SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

PREDIKTÍVNE RIADENIE AUTOMOBILOVÝCH SKUPÍN

DIPLOMOVÁ PRÁCA

FCHPT-5414-58722

Študijný program: automatizácia a informatizácia v chémii a potravinárstve

Číslo študijného odboru: 2621

Názov študijného odboru: 5.2.14 automatizácia

Školiace pracovisko: oddelenie informatizácie a riadenia procesov

Vedúci záverečnej práce/školiteľ: doc. Ing. Michal Kvasnica, PhD.

Bratislava 2013 **Bc. Dalibor Páleš**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

PREDIKTÍVNE RIADENIE AUTOMOBILOVÝCH SKUPÍN

DIPLOMOVÁ PRÁCA

FCHPT-5414-58722

Študijný program: automatizácia a informatizácia v chémii a potravinárstve

Číslo študijného odboru: 2621

Názov študijného odboru: 5.2.14 automatizácia

Školiace pracovisko: oddelenie informatizácie a riadenia procesov

Vedúci záverečnej práce/školiteľ: doc. Ing. Michal Kvasnica, PhD.

Bratislava 2013 **Bc. Dalibor Páleš**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2012/2013
Evidenčné číslo: FCHPT-5414-58722

STU
FCHPT

ZADANIE DIPLOMOVEJ PRÁCE

Študent:	**Bc. Dalibor Páleš**
ID študenta:	58722
Študijný program:	automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor:	5.2.14 automatizácia
Vedúci práce:	doc. Ing. Michal Kvasnica, PhD.

Názov práce: **Prediktívne riadenie automobilových skupín**

Špecifikácia zadania:

Cieľom diplomovej práce je navrhnúť a simulačne overiť prediktívne riadenie aplikované na problém riadenie automobilov, ktoré sa pohybujú v kolóne. Pri riadení jednotlivých automobilov je potrebné dodržiavať ohraničenia na rýchlosť a minimálny rozostup, ako i maximalizovať komfort pasažierov vyjadrený ako minimalizácia požadovanej akcelerácie.

Čiastkové ciele:
* navrhnúť a verifikovať matematický model jednotlivých áut ako i celej skupiny
* implementovať simulačné prostredie na overenie výsledkov
* formulovať optimalizačnú úlohu riadenia
* porovnať centralizovaný a decentralizovaný prístup k riadeniu

Rozsah práce: 40

Riešenie zadania práce od: 18. 02. 2013
Dátum odovzdania práce: 25. 05. 2013

L. S.

Bc. Dalibor Páleš
študent

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

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

Abstrakt

V diplomovej práci sme sa zaoberali problematikou prediktívneho riadenia a jeho použitím pri riadení automobilových skupín. V úvodnej kapitole možno nájsť stručný prehľad základných teoretických princípov prediktívneho riadenia, ktoré boli neskôr použité v experimentálnej časti. Nasleduje zostavenie matematického modelu procesu, ktorý bol neskôr potrebný pre zostavenie celej optimalizačnej úlohy. Praktická časť pozostáva z dvoch hlavných celkov. V prvom sa práca zaoberá implementáciou prediktívneho riadenia v Matlabe, v druhom už nájdeme konkrétne simulácie riadenia automobilových skupín. Simulácie boli vykonané za rôznych podmienok tak, aby čo najvýstižnejšie opisovali situácie, ktoré sa stávajú v bežnej premávke. Čiastkovým cieľom bolo aj nájdenie vhodných spôsobov riadenia pri snahe znížiť spotrebu paliva. Výsledky simulácií sú doložené grafickými priebehmi riadených a akčných veličín. Pre názornejšiu prezentáciu dosiahnutých výsledkov bola pre každú simuláciu vytvorená animácia v Matlabe, ktorú možno nájsť na priloženom CD. Záver praktickej časti je venovaný opisu fungovania tejto animácie.

Abstract

This diploma thesis deals with a model predictive control (MPC) approach to control of a group of cars. The first part of the thesis introduces basic principle of optimisation and of model predictive control. Subsequently, we formulate the mathematical model of the group of cars, which can be embedded into the MPC framework. A Matlab-based implementation of proposed MPC algorithms is then discussed. The second part of the thesis then discusses obtained simulation data. A wide scale of simulation scenarios was considered which correspond to situations that frequently occur in practice. One of the scenario considers finding suitable ways of control to decrease fuel consumption. Results of simulations are shown on figures containing both controlled variables and control actions. A simple animation was made for each simulation, so that results can be better illustrated and understood. The end of the practical part is focused on description of creating the animation in Matlab.

