo. Vzhľadom k tomu, že tieto snímače vzdialenosti sú dosť citlivé na intenzitu osvetlenia v miestnosti, bude potrebné zariadenie kalibrovať častejšie. Pre kalibráciu som vytvoril jednoduchý skript *kalibracia_aut.m*. Po spustení skriptu sa zobrazí dialógový box (obr. 2.2) s otázkou, či užívateľ chce vykonať kalibráciu. Po odkliknutí možnosti Áno je

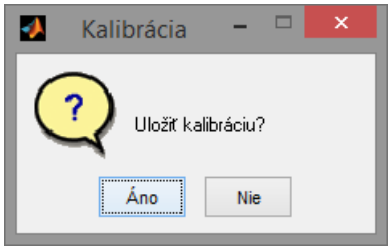


Obr. 2.2: Spustenie kalibrácie

užívateľ vyzývaný k tomu, aby guľu posúval po centimetroch v celom rozsahu pravítka (obr. 2.3). Na záver sa skript opýta, či má byť nová kalibrácia uložená (obr. 2.4). Počas kalibrácie

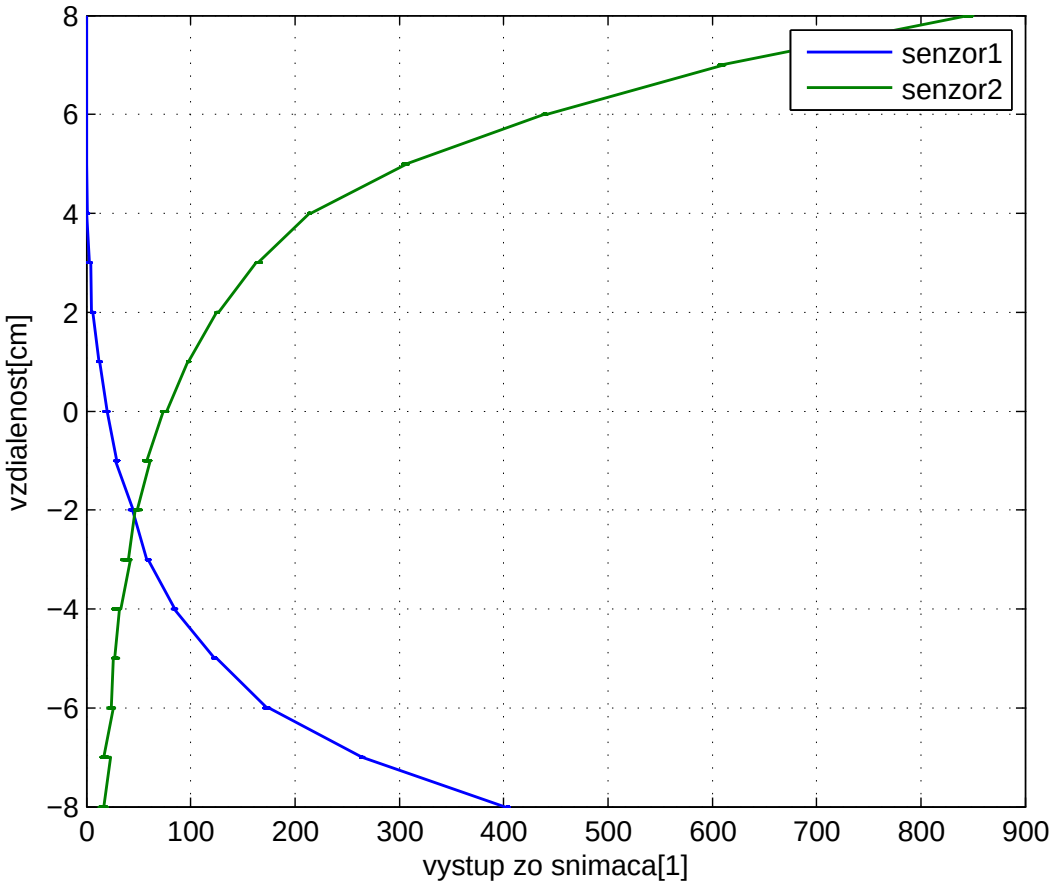


Obr. 2.3: Kalibrácia



Obr. 2.4: Uloženie kalibrácie

sa vytvorí matica údajov, ktorá obsahuje reálne vzdialenosti a prislúchajúce výstupy snímačov. Závislosť výstupov od vzdialeností (obr. 2.5) sa aproximuje vhodnou funkciou, ktorá bude slúžiť na prepočet výstupu zo snímačov na pozíciu v centimetroch. Na obrázku 2.5 je vidieť, že senzory už nie sú v stopercentnom stave a interval výstupných hodnôt je značne menší, hlavne u prvého senzora.



Obr. 2.5: Závislosť výstupu snímačov od vzdialenosti

Na základe priebehu funkcií je najvhodnejšia aproximácia racionálnou lomenou funkciou. Najjednoduchšia racionálna lomená funkcia je podiel dvoch lineárnych funkcií (2.1),

$$y = \frac{ax + c}{x - b}, \tag{2.1}$$

ktorá je v tomto tvare nelineárna, a na to aby sme mohli získať parametre a, b, c metódou najmenších štvorcov ju musíme upraviť do tvaru (2.2c).

$$(x - b)y = ax + c \tag{2.2a}$$

$$xy - by = ax + c \tag{2.2b}$$

$$xy = ax + by + c \tag{2.2c}$$

Následne metódou najmenších štvorcov získame najlepšiu aproximáciu a, b, c.

$$\min \quad F(a, b, c) = \sum_{i=1}^{n} (x_i y_i - a x_i - b y_i - c)^2 \tag{2.3}$$

$$\frac{\partial F}{\partial a} = 0 \quad \frac{\partial F}{\partial b} = 0 \quad \frac{\partial F}{\partial c} = 0 \tag{2.4}$$