Zoznam obrázkov

Obr. 1.1 Znázornenie fungovania MPC. 8
Obr. 1.2 Graf konvexnej funkcie. 10
Obr. 1.3 Priamka prechádzajúca bodmi $x1$, $x2$. Množina C je afinná, ak jej patria body $x1$, $x2$ a priamka prechádzajúca cez $x1$, $x2$ leží v C. 11
Obr. 1.4 Dva polpriestory vytvorené v $R2$ nadrovinou $aTx = b$ 11
Obr. 1.5 Mnohosten vzniknutý prienikom polpriestorov s normálovými vektormi $a1, \dots, a5$. ... 12
Obr. 2.1 Geometrická ilustrácia QP. 17
Obr. 4.1 Grafické priebehy prvého simulačného scenára. 28
Obr. 4.2 Grafické priebehy prvého simulačného scenára so zastavením auta 2. 30
Obr. 4.3 Grafické priebehy pre druhý simulačný scenár, $Qd = Qa = 1$. 31
Obr. 4.4 Grafické priebehy pre druhý simulačný scenár, $Qd = 2$, $Qa = 1$. 32
Obr. 4.5 Grafické priebehy pre druhý simulačný scenár, $Qd = 1$, $Qa = 2$. 33
Obr. 4.6 Grafické priebehy pre tretí simulačný scenár. 35
Obr. 4.7 Relatívny koeficient aerodynamického odporu v závislosti od rozostupu pre 2 autá. ... 37
Obr. 4.8 Výsledná závislosť relatívneho koeficientu aerodynamického odporu od medzier medzi vozidlami pre 3 autá. 38
Obr. 4.9 Grafické priebehy štvrtého simulačného scenára. 39
Obr. 4.10 Grafické priebehy modifikovaného štvrtého simulačného scenára. 41
Obr. 4.11 Grafické výsledky piatej simulácie s rozostupmi 0,1336 dĺžky auta. 43
Obr. 4.12 Grafické výsledky piatej simulácie s rozostupmi 0,8 dĺžky auta. 45
Obr. 4.13 Grafické výsledky piatej simulácie s rozostupmi 0,8 dĺžky auta a zavedením E. 46
Obr. 4.14 Vzhľad grafického okna pri priebehu animácie. 47

Obsah

Úvod **6**
1. Úvod do MPC **7**
1.1. Základné prvky MPC 7
1.2. Matematická formulácia MPC 8
1.3. MPC a konvexná optimalizácia 9
2. Riadenie automobilových skupín **13**
2.1. Zostavenie matematického modelu 13
2.2. Definovanie účelovej funkcie s obmedzeniami 15
2.3. Zostavenie úlohy kvadratického programovania 16
3. Riešenie optimalizačného problému v Matlabe **21**
3.1. Definovanie úlohy 21
3.2. Simulácia riadenia 23
4. Simulačné výsledky **27**
4.1. Scenár 1 27
4.2. Scenár 2 30
4.3. Scenár 3 34
4.4. Určenie optimálneho rozostupu medzi dvoma autami 36
4.5. Určenie optimálneho rozostupu medzi troma autami 37
4.6. Scenár 4 38
4.7. Scenár 5 41
4.8. Vytvorenie animácie 46
Záver **51**
Zoznam použitej literatúry **52**

Úvod

Predstavme si situáciu vo väčšom meste počas rannej alebo popoludňajšej dopravnej špičky. Ľudia vo svojich autách stoja v kolónach, prejdú prinajlepšom niekoľko desiatok metrov a opäť stoja na mieste. Všetci sú nervózni a určite nikoho nebaví štýl jazdy, pri ktorom je potrebné nespočetne mnohokrát sa pohnúť a vzápätí zastaviť. Takáto situácia je najmä vo veľkomestách úplne bežná, a preto sa ľudia snažia s týmto problémom niečo urobiť. Snaha smeruje k tomu, aby bolo možné do automobilov inštalovať zariadenie, ktoré by po aktivácii prevzalo kontrolu nad vozidlom. Samozrejme, takéto riadenie nie je jednoduché navrhnúť a pri jeho návrhu je nutné myslieť na množstvo obmedzení a situácií, ktoré môžu nastať. Pri tom je nutné brať ohľad v prvom rade na bezpečnosť pasažierov, komfort jazdy, úsporu paliva v protiklade so snahou zlepšiť plynulosť premávky, a teda zrýchliť reakcie vozidla na okolitý vývoj.

Implementácia takéhoto riadenia je smer, ktorý sa určite v blízkej budúcnosti bude rozvíjať a predpokladám, že časom sa stane úplne bežné, že človek riadiaci vozidlo v kolóne alebo aj mimo nej stlačí tlačidlo a o ostatné sa už postará riadiaci mechanizmus. Tým by sa mohla dosiahnuť plynulejšia doprava v mestách a samozrejme väčší komfort zákazníkov, ktorý by takúto možnosť využili.

To sú základné myšlienky, ktoré ma presvedčili k tomu, aby som sa na tento problém pozrel detailnejšie a pokúsil sa simulovať situácie, ktoré by v reálnom živote mohli nastať. Hlavným cieľom bolo sformulovať úlohu MPC (z angl. Model predictive control) ako optimalizačný problém, riešiť ho a simulovať riadenie pomocou MPC regulátora. Prvou úlohou bolo stručné zhrnutie teoretických poznatkov súvisiacich s prediktívnym riadením v rozsahu, v akom sú potrebné pre realizáciu experimentálnej časti. Na základe týchto poznatkov bolo ďalším cieľom vytvorenie matematického modelu a formulácia celého MPC problému s obmedzeniami, ktoré sú blízke skutočnosti. Hlavným cieľom experimentálnej časti bolo simulovanie autonómneho riadenia áut idúcich za sebou v kolóne. Dôležitú úlohu mala zohrávať bezpečnosť riadenia, jeho kvalita a v neposlednom rade úsilie potrebné na jej dosiahnutie. Čiastkovým cieľom bolo aj skúmanie možností úspory paliva pri riadení áut idúcich tesne za sebou.

1. Úvod do MPC

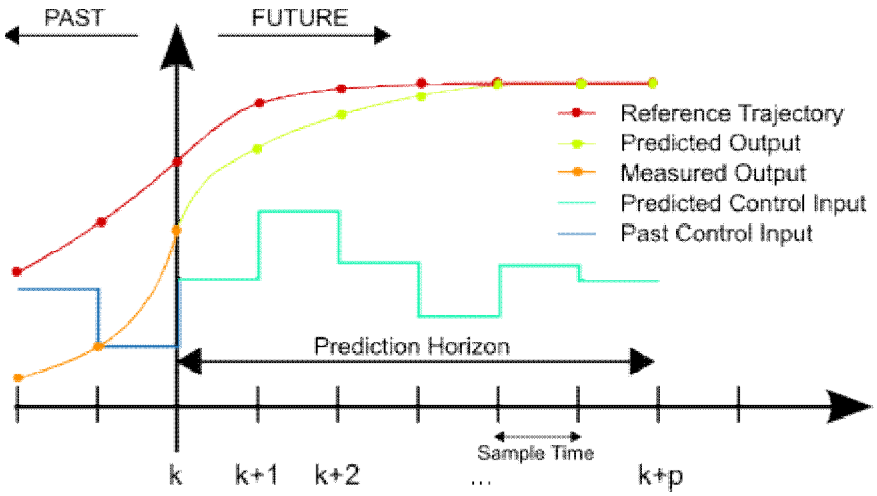
V ostatných rokoch prediktívne riadenie zaznamenalo pomerne značný posun od teoretického výskumu k praktickým aplikáciám. Ako moderný spôsob riadenia mnohorozmerných systémov s ohraničeniami nachádza široké uplatnenie najmä v ropných rafinériách, petrochemickom a chemickom priemysle. Spočiatku bolo použitie MPC obmedzené na pomalé procesy. S rozvojom výpočtovej techniky a výkonných numerických metód je MPC možné použiť aj pri rýchlejších procesoch. Tým pádom sa výpočtová náročnosť ako hlavná nevýhoda MPC pomaly eliminuje, a tým sa rozširuje spektrum jeho praktických aplikácií.

Značnou výhodou pri aplikáciách prediktívneho riadenia je jeho schopnosť efektívne zaobchádzať s obmedzeniami vstupných, stavových a výstupných veličín. Tieto obmedzenia sa v praktickom živote vyskytujú takmer vždy. Sú to hlavne obmedzenia akčných členov, bezpečnostné obmedzenia určujúce dovolené hodnoty stavových a výstupných veličín a technologické obmedzenia súvisiace s dosiahnutím kvality jednotlivých produktov. [3]

1.1 Základné prvky MPC

MPC je metóda riadenia, pri ktorej sa model riadeného systému používa na výpočet riadiaceho vstupu na základe predikcií účinkov riadenia na riadený výstup. Nasledovné prvky a vlastnosti sú pre MPC charakteristické:

- Na základe *matematického modelu* riadeného systému, dostupných informácii o systéme a zvolenej trajektórii riadenia sa predpovedá budúci riadený výstup systému.
- Určité *kritérium riadenia* je minimalizované na zvolenom časovom horizonte predikcie. Výsledkom je postupnosť riadiacich vstupov pre daný horizont predikcie.
- MPC pracuje v zmysle spätnoväzbového regulátora, čo umožňuje tzv. *stratégia pohyblivého horizontu*. V každom kroku je použitý len prvý člen z vypočítanej postupnosti riadenia a na základe nových informácií sa v ďalšom kroku určí nová optimálna postupnosť riadenia.
- Zaobchádzanie s *obmedzeniami* vstupných, stavových a výstupných veličín je pre MPC prirodzené. [3]



Obr. 1.1 Znázornenie fungovania MPC. [7]

Na obr. 1.1 môžeme vidieť, že MPC na základe informácií, ktoré má k dispozícii v čase k a na základe známej budúcej trajektórie žiadanej veličiny vypočítava sekvenciu riadiacich vstupov až do času $k+p$, kde p predstavuje predikčný horizont. V čase k sa prvý vypočítaný vstup aplikuje na riadený proces a znova prebehne optimalizácia, ktorej výsledkom bude nová sekvencia riadiacich vstupov a takisto aj nový odhad budúcich stavov. Takýto postup sa nazýva stratégia pohyblivého horizontu a jej implementáciou zavádzame do riadenia spätnú väzbu.

1.2 Matematická formulácia MPC

Základná formulácia pozostáva z účelovej funkcie a obmedzení, ktoré tvorí predikčná rovnica a obmedzenia vstupných, stavových, prípadne výstupných veličín. Všeobecný matematický zápis vyzerá nasledovne:

$$\min \sum_{k=0}^{N-1} (||Q_x x_{t+k}||_p + ||Q_u u_{t+k}||_p) \tag{1.1}$$

$$\text{s.t. } x_{t+1} = Ax_t + Bu_t \tag{1.1a}$$

$$x_t = x(t) \tag{1.1b}$$

$$x_t \in X \tag{1.1c}$$

$$u_t \in U \tag{1.1d}$$

Kritérium (1.1) tu reprezentuje účelovú funkciu, ktorá je súčtom N členov, kde N je predikčný horizont, x_{t+k} a u_{t+k} sú vektory stavov a riadiacich veličín v čase $t+k$, p určuje normu (napr. 1-norma, 2-norma, ∞-norma) a matice Q_x a Q_u sú tzv. váhové matice. X a U predstavujú prípustné množiny pre stavy a vstupy. Rovnica (1.1a) vyjadruje stavový opis daného modelu. [5]