Zápis v maticovom tvare vyzerá nasledovne

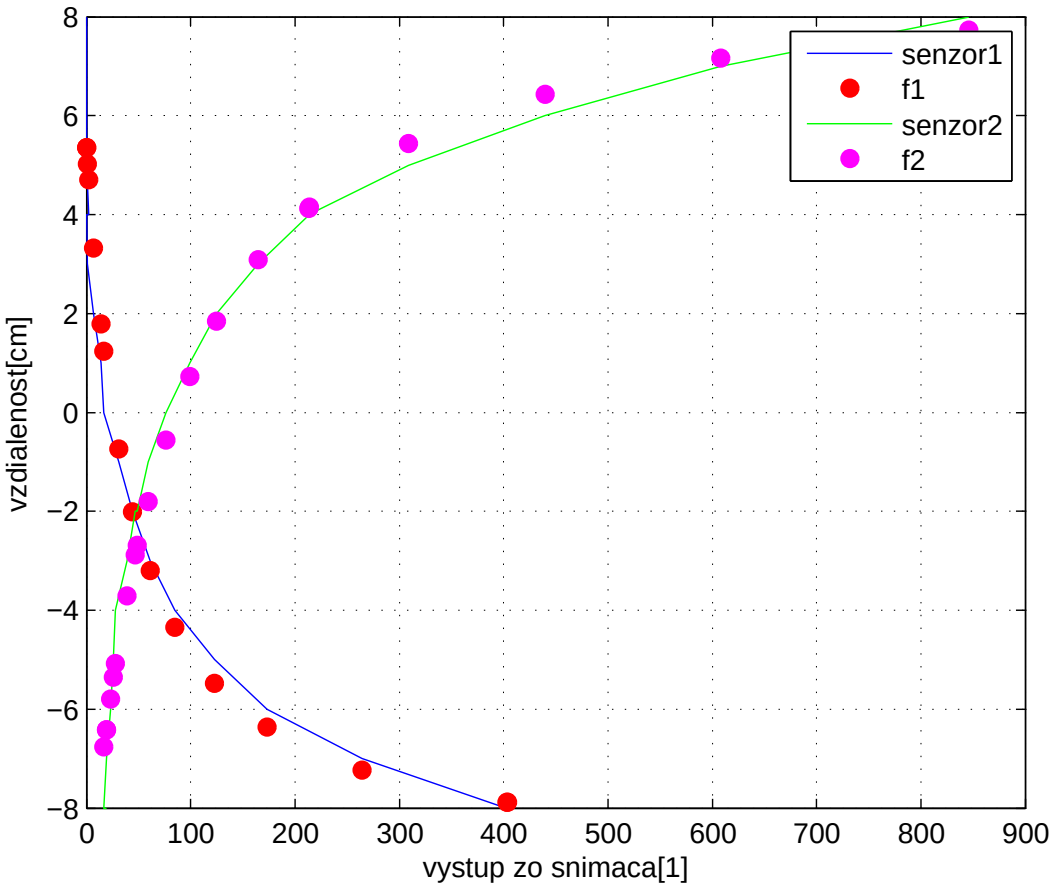
$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum x_i^2 y_i \\ \sum x_i y_i^2 \\ \sum x_i y_i \end{bmatrix}. \tag{2.5}$$

Vyriešením sústavy rovníc (2.5) dostaneme parametre a, b, c. Aproximovaná vzdialenosť sa vypočíta dosadením parametrov do rovnice (2.1). Rovnaký postup sa použije pre obidva senzory. Porovnanie obidvoch závislostí a ich aproximácii je na obrázku (2.6). Tieto snímače majú najväčšiu citlivosť v blízkosti snímača a so zväčšujúcou vzdialenosťou sa presnosť zmenšuje. Váhovaním som zabezpečil, aby sa väčšia váha dávala tomu snímaču, pri ktorom sa nachádza detekovaný objekt. Váhovanie som vykonal tak, že som vyrátal priemer $y_{1,2}$ (2.6) a použíl som ho pri výpočte váhovanej vzdialenosti y_v (2.7),

$$y_{1,2} = \frac{y_1 + y_2}{2} \tag{2.6}$$

$$y_v = y1\Big(1 - \frac{y_{1,2} + 8}{16}\Big) + y_2\Big(\frac{y_{1,2} + 8}{16}\Big), \tag{2.7}$$

kde y_1 a y_2 sú aproximované vzdialenosti zo snímačov a dĺžka pásu je 16 centimetrov.



Obr. 2.6: Porovnanie závislostí s aproximáciami

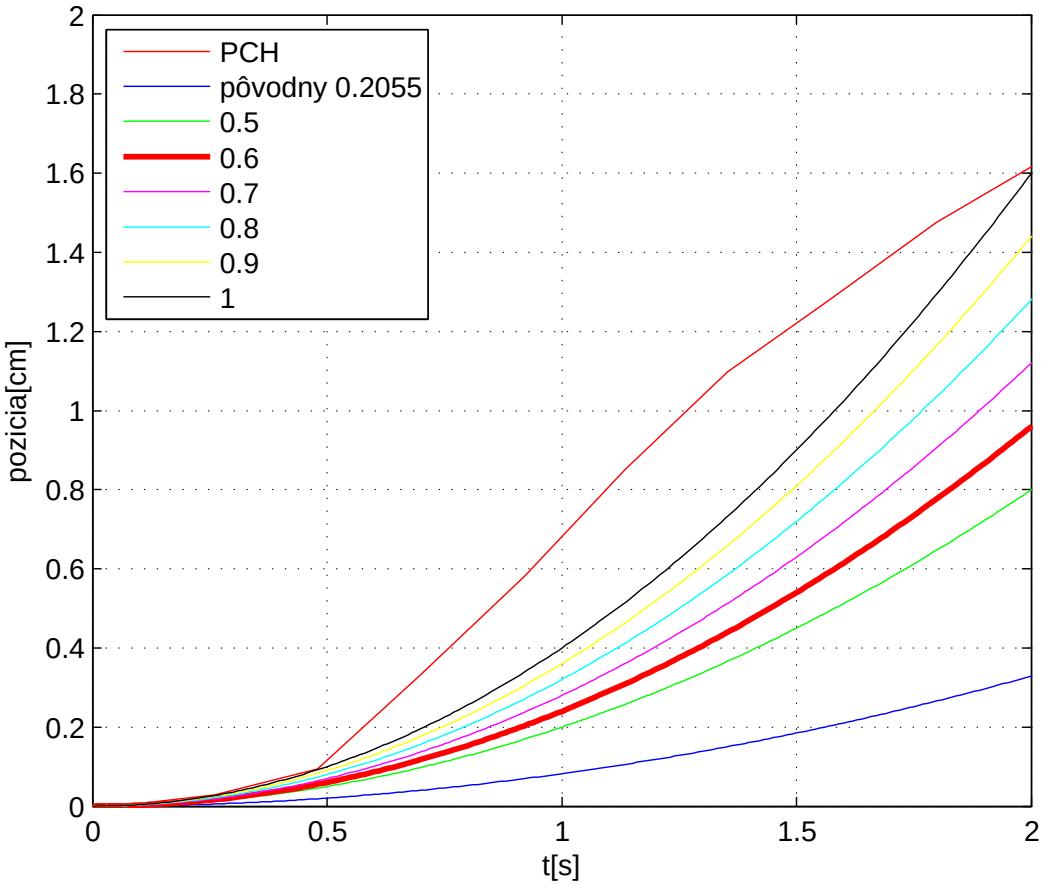
Spôsob celého prepočtu pozície na centimetre je, že každú periódu vzorkovania sa zavolá vytvorená funkcia *getBallPosition*, v ktorej sa prerátajú výstupy zo snímačov na konkrétne vzdialenosti v centimetroch. Na prepočet sa použije funkcia získaná aproximáciou údajov z kalibrácie. Na záver sa výsledná hodnota získa váhovaním obidvoch vzdialeností.

2.2 Úprava stavového modelu

Stavový model (1.5) neuvažuje trenie, ktorému sa v prípade reálneho zariadenia nevyhneme. Prvým spôsobom ako vplyv trenia zakomponovať do modelu je znížiť hodnotu jedna v matici A. Druhý spôsob je snažiť sa zmeniť člen (2.8), ktorý v sebe zahŕňa moment inercie a gravitačné zrýchlenie tak, aby čo najpresnejšie vystihoval správanie gule na dopravnom páse.

$$\frac{mg}{(\frac{J}{R^2}+m)}\frac{2\pi}{180} = 0.2055 \tag{2.8}$$

V mojom prípade som nechal maticu A nezmenenú a snažil som sa upraviť člen zahrňujúci moment inercie a gravitačné zrýchlenie. Vykonal som niekoľko skokových zmien pre rôzne umiestnenie gule zmenou uhla α na hodnoty jeden, dva a tri stupne. Vplyv trenia bol značný, lebo niekedy aj pri uhle rovnom trom stupňom sa guľa nepohla. Nakoniec som všetky získané odozvy na skokové zmeny normoval a získanú prechodovú charakteristiku vykreslil do grafu (2.7) spolu s pôvodným modelom, a inými testovanými modelmi s rôznou hodnotou člena (2.8) v stavovej reprezentácií.