1.3 MPC a konvexná optimalizácia

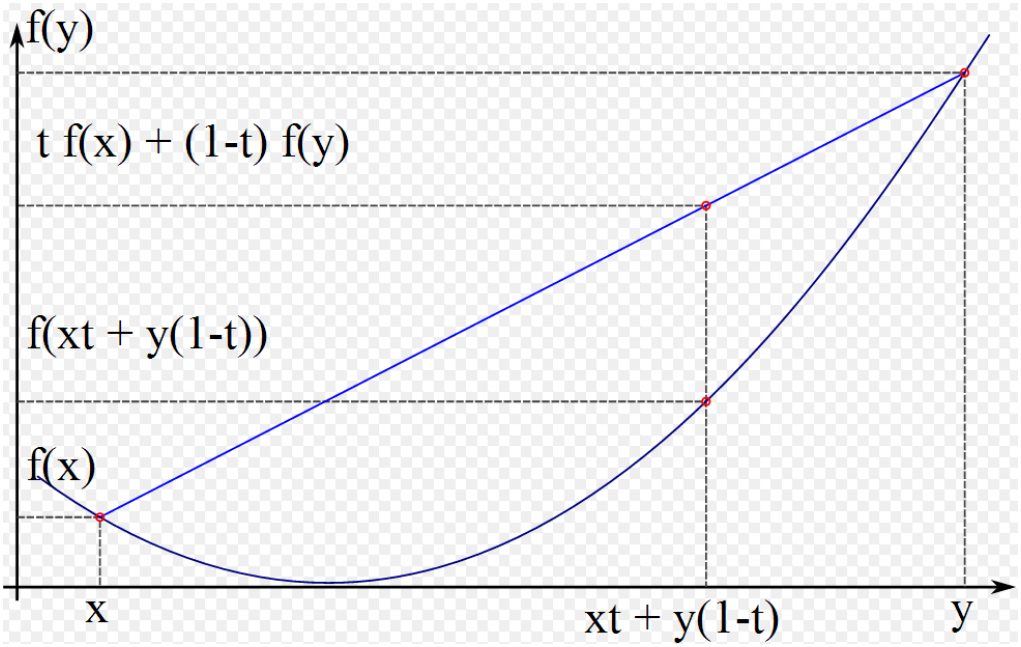
Pri návrhu a implementácii MPC musíme mať na pamäti pomerne veľkú výpočtovú náročnosť v porovnaní s klasickými metódami riadenia. Pre praktické použitie je dôležité, aby naše výpočtové prostriedky boli schopné vypočítať akčný zásah v priebehu maximálne jednej periódy vzorkovania. Tento výpočet znamená, že je potrebné vyriešiť optimalizačný problém vzhľadom na všetky obmedzenia, a to v každej perióde vzorkovania. Preto sa v MPC používajú také účelové funkcie a prípustné množiny, aby bolo možné pomerne rýchle riešenie.

Veľmi rozšírená je v tomto prípade konvexná optimalizácia. Jedná sa o takú triedu optimalizačných problémov, pri ktorých sú účelová funkcia a takisto aj obmedzujúce podmienky konvexné. Na obr. 1.2 vidíme graf konvexnej funkcie. Aby bola funkcia konvexná, musí spĺňať nasledujúcu podmienku:

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y), \tag{1.2}$$

pre všetky $x, y \in \boldsymbol{R}^n$ a $t \in [0, 1]$. [1]

Na obr. 1.2 je graf kvadratickej funkcie a vidíme, že táto spĺňa požiadavku (1.2), teda je konvexná.



Obr. 1.2 Graf konvexnej funkcie.

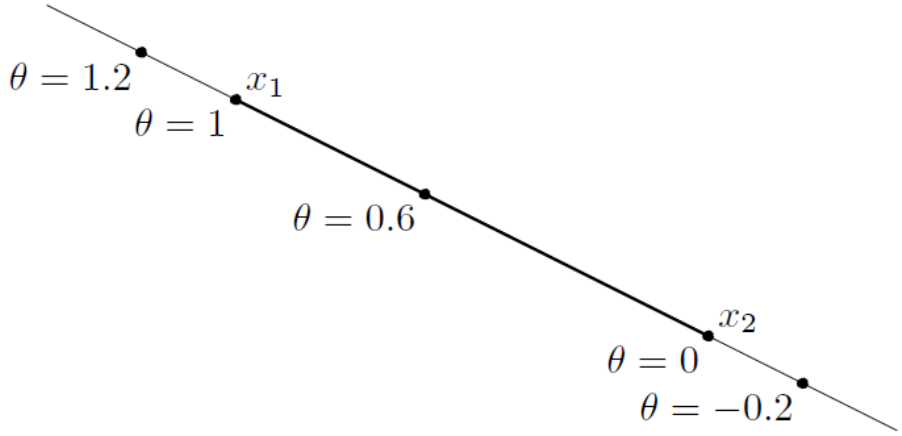
Ako sme si povedali, ak chceme formulovať problém konvexnej optimalizácie, okrem účelovej funkcie musí byť zároveň konvexný aj jej definičný obor, čiže množina prípustných hodnôt. Všeobecne platí, že posúdiť konvexnosť množiny prípustných hodnôt nie je až také jednoduché, avšak existujú určité typy množín, pri ktorých sa to dá pomerne jednoducho.