Obr. 2.7: Vykreslenie prechodových charakteristík

Na základe priebehov na grafe a správania zariadenia som si zvolil konštantu v stavovej rovnici rovnú 0.6. Na toto číslo som prišiel experimentálne po mnohých pokusoch návrhov už samotného riadenia. Stavová reprezentácia po úprave vyzerá takto

$$\underbrace{\begin{bmatrix} \dot{r} \\ \ddot{r} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix}}_{Ax} + \underbrace{\begin{bmatrix} 0 \\ 0.6 \end{bmatrix} \alpha}_{Bu} \tag{2.9a}$$

$$y = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix}}_{Cx} + \underbrace{\begin{bmatrix} 0 \end{bmatrix} \alpha}_{Du}. \tag{2.9b}$$

Celý systém sa chová diskrétne, a preto je potrebné stavový model previesť do diskrétneho tvaru. Diskretizovaný stavový model, ktorý som používal pri návrhoch riadenia je definovaný rovnicami (2.10).

$$x_{k+1} = \underbrace{\begin{bmatrix} 1 & 0.15 \\ 0 & 1 \end{bmatrix}}_{A_d} x_k + \underbrace{\begin{bmatrix} 0.0067 \\ 0.09 \end{bmatrix}}_{B_d} u_k \tag{2.10a}$$

$$y_k = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C_d} x_k + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{D_d} u_k \tag{2.10b}$$

Perióda vzorkovania Ts je 0.15 sekundy.

2.3 Vytvorenie Kalmanovho filtra

Pri návrhu pokročilejších systémov riadenia (LQR, MPC) je potrebné poznať všetky stavy. V prípade dopravného pásu sa meria len vzdialenosť a rýchlosť je nutné odhadovať. Optimálnym spôsobom je využiť Kalmanov filter. Návrh diskrétneho Kalmanovho filtra v MATLABE realizuje príkaz *dlqe*, ktorý vracia takú maticu zosilnení L, že optimálny odhad stavu v ďalšom kroku sa uskutoční pomocou rovníc (2.11),

$$x_k = x_{k-1} + L(y_k - C_d x_{k-1} - D_d u_k) \tag{2.11a}$$

$$x_{k+1} = A_d x_k + B_d u_k, \tag{2.11b}$$

kde A_d, B_d, C_d a D_d sú matice stavového opisu diskrétneho systému (2.10). Zápis rovníc (2.11) vo vektorovom tvare a návrh pozorovača v MATLABE zobrazuje kód,

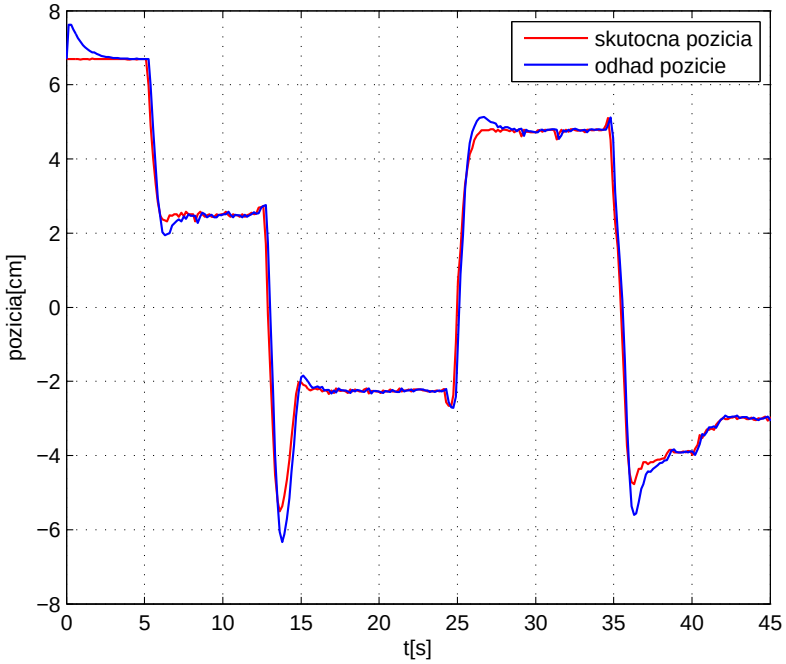
```
Qq = diag([100, 1]);
Rr = diag(1);
Gg = diag([1, 1]);
L = dlqe (Ad , Gg , Cd , Qq , Rr ) ;
Ae = [eye(2)-L*Cd zeros(2, 2); zeros(2, 2) Ad];
Be = [L zeros(2, 1); zeros(2, 1) Bd];
z = Ae*z + Be*[previous_position; previous_angle];
x = z(1:2);
```

kde matice Q_q a R_r sú váhové matice a G_g je diagonálna jednotková matica. Pomocou váhových matíc sa dajú penalizovať stavy systému (zvyšovaním Q_q, alebo znižovaním R_r) a vstupy systému (zvyšovaním R_r, alebo znižovaním Q_q). Najdôležitejšie je nájsť vhodný pomer týchto matíc. V mojom prípade boli najvhodnejšie matice (2.12).

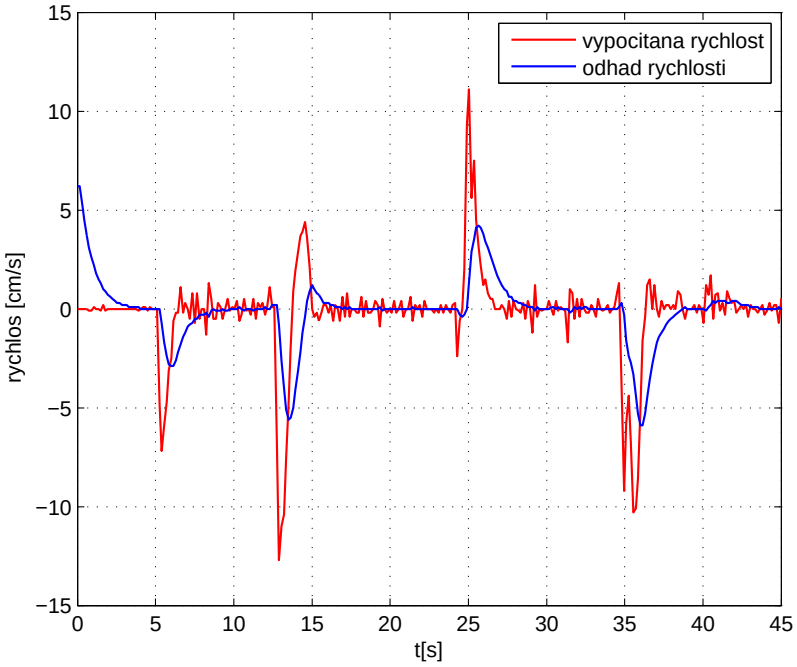
$$Q_q = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.12a}$$

$$R_r = 1 \tag{2.12b}$$

Pri inicializácii je počiatočný odhad stavu z nulový. Premenná z je definovaná ako perzistentná a v každom volaní funkcie sa prepisuje novou hodnotou. Implementácia pozorovača v S-funkcií je realizovaná tak, že každú periódu vzorkovania sa vypočíta aktuálna hodnota vektora z na základe predchádzajúcej hodnoty. Z vektora z sa vyberú aktuálne stavy x a použijú sa na výpočet akčného zásahu podľa zvolenej stratégie riadenia. Pre lepšie zachytenie odhadu stavov je implementácia navrhnutá tak, že prvých desať periód vzorkovania sa len odhaduje bez riadenia a až potom sa začína riadiť. Vhodnosť použitých váhových matíc som overil experimentálne. Na grafoch (2.8) a (2.9) sú vykreslené porovnania skutočnej pozície a rýchlosti s ich odhadmi. Skutočnú rýchlosť som vyrátaval, ako rozdiel aktuálnej a predchádzajúcej pozície podelený časom jednej periódy vzorkovania.



Obr. 2.8: Odhad pozície



Obr. 2.9: Odhad rýchlosti

2.4 Návrh a implementácia stavovej regulácie

Cieľom stavovej regulácie je dostať stavy v nejakom konečnom čase do nuly. Táto časť práce sa venuje návrhu a implementácii LQR stavového regulátora bez ohraničení a MPC stavového regulátora s ohraničeniami. Pri návrhu obidvoch regulátorov som použil rovnaké váhové matice Q a R pre zabezpečenie rovnakých podmienok.

2.4.1 LQR bez ohraničení

Implementácia LQR stavového regulátora bez ohraničení zabezpečuje S-funckia *bnb_lqr*(*position*). Funkcia ma jeden vstup pozíciu (*position*) a jeden výstup akčný zásah (*angle*), ktorý sa posiela do procesu. Štruktúra S-funkcie je nasledovná

```
function angle = bnb_lqr(position)
persistent previous_position previous_angle z
if nargin==0
    previous_position = bnb_getBallPosition()/100;
    previous_angle = 0;
    z = [0; 0; 0; 0];
    return
```

```
end
    position = position/100;
```

Na začiatku sú zadefinované perzistentné premenné predchádzajúca pozícia (*previous_position*), predchádzajúci uhol (*previous_angle*) a vektor z pre pozorovač. Pri inicializácii sú im priradené nulové začiatočné podmienky a v prípade poslednej známej pozície je to aktuálna poloha. Predchádzajúca aj aktuálna pozícia je premenené na metre, vzhľadom k tomu, že celý model uvažuje pozíciu v metroch. Potom nasleduje definovanie matíc spojitého stavového modelu a ich prevedenie do diskrétnej formy.

```
    A = [0 1; 0 0];
    B = [0 ; 0.6];
    C = [1 0];
    D = 0;
    Ts = 0.15;
    sys=c2d(ss(A,B,C,D),Ts);
    Ad = sys.a;
    Bd = sys.b;
    Cd = sys.c;
    Dd = sys.d;
```

S-funkcia ďalej obsahuje pozorovač bližšie popísaný v kapitole 2.3. Potom, čo poznáme všetky stavy, tak zosilnenie stavovej spätnej väzby z riešenia LQ problému dostanem v prípade diskrétneho procesu takto,

```
Q = diag([100; 1])/0.08;
R = 0.1/5;
K = dlqr(Ad, Bd, Q, R);
```

kde K je matica zosilnenie a akčný zásah, ktorý je výstupom funkcie, sa výráta ako

```
angle = - K*x;
```

Na konci S-funkcie sa ešte prepíšu perzistentné premenné novými hodnotami pre volanie funkcie v ďalšej perióde vzorkovania.

```
previous_position = position;
previous_angle = angle;
end
```

2.4.2 MPC s ohraničeniami

Pri návrhu MPC stavovej regulácie s ohraničeniami som vychádzal z MPC formulácie definovanej rovnicami (1.82). Predikčný horizont som zvolil $N = 5$. Váhové matice Q a R som najskôr škáloval a potom som dal väčšiu váhu na stavy rovnako ako u LQR. Ohraničenie na prvý stav, pozíciu som definoval v tvare $-7.4 \leq x_{1k} \leq 7.4$ (vzdialenosť v centimetroch). Ohraničenie na vstup je $-5 \leq u_k \leq 5$ (uhol v stupňoch). S-funkcia zabezpečujúca implementáciu prediktívneho stavového regulátora s ohraničeniami *mpc_sr(position)* začína podobne ako v prípade LQR. Na začiatku sa definujú perzistentné premenné, stavové matice a navrhne sa pozorovač. Návrh prediktívneho regulátora začína zistením počtov stavov nx, vstupov nu a výstupov ny, definovaným váhových matíc a predikčného horizontu.

```
[nx, nu] = size(B);
ny = size(C, 1);
N = 5;
Q = diag([100; 1])/0.08;
R = 0.1/5;
```

Potom sa definujú optimalizované premenné xs, us, ys pomocou *sdpvar*

```
xs = cell(N+1, 1);
us = cell(N, 1);
ys = cell(N, 1);
for k = 1:1:N+1
    xs{k} = sdpvar(nx, 1);
end
for k = 1:1:N
    us{k} = sdpvar(nu, 1);
end
for k = 1:1:N
    ys{k} = sdpvar(ny, 1);
end
```

a nasleduje zadefinovanie ohraničení a účelovej funkcie.

```
cst = [];
obj = 0;
for k = 1:1:N
    cst = [cst; xs{k+1} == Ad*xs{k} + Bd*us{k}; ys{k} == Cd*xs{k} + Dd*us{k}];
    cst = [cst; -5 <= us{k} <= 5];
```

```
    cst = [cst; -0.074 <= xs{k}(1) <= 0.074];
    obj = obj + ((xs{k})'*Q*(xs{k}) + (us{k})'*R*(us{k}));
end
```

Výpočet akčného zasáhu realizuje *optimizer*, kde vstupujú ohraničenia *cst*, potom účelová funkcia *obj*, nastavenia *options*, počiatočný stav $xs1$ a prvý akčný zásah $us1$. Ohraničenia sú definované na akčný zásah *us* a prvý stav, ktorým je pozícia. Cieľom je zabezpečiť, aby guľa nenarážala do mechanických zábran na koncoch pásu. Príkaz *controller* vráti optimálny akčný zásah pre aktuálny odhadnutý stav x. Premenná b imformuje o riešiteľnosti optimalizačného problému

```
options = sdpsettings('solver', 'gurobi','verbose', 0, 'cachesolvers', 1);
controller = optimizer(cst, obj, options,[xs{1}], us{1});
[angle, b, c] = controller{[x]};
if b~=0
   error('neriesitelny problem')
end
```

a ak by b nebolo nulové, problém by bol neriešiteľný a vypíše sa error. Do premennej angle sa vloží vypočítaný akčný zásah a nakoniec sa prepíšu perzistentné premenné rovnako ako v prípade LQR. Na výpočet akčného zásahu je možné použiť aj príkaz $solvesdp(cst, obj)$, ktorý je však omnoho pomalší a nestihol by za čas jednej periódy vzorkovania vyriešiť optimalizačný problém.