Platí, že množina *C* je *konvexná* vtedy, ak úsečka medzi ľubovoľnými dvoma bodmi patriacimi do *C*, celá leží v množine *C*, a teda pre ľubovoľné $x_1, x_2 \in C$ a ľubovoľné $\theta \in [0, 1]$ platí:

$$\theta x_1 + (1-\theta)x_2 \in C. \tag{1.3}$$

Tzv. afinné množiny sú špeciálnym typom konvexných množín. Množina *C* je *afinná*, ak **priamka** prechádzajúca cez ľubovoľné dva body patriace do *C*, leží v *C*. Táto myšlienka sa dá zovšeobecniť pre ľubovoľný počet bodov, a teda musí platiť, že ak $x_{1,\dots,}x_k \in C$, aj

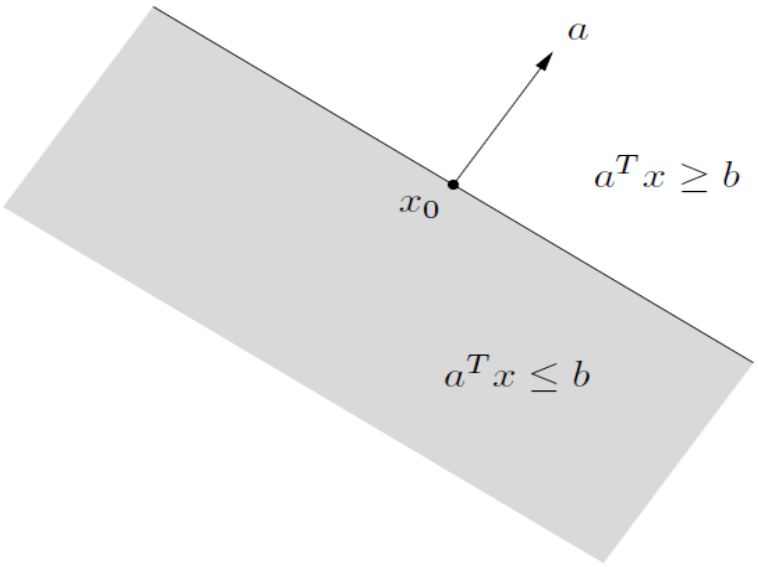
$$\theta_1 x_1 + \cdots + \theta_k x_k \in C, ak\ \theta_1 + \cdots + \theta_k = 1. \tag{1.4}$$



Obr. 1.3 Priamka prechádzajúca bodmi $\boldsymbol{x_1}$, $\boldsymbol{x_2}$. Množina C je afinná, ak jej patria body $\boldsymbol{x_1}$, $\boldsymbol{x_2}$a priamka prechádzajúca cez $\boldsymbol{x_1}$, $\boldsymbol{x_2}$ leží v C. [1]

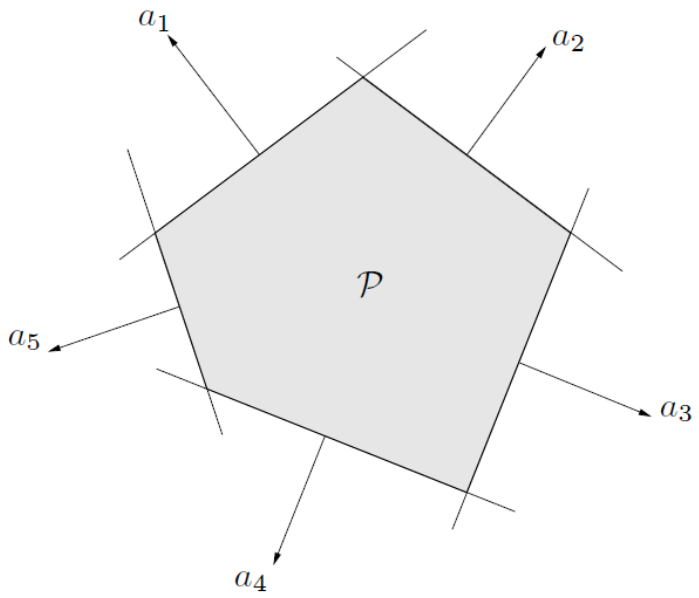
Ďalším významným druhom množín sú *nadroviny*. Je to množina, pre ktorú platí $\{ x \mid a^T x = b\}$, kde $a \in \boldsymbol{R}^n, a \neq 0, b \in \boldsymbol{R}$.

Nadrovina rozdeľuje priestor $\boldsymbol{R}^n$ na dva *polpriestory* $\{ x \mid a^T x \leq b\}$ a $\{ x \mid a^T x \geq b\}$, môžeme to vidieť na obr. 1.4. [1]



Obr. 1.4 Dva polpriestory vytvorené v $\boldsymbol{R^2}$ nadrovinou $\boldsymbol{a^T x = b}$. [1]

Dostávame sa k najbežnejším množinám, ktoré sa vyskytujú v prediktívnom riadení. Prienik konečného počtu polpriestorov budeme nazývať *mnohosten*. Na obr. 1.5 vidíme príklad mnohostena, ktorý vznikne prienikom piatich polpriestorov.



Obr. 1.5 Mnohosten vzniknutý prienikom polpriestorov s normálovými vektormi $\boldsymbol{a_1}, \ldots, \boldsymbol{a_5}$. [1]

2. Riadenie automobilových skupín

V ostatných rokoch existuje snaha o implementáciu prediktívneho riadenia v automobilovom priemysle. Od zavedenia takéhoto riadenia v praxi si vedci a inžinieri sľubujú, že by sa mala zlepšiť plynulosť a bezpečnosť cestnej premávky a to hlavne vo veľkých mestách. Preto vznikla myšlienka, že by bolo vhodné inštalovať do automobilov akéhosi „autopilota“, ktorý by sa po aktivácii postaral o riadenie vozidla, a to najmä počas jazdy v kolóne áut. Tu sa vynára otázka, ako navrhnúť také riadenie, ktoré by dokázalo autonómne riadiť každý automobil tak, aby sa zvýšila efektívnosť cestnej premávky, ale aby sa pri tom zachovala, prípadne zlepšila bezpečnosť a komfort cestujúcich.

Ako jedna z vhodných možností zabezpečenia riadenia každého automobilu sa ukazuje práve MPC. Keďže sa jedná o optimálne riadenie, ktoré ľahko pracuje aj s obmedzeniami, je to pre tento účel ideálny kandidát. Ďalším prínosom implementácie MPC v tejto oblasti by mohla byť úspora prevádzkových nákladov. Keď budeme v každom kroku vypočítavať optimálnu sekvenciu akčných zásahov, dokážeme minimalizovať aj spotrebu paliva.

2.1 Zostavenie matematického modelu

Ako prvé si potrebujeme vytvoriť vhodný model procesu a vykonať jeho diskretizáciu. Pre potreby tejto práce budeme uvažovať, že sa automobil pohybuje len smerom dopredu a dozadu, a teda že regulátor bude dané auto len zrýchľovať, resp. brzdiť a nebude meniť jeho smer.

Ak uvažujeme takýto predpoklad, ako stavové veličiny si zvolíme rýchlosť a vzdialenosť medzi vozidlami. Budeme potrebovať poznať jednak aktuálnu rýchlosť auta, ktoré riadime, ako aj rýchlosť a vzdialenosť automobilu, ktorý sa nachádza v kolóne pred nami riadeným vozidlom. Vstupnou alebo riadiacou veličinou, bude akcelerácia, či už v pozitívnom alebo negatívnom zmysle.

Pre zostavenie dynamického modelu použijeme základné fyzikálne poznatky. Vieme, že časová derivácia dráhy je rovná rýchlosti a že časová derivácia rýchlosti sa rovná zrýchleniu. Ďalej budeme potrebovať nahradiť derivácie diferenciami, aby sme sa nakoniec dopracovali k diskretizovanému modelu. Ak si rýchlosť riadeného auta označíme v, rýchlosť auta idúceho pred nami w a vzdialenosť medzi nimi d, potom pre spojitý model platí:

$$v' = a \tag{2.1a}$$

$$w' = 0 \tag{2.1b}$$

$$d' = w - v\,. \tag{2.1c}$$

Derivácia (2.1b) je nulová, pretože budeme predpokladať, že v čase výpočtu akčného zásahu je rýchlosť auta pred nami konštantná. Spojitý model v maticovom tvare bude teda vyzerať nasledovne:

$$\begin{bmatrix} d' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ v \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} a. \tag{2.2}$$

Diskretizácia modelu (2.2) má nasledovný tvar:

$$\frac{v(t+\Delta t)-v(t)}{\Delta t} \approx v' = a. \tag{2.3}$$

Po úprave rovnice (2.3):

$$v(t + \Delta t) = v(t) + \Delta t a, \tag{2.3a}$$

Keďže pre rýchlosť vozidla idúceho pred nami predpokladáme, že je v čase výpočtu akčného zásahu konštantná, potom platí:

$$w(t + \Delta t) = w(t). \tag{2.4}$$

Deriváciu vzdialeností *d* medzi vozidlami takisto aproximujeme diferenciou:

$$\frac{d(t + \Delta t) - d(t)}{\Delta t} \approx d' = w - v. \tag{2.5}$$

Po úprave rovnice (2.3):

$$d(t + \Delta t) = d(t) + \Delta t(w - v). \tag{2.5a}$$

Po prepísaní rovníc (2.3a), (2.4) a (2.5a) v maticovom tvare dostaneme:

$$\begin{bmatrix} d \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 & -\Delta t & \Delta t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ v \\ w \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t \\ 0 \end{bmatrix} a, \tag{2.6}$$

pričom ľavá strana rovnice (2.6) je v čase $t + \Delta t$ a pravá v čase *t*.
Tým sme dostali stavový opis systému v tvare:

$$x(t + \Delta t) = Ax(t) + Bu(t) \tag{2.7}$$

pri obmedzení vstupnej veličiny $u_{min} \leq u \leq u_{max}$. Vektor *x* v tomto prípade obsahuje 3 stavové veličiny, a to *d (vzdialenosť medzi autami)*, *v (rýchlosť nášho riadeného auta)* a *w (rýchlosť auta idúceho pred nami)*. Matice stavového opisu nášho modelu sú teda nasledovné:

$$\text{A} = \begin{bmatrix} 1 & -\Delta t & \Delta t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{B} = \begin{bmatrix} 0 \\ \Delta t \\ 0 \end{bmatrix} \text{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.8}$$