2.5 Návrh a implementácia sledovania referencie

Cieľom sledovania je zabezpečiť žiadanú hodnotu v nejakom konečnom čase. Ako prvé som použil na dosiahnutie žiadanej referencie diskrétny PID regulátor, potom LQR regulátor s integračnou činnosťou bez ohraničení a na záver MPC regulátor s integračnou činnosťou spolu s ohraničeniami.

2.5.1 PID regulátor

Proporcionálno-integračno-derivačný (PID) regulátor patrí k najčastejšie využívaným typom regulátora v priemysle kvôli jednoduchosti, robustnosti a hlavne možnosti realizácie v rôznych analógových prevedeniach (elektrický, pneumatický) a v súčasnosti preferovaných digitálnych prevedeniach. Vzhľadom k tomu, že môj systém má diskrétnu charakteristiku tak aj PID regulátor musí byť diskrétny. Implementácia diskrétneho PID regulátora pomocou S-funkcie vyzerá nasledujúco

```
function angle = bnb_pid(position, w, dt)
persistent um1 em1 em2
if nargin==0
    um1 = 0;
    em1 = 0;
    em2 = 0;
    return
end
Kp = 1;
Ti = 3;
Td = 0.3;
e = w-position;
angle = um1+Kp*((1.0+dt/(Ti)+Td/dt)*e+(-1.0-2.0*Td/dt)*em1+Td/dt*em2);
um1 = angle;
em2 = em1;
em1 = e;
end
```

Na začiatku som si zadefinoval perzistentné premenné $um1$, $em1$, $em2$. Parametre PID regulátora som navrhol experimentálne s cieľom uregulovať systém v rozumnom čase. Najvhodnejšie parametre boli $Kp = 1, Ti = 3$ a $Td = 0.3$. V každej perióde vzorkovania dt sa najprv vypočíta regulačná odchýlka e, a potom akčný zásah ako súčet predchádzajúceho akčného zásahu a aktuálneho akčného zásahu. K výpočtu dochádza rekurentne na základe regulačnej odchýlky e a regulačných odchýlok v k-1 perióde vzorkovanie $em1$, a v k-2 perióde vzorkovania $em2$.

2.5.2 LQR regulátor s integračnou činnosťou bez ohraničení

Návrh LQR regulátora s integračnou činnosťou je klasický LQR návrh pre prípad rozšíreného stavového vektora a taký počet nových stavových veličín, pre ktoré je potrebné mať integračnú činnosť. Vektor derivácií nových stavových veličín je súčinom matice konštánt a vektora pôvodných stavových veličín. S-funkcia, ktorá zabezpečí implementáciu LQR regulátora s integračnou činnosťou je $bnb\_lqr\_int(position, r)$. Vstupom do funkcie je okrem pozície aj žiadaná referencia r. Štruktúra S-funkcie vyzerá nasledovne

```
function angle = bnb_lqr_int(position, r)
persistent previous_position previous_angle z v
if nargin==0
    previous_position = bnb_getBallPosition()/100;
    previous_angle = 0;
```

```
    z = [0; 0; 0; 0];
    v = 0;
    return
end
```

Začiatok S-funkcie je podobný ako v prípade LQR stavového regulátora akurát pribudla premenná v. Po inicializácii všetkých premenných nasleduje definovanie stavových matíc a návrh pozorovača. Rozdiel nastáva pri návrhu zosilnenia pomocou príkazu *dlqr*, kde je potrebné zadefinovanie rozšírených stavových matíc (2.13).

$$A_e = \begin{bmatrix} A_d & 0 \\ -C_d & I \end{bmatrix} \qquad B_e = \begin{bmatrix} B_d \\ 0 \end{bmatrix} \tag{2.13}$$

V S-funkcii to zabazpečí kód

```
Ae = blkdiag(Ad, 1);
Ae(3, 1:2) = -Cd;
Be = [Bd; 0];
```

Nasleduje vytvorenie nových váhových matíc

```
Qe = diag([100, 10, 0.1])/0.08;
Re = 0.08/5;
```

Ďalší krok je vyrátanie matice zosilnení KPI pomocou príkazu *dlqr*

```
KPI = dlqr(Ae, Be, Qe, Re);
KP = KPI(:, 1:2);
KI = -KPI(:, 3);
```

Prvé dva elementy som vložil do KP a posledný do KI a zákon riadenia s integrátorom potom vyzerá

```
angle = KI*v - KP*x;
```

Integračnú činnosť zabezpečí premenná v, ktorá sa zistí sčítaním poslednej hodnoty v so žiadanou veličinou r a odčítaním pozície gule.

```
 v = v + r - position;
```

A na záver sa uložia nové hodnoty perzistentných premenných.

```
previous_position = position;
previous_angle = angle;
end
```

2.5.3 MPC regulátor s integračnou činnosťou s ohraničeniami

MPC regulácia s integračnou činnosťou sa zabezpečí použitím Δu formulácie prediktívneho riadenia (rovnice (1.100)). Implementácie je realizovaná pomocou S-funkcie $mpc\_du(position, r)$, ktorá ma dva vstupy pozíciu a referenciu. Štruktúra funkcie je podobná ako u MPC stavového regulátora. Zmenili sa váhové matice a pribudla nová optimalizovaná premenná prírastok riadenia dus.

```
Q = 100/0.08;
R = 0.08/5;
xs = cell(N+1, 1);
us = cell(N, 1);
ys = cell(N, 1);
for k = 1:1:N+1
    xs{k} = sdpvar(nx, 1);
end
for k = 1:1:N
    us{k} = sdpvar(nu, 1);
    dus{k} = sdpvar(nu, 1);
    ys{k} = sdpvar(ny, 1);
end
um = sdpvar(nu, 1);
```

Zároveň sa zmenili ohraničenia a účelová funkcia.