2.2 Definovanie účelovej funkcie s obmedzeniami

Pre praktické účely tejto práce sme použili ako účelovú funkciu kritérium, ktoré minimalizuje kvadrát rozdielu medzi skutočnou a referenčnou vzdialenosťou medzi automobilmi a kvadrát akčných zásahov. Konkrétny matematický zápis tejto účelovej funkcie je nasledovný:

$$\min \sum_{k=0}^{N-1} \left((d_{t+k} - d_{ref,k})^T Q_x (d_{t+k} - d_{ref,k}) + u_{t+k}^T Q_u u_{t+k} \right), \tag{2.9a}$$

kde d_{t+k} a $d_{ref,k}$ sú skutočná, resp. referenčná vzdialenosť medzi autami, pričom d_{t+k} je prvou zložkou stavového vektora *x* v diskrétnom čase $t+k$. (Pre pripomenutie, vektor *x* obsahuje tri stavy, a to vzdialenosť medzi vozidlami *d*, rýchlosť nášho vozidla *v* a rýchlosť vozidla idúceho pred nami *w*).

Obmedzeniami pre náš prípad bude predikčná rovnica (2.7), ďalej počiatočná podmienka (1.1b), ktorá je vždy stav v danom čase, obmedzenia stavov (1.1c) a obmedzenia vstupov (1.1d) zadefinujeme v tvare:

$$Hx_t \leq K \tag{2.10}$$

$$Lu_t \leq M, \tag{2.11}$$

pričom index t tu treba chápať tak, že tieto obmedzenia musia platiť v každom čase. Stavové obmedzenia a obmedzenia vstupov sme zvolili nasledovne:

$$x'_{min} = [0\ 0\ 0], x'_{max} = [500\ 20\ 20],\ u_{min} = -9,\ u_{max} = 9, \tag{2.12}$$

pričom vzdialenosť uvažujeme v [m], jednotlivé rýchlosti v [m.s^{-1}] a zrýchlenie v [m.s^{-2}]. Po prepísaní obmedzení (2.12) do tvaru (2.10), resp. (2.11) dostávame matice *H, K, L a M* nasledovne:

$$H = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad K = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 500 \\ 20 \\ 20 \end{bmatrix} \tag{2.13}$$

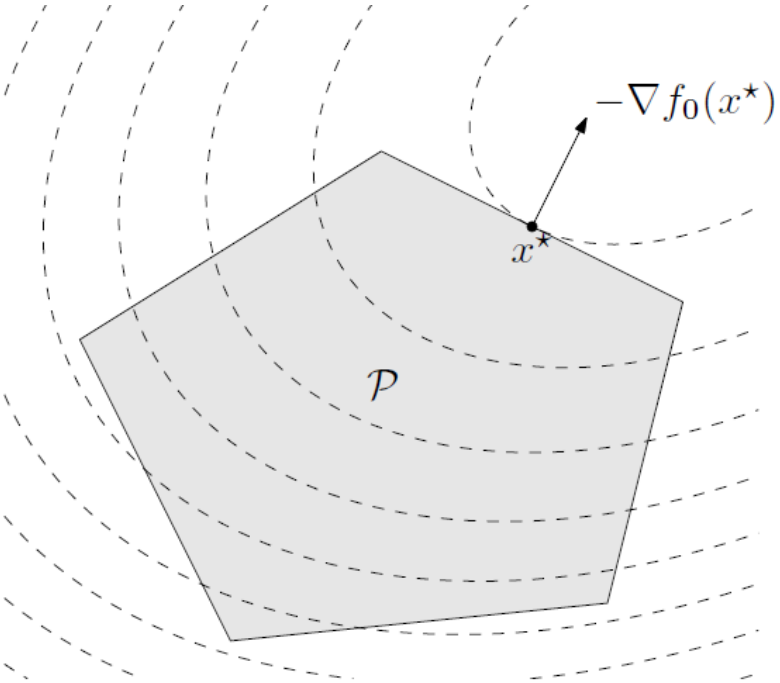
$$L = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad M = \begin{bmatrix} 9 \\ 9 \end{bmatrix} \tag{2.14}$$

2.3 Zostavenie úlohy kvadratického programovania

Pri riešení optimalizačných problémov sa snažíme úlohy formulovať čo najjednoduchšie, resp. tak, aby boli pomerne jednoducho riešiteľné. To je obzvlášť dôležité pri implementácii MPC, ktorým vo všeobecnosti dokážeme riadiť mnohorozmerné systémy. Pre úspešné zavedenie MPC musíme zostaviť optimalizačnú úlohu tak, aby ju bolo možné vyriešiť v každej perióde vzorkovania. Z toho dôvodu sa pri návrhu MPC väčšinou snažíme dopracovať ku konvexnej

optimalizácii. Výhoda spočíva v tom, že keď nájdeme optimum konvexnej funkcie, môžeme povedať, že ide určite o globálne optimum. Ak chceme sformulovať MPC ako problém konvexnej optimalizácie, potrebujeme zabezpečiť, aby jednak účelová funkcia bola konvexná, a aby boli konvexné aj jej obmedzenia.