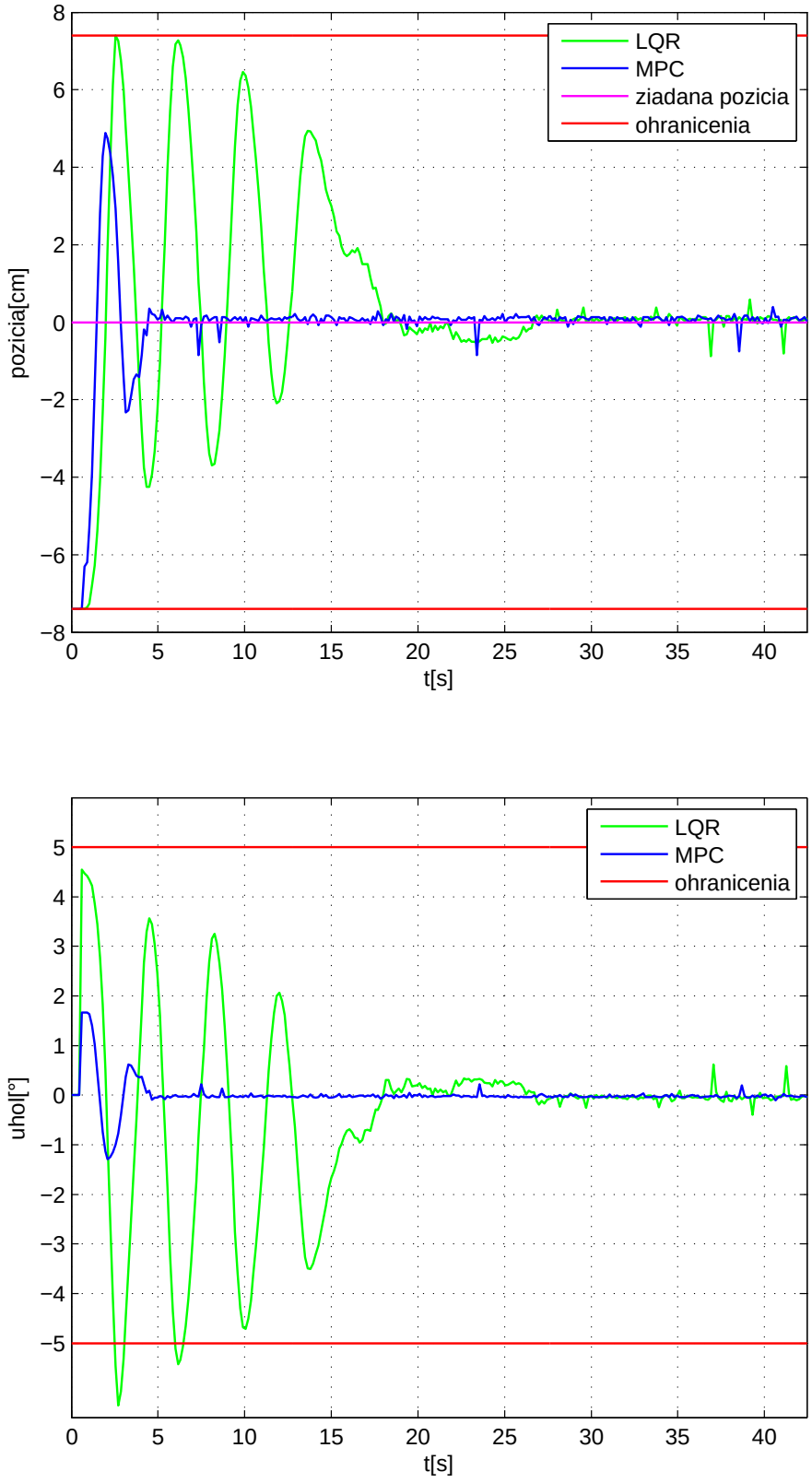
```
for k = 1:1:N
    obj = obj + ((ys{k} - r)'*Q*(ys{k} - r) + (dus{k})'*R*(dus{k}));
    cst = [cst; xs{k+1} == Ad*xs{k} + Bd*us{k}; ys{k} == Cd*xs{k} + Dd*us{k}];
    cst = [cst; -5 <= us{k} <= 5];
    cst = [cst; -0.074 <= xs{k}(1) <= 0.074];
       if k == 1
        cst = [cst; dus{k} == us{k} - um];
    else
        cst = [cst; dus{k} == us{k} - us{k-1}];
    end
end
```

V cykle sa nachádza podmienka, ktorá zabezpečí, že v prvom kroku je prírastok riadenia rovný rozdielu aktuálneho a počiatočného riadenia um, a v každom ďalšom kroku je to rozdiel aktuálneho a predchádzajúceho riadenia. Na výpočet akčného zásahu sa tiež používa optimizer, kde pribudol vstup, počiatočný akčný zásah um.

```
options = sdpsettings('solver', 'gurobi','verbose', 0, 'cachesolvers', 1);
controller = optimizer(cst, obj, options,[xs{1}; um], us{1});
[angle, b, c] = controller{[x; previous_angle]};
if b~=0
   error('neriesitelny problem')
end
```

2.6 Experimentálne výsledky a porovnania

Jednotlivé návrhy riadenia som experimentálne overil. Ako prvé som testoval návrh stavovej regulácie pomocou LQR bez ohraničení a MPC s ohraničeniami na akčný zásah a pozíciu. Testovanie som realizoval tak, že som guľu postavil na jeden koniec dopravného pásu s cieľom dostať sa na nulovú pozíciu. Výsledky sú zobrazené na obrázku (2.10). Prvý graf na obrázku zobrazuje trajektóriu riadenej veličiny, pozície a druhý trajektóriu riadiacej veličiny, čiže veľkosť uhla poslaného do procesu. Na základe priebehov riadenie môžem povedať, že obidva regulátory zabezpečia stavy do nuly v relatívne krátkom čase. Z pohľadu riadenia reálneho zariadenia však LQR regulátor bez ohraničení výrazne zaostáva za MPC regulátorom s ohraničeniami. Dôvodom je porušenie ohraničení, ktoré každé reálne zariadenie má. V tomto prípade je to dĺžka pásu a veľkosť uhla, ktorú môžu piesty vysúvaním vytvoriť. V prípade, že by zariadenie neobsahovalo mechanické zábrany na konci pásu, guľa by sa skotúľala z pásu. A zároveň nie je možné vysunúť piesty viac, ako sú konštrukčne navrhnuté. Príliš veľký prvý akčný zásah LQR regulátora vedie k rozkmitaniu celého zariadenia a je potrebné dodať omnoho viac energie na dosiahnutie nulových stavov.

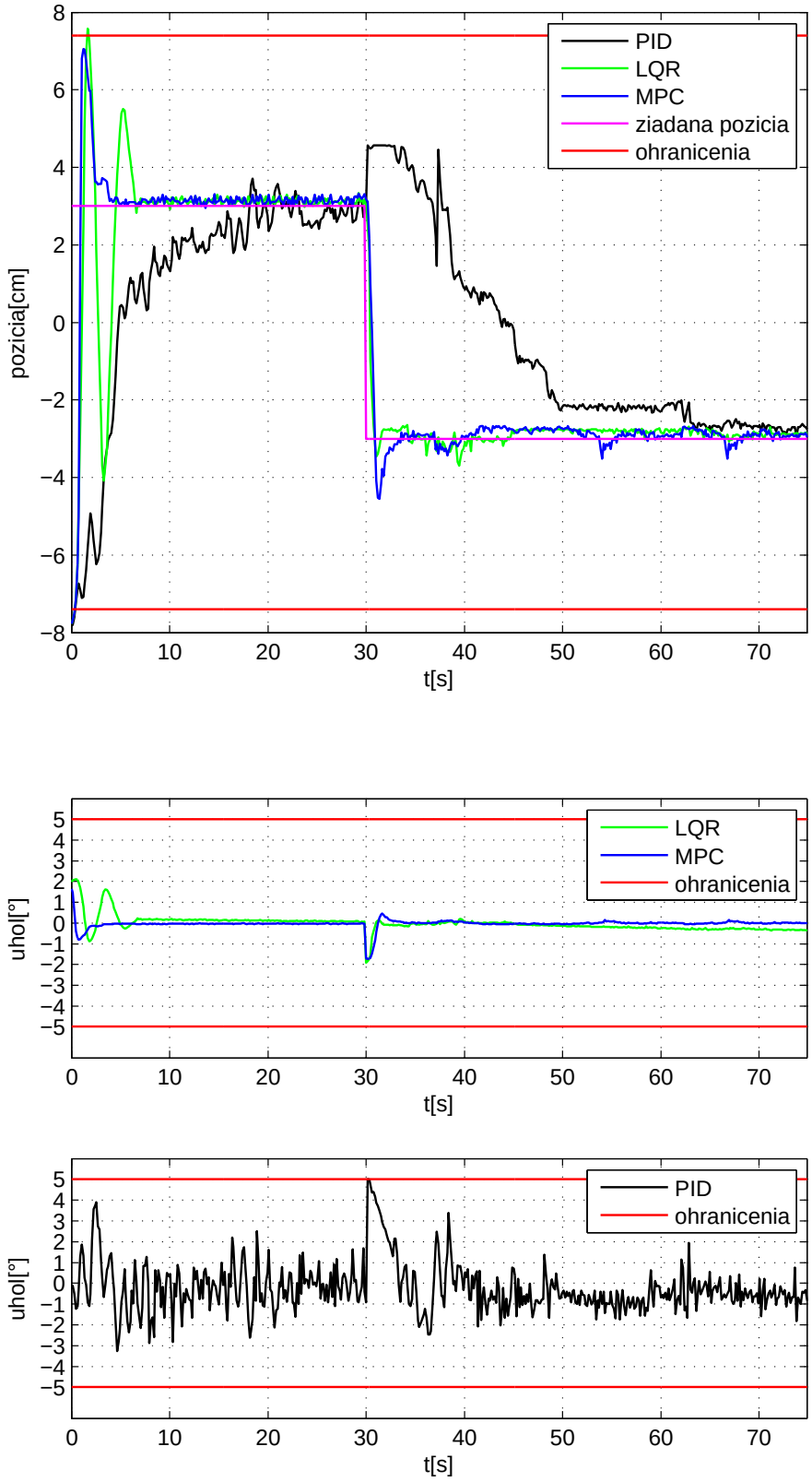


Obr. 2.10: Stavová regulácia

Hlavnou náplňou práce bolo navrhnúť také riadenie, ktoré zabezpečí presun gule na ľubovoľné miesto na dopravnom páse. Na sledovanie žiadanej pozície som použil diskrétny PID regulátor, LQR regulátor s integračnou činnosťou a MPC regulátor s integračnou činnosťou s ohraničeniami. Činnosť všetkých navrhnutých regulátorov som experimentálne overil. Cieľom testov bolo dostať guľu z pozície -8 centimetrov najprv na žiadanú hodnotu 3 centimetre a po ustálení ešte na pozíciu -3 centimetre. Priebehy realizovaných experimentov sú zobrazené na obrázku (2.11). Podobne ako pri stavovej regulácii prvý graf zobrazuje trajektóriu riadenej veličiny a druhý trajektóriu riadiacej veličiny. Pre lepšie znázornenie priebehov riadiacej veličiny som vykreslil samostatne PID regulátor, ktorý ma kmitavý priebeh, čím by zneprehľadnil všetky ostatné priebehy. Cieľ riadenia splnili všetky tri navrhnuté regulátory. Výrazne zaostáva diskrétny PID regulátor, u ktorého sa dosiahne žiadaná referencia za najdlhší čas. Vyznačuje sa výrazne kmitavým priebehom riadenej veličiny. Pri PID je zároveň potrebné dodať najväčšie množstvo energie do systému na dosiahnutie stanovených cieľov. Ostatné dve regulátory majú podobné priebehy riadenej a riadiacej veličiny. Avšak pri LQR s integračnou činnosťou dochádza k nárazu do mechanickej zábrany na konci pásu, lebo nezohľadňuje ohraničenia na pozíciu. Tento fakt značne znevýhodňuje LQR voči MPC regulátoru s ohraničeniami. MPC vykoná v každom kroku len taký akčný zásah, aby sa guľa príliš nerozbehla a neprekročila ohraničenia. V prípade sledovania nedochádza k porušení ohraničení na vstup u žiadneho z navrhnutých regulátorov. Pri PID regulátore je to zabezpečené tým, že som volil menej agresívnejší regulátor, ktorý sa dostane na žiadanú pozíciu za dlhší čas ale bez porušovania ohraničení. U LQR je neporušenie ohraničení na vstup spôsobené voľbou váhových matíc.

Posledným cieľom práce bolo kvalitatívne overiť činnosť navrhnutých regulátorov. Kvalitu riadenia všetkých regulátorov som porovnal na základe troch kritérií. Prvé je štandardné IAE kritérium (*integral absolute value of error*) (2.14), kde k_f reprezentuje poslednú periódu vzorkovania v simuláciách.

$$\text{IAE} = \int_0^T |(r(t) - y(t))|\, dt \Rightarrow \text{IAE} = \sum_{k=1}^{k_f} |(r_k - y_k)| \tag{2.14}$$



Obr. 2.11: Sledovanie referencie

Druhé moje zvolené kritérium je IACE (*integral absolute control error*), ktoré je definované rovnicou (2.15).

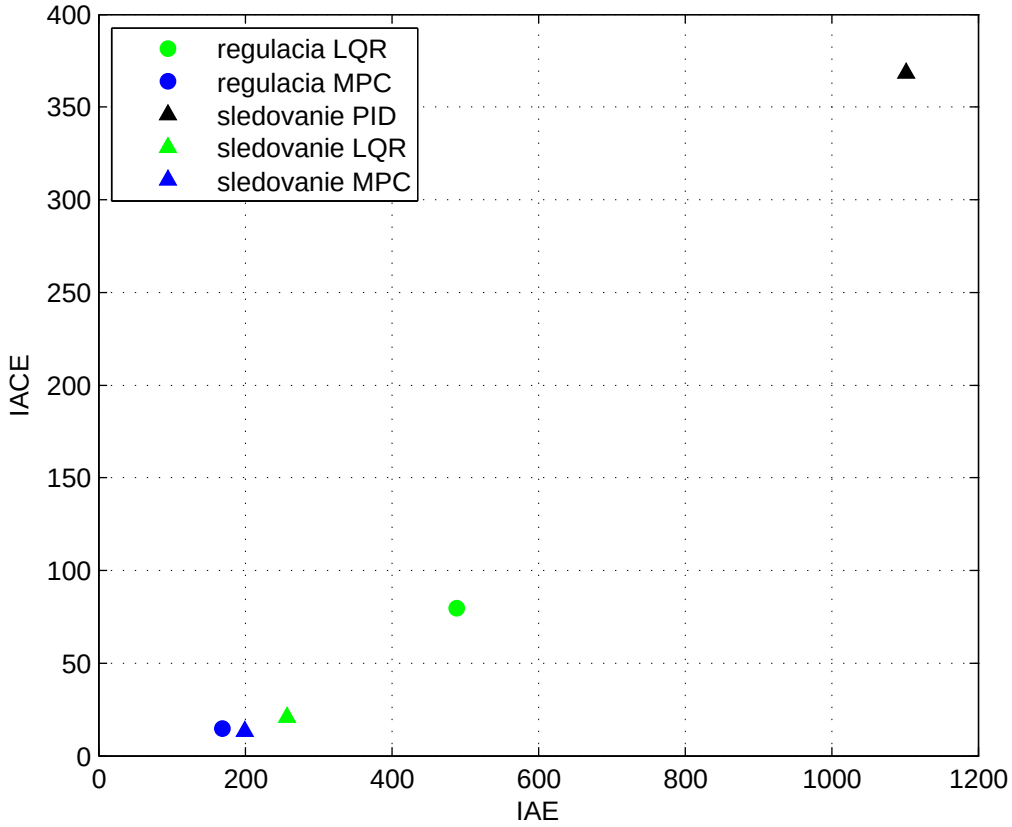
$$\text{IACE} = \int_0^T |\Delta u(t)|\, dt \Rightarrow \text{IACE} = \sum_{k=1}^{k_f} |\Delta u_k| \tag{2.15}$$