V kapitole 1.3. sme videli, že za konvexnú účelovú funkciu môžeme zvoliť kvadratickú funkciu. Ak navyše naše obmedzenia budú afinné, dostaneme jednoduchú optimalizačnú úlohu, ktorú je možné riešiť pomerne rýchlo. Ide o známu triedu úloh, ktoré označujeme ako kvadratické programovanie (QP, z angl. Quadratic programming). Na obr. 2.1 vidíme geometrickú reprezentáciu takéhoto problému. Mnohosten *P* je prípustnou množinou, čiarkované čiary predstavujú vrstevnice účelovej funkcie a bod *x** je optimom. [1]



Obr. 2.1 Geometrická ilustrácia QP. [1]

V nasledujúcej časti si ukážeme, ako možno náš problém riadenia pretransformovať na problém QP, a teda do tvaru:

$$\min z^T P z + 2Q^T z + R \tag{2.15}$$

$$\text{s.t. } Gz \leq H \tag{2.15a}$$

$$Az = B. \tag{2.15b}$$

Uvažujme teda účelovú funkciu (2.9a) pri obmedzeniach (2.7), (1.1b), (2.10) a (2.11). Optimalizovanú premennú *d* si označme *y* ako výstup zo systému. Účelová funkcia má teda tvar

$$\min \sum_{k=0}^{N-1} \left((y_{t+k} - y_{ref,k})^T Q_y (y_{t+k} - y_{ref,k}) + u_{t+k}^T Q_u u_{t+k} \right). \tag{2.9b}$$

Keďže rovnica výstupu v našom prípade je $y_{t+k} = C x_{t+k}$, dosadením do (2.9b) dostaneme:

$$\min \sum_{k=0}^{N-1} \left((x_{t+k} - x_{ref,k})^T Q_x (x_{t+k} - x_{ref,k}) + u_{t+k}^T Q_u u_{t+k} \right), \tag{2.9c}$$

pričom matica $Q_x = C^T Q_y C$. Kritérium (2.9c) s obmedzeniami prepíšeme nasledovne:

$$\min \left((X - X_{ref})^T Q_X (X - X_{ref}) + U^T Q_U U \right) \tag{2.16}$$

$$\text{s.t. } X = A_n X + B_n U + E_n x(t) \tag{2.16a}$$

$$H_n X \leq K_n \tag{2.16b}$$

$$L_n U \leq M_n \,, \tag{2.16c}$$

kde X aj U sú matice obsahujúce príslušné vektory stavov a riadiacich vstupov pre celý predikčný horizont. Pri simuláciách sme používali predikčný horizont $N = 10$, a teda matice X a U vyzerajú nasledovne

$$X = \begin{bmatrix} x_t \\ x_{t+1} \\ \vdots \\ x_{t+9} \end{bmatrix} \qquad U = \begin{bmatrix} u_t \\ u_{t+1} \\ \vdots \\ u_{t+9} \end{bmatrix}. \tag{2.17}$$

Nové váhové matice majú tvar $Q_X = diag(Q_x, 10x10)$ a $Q_U = diag(Q_u, 10x10)$, $H_n = diag(H, 10x10)$ *a* $L_n = diag(L, 10x10)$, matice A_n a B_n majú takisto rozmer $10\,x10$, pričom prvý riadok obsahuje len nulové prvky a pod ním sa nachádzajú matice $diag(A, 9x10)$, resp. $diag(B, 9x10)$. Pravé strany nerovností K_n a M_n majú rozmer $10\,x1$ a jednotlivými prvkami sú pôvodné vektory *K*, resp. *M*.
Kritérium (2.16) možno po roznásobení prepísať do tvaru

$$\min \left(X^T Q_X X - 2X_{ref}^T Q_X X + X_{ref}^T Q_X X_{ref} + U^T Q_U U\right). \tag{2.18}$$

Označme $-X_{ref}^T Q_X = S$ a $X_{ref}^T Q_X X_{ref} = R$. Po tomto označení nadobúda účelová funkcia tvar

$$\min \left(X^T Q_X X + 2SX + R + U^T Q_U U\right). \tag{2.19}$$

Ak chceme formulovať úlohu do tvaru (2.15), musíme ďalej zaviesť novú optimalizovanú premennú *z*, v ktorej budú obsiahnuté aj stavy *X*, aj riadiace vstupy *U*, a teda $z = [U\ X]^T$. Ako posledný krok je potrebné vhodne zvoliť matice *P*, *Q* a *R* tak, aby sme dosiahli transformáciu na úlohu QP podľa (2.15). V maticovom zápise bude účelová funkcia s obmedzeniami vyzerať nasledovne:

$$\min z^T \begin{bmatrix} Q_U & 0 \\ 0 & Q_X \end{bmatrix} z + 2[0\ S]z + R. \tag{2.20}$$

$$\text{s.t. } [-B_n\ \ (I - A_n)]z = E_n x(t) \tag{2.20a}$$

$$\begin{bmatrix} 0 & H_n \\ L_n & 0 \end{bmatrix} z \leq \begin{bmatrix} K_n \\ M_n \end{bmatrix} \tag{2.20b}$$

Tým sme sformulovali našu úlohu ako QP problém, jednotlivé matice z (2.15), (2.15a) a (2.15b) sú:

$$P = \begin{bmatrix} Q_U & 0 \\ 0 & Q_X \end{bmatrix}\ Q^T = [0\ S]\quad R = X_{ref}^T Q_X X_{ref}\quad A = [-B_n\ \ (I - A_n)]\quad B = E_n x(t),$$

$$G = \begin{bmatrix} 0 & H_n \\ L_n & 0 \end{bmatrix}\ H = \begin{bmatrix} K_n \\ M_n \end{bmatrix}.$$

Ako bolo povedané, QP je triedou úloh konvexnej optimalizácie a takúto úlohu je možné pomerne rýchlo a efektívne riešiť. Pre praktické aplikácie je nutné pripraviť si všetky