Posledným kritériom je porušenie ohraničení. Voľbu IAE spolu s IACE som si zvolil z dôvodu porovnať nielen sumu odchýliek, ale aj množstvo energie potrebné na jej dosiahnutie. Vyhodnotenie kritérií sa nachádza v tabuľke (2.1),

Tabuľka 2.1: Kritéria kvality

		IAE	IACE	Porušenie ohraničení
regulácia	LQR	488.3	79.6	ÁNO
	MPC	168.8	14.7	NIE
sledovanie	PID	1101.8	368.2	ÁNO
	LQR	257.5	20.9	ÁNO
	MPC	199.2	13.5	NIE

kde môžeme vidieť, že MPC regulátor má pri regulácii aj sledovaní najnižšie hodnoty IAE a IACE a samozrejme bez porušenia ohraničení. Závislosť IAE a IACE som aj graficky vyjadril pomocou grafu 2.12, kde je jasne vidieť rozdiel medzi navrhnutými regulátormi. PID je najhorší, čo sa týka zabezpečenia rýchleho sledovania a aj množstva dodanej energie, a preto sa nachádza najďalej od nuly. MPC regulátor je najlepší, lebo zabezpečí najrýchlejšie sledovanie pri najmenšej dodanej energii, čo zobrazuje na grafe pozíciu najbližsie k nule. LQR až tak výrazne nezaostáva za MPC, ale treba brať do úvahy najdôležitejšie kritérium pri riadení reálneho zariadenia a to je porušenie ohraničení.



Obr. 2.12: Závislosť IACE od IAE

Kapitola 3

Záver

V práci som sa zaoberal viacerými spôsobmi riadenia polohy gule umiestnenej na dopravnom flexibilnom páse. Prvým dôležitým krokom bola zoznámenie sa čo najlepšie s procesom a vytvorenie modelu, ktorý bude čo najlepšie vystihovať správanie gule na naklonenej ploche. Pri návrhu modelu som použil Langrangeovu rovnicu popisujúce pohyb gule po naklonenej rovine. Túto rovnice som linearizoval a vytvoril z nej prenosovú funkciu a stavovú reprezentáciu systému, pričom stavy systému sú pozícia a rýchlosť.

Práca so zariadením začala kalibráciou snímačov vzdialenosti tak, aby výstupom zo zariadenia bola pozícia v centimetroch zodpovedajúca stupnici na páse. Pri prepočte na centimetre som použil aproximáciu racionálnou lomenou funkciou a výstupy zo snímačov som následne váhoval podľa vzdialenosti od jednotlivých snímačov.

Pokročilejšie stratégie riadenie (LQR, MPC) sú založené na stavovej spätnej väzbe a je potrebné poznať všetky stavy. Dopravný pás disponuje možnosťou merať len pozíciu a rýchlosť je potrebné odhadovať pomocou pozorovača stavu. Použil som Kalmanov filter a kvalitu odhadu stavu som si overil tak, že som porovnal skutočnú pozíciu a vypočítanú rýchlosť s odhadovanou pozíciou a rýchlosťou. Po navrhnutí dostatočne presného pozorovača som pokračoval v návrhu riadenia.

Ako prvé som tento proces riadil pomocou diskrétneho PID regulátora. Parametre regulátora som zvolil rovné jednej a tie som menil dovtedy, kým som nenašiel vhodný regulátor. Tento regulátor nie je až taký rýchly, ale zabezpečí dosiahnutie žiadanej hodnoty bez porušení ohraničení. Potom som pokračoval v návrhu LQR riadenia. Pri prvých pokusoch som zistil, že by bolo vhodné najskôr upraviť model, ktorý by zohľadnoval značné trenie. Realizoval som to tak, že som vytvoril viacero podobných modelov, a jednotlivé modely otestoval. Najvhodnejší model som používal pri návrhu LQR a MPC regulátorov. Ako prvé som sa snažil zabezpečiť stavovú reguláciu pomocou

LQR regulátora bez ohraničení a MPC regulátora s ohraničeniami. Cieľ som splnil a potvrdilo sa, že pri voľbe rovnakých váhových matíc LQR poruší ohraničenia a MPC dokázalo pracovať medzi ohraničeniami. Po návrhu stavovej regulácie som pokračoval v návrhu a implementácii sledovania žiadanej referencie pomocou LQR regulátora bez ohraničení a MPC regulátora s ohraničeniami. Obidva regulátory zabezpečili sledovanie žiadanej veličiny pri skokovej zmene smerom nahor aj nadol. Trajektórie riadenej a riadiacej veličiny mali dosť podobný priebeh. Ohraničenia na vstupy neboli prekročené ani v prípade LQR v dôsledku vhodne zvolených váhových matíc. Ohraničenia na pozíciu však už porušené u LQR boli a guľa narazila do zábrany. U MPC regulátora prekročené neboli. Na základe všetkých experimentov môžem povedať, že prediktívne riadenie splnilo očakávania a je najvhodnejším z pomedzi testovaných stratégii riadenia, ale všetko má aj svoje nevýhody. Nevýhodou prediktívneho riadenia, ktorú som aj sám postrehol pri simuláciách, je vysoká výpočtová náročnosť a pri niektorých z mnohých simulácii nebol schopný môj osobný počítač vypočítať akčný zásah v čase jednej periódy vzorkovania.

Do budúcnosti by som navrhoval použiť pri práci lepšiu guľu, ktorá by nebola spojená z dvoch častí a trocha sploštená, čo spôsobovalo občasné zaseknutia, a aj v pri väčších uhloch naklonenia guľa zostávala stáť na mieste.

Prvotným cieľom práce bolo zabezpečiť ovládanie všetkých šiestich piestov pod pásom a vytvárať vlnu, ktorá bude transportovať objekt. Z dôvodu častých porúch, opotrebenia natiahnutého pásu, použitia neideálnej gule a celkovej zložitosti problému som sa sústredil na ovládanie iba dvoch krajných piestov. Okrem návrhu a implementácie jednotlivých stratégii riadenia som vytvoril užívateľsky jednoduché grafické prostredia pre kalibráciu snímačov, ktoré môže byť použité pri ďalšej práci so zariadením.

Literatúra

[1] Ball & beam: System modeling @ONLINE. http://ctms.engin.umich.edu/CTMS/index.php?example=BallBeam&section=SystemModeling.

[2] Rwth - mindstorms nxt toolbox @ONLINE. http://www.mindstorms.rwth-aachen.de/trac/wiki/Documentation.

[3] Maciejowski J.M. *Predictive Control with Constraints*. Prentice-Hall, 2002. Japanese translation published by Tokyo Denki University Press, 2005. "Best Control Engineering Textbook"prize, SICE, 2007.

[4] J. Mikleš and M. Fikar. *Modelovanie, identifikácia a riadenie procesov II*. STU Press, Bratislava, 2004. 266 pp.

[5] Bátora Vladimír. Lego mindstorms @ONLINE. http://www.posterus.sk/?p=8553, 2010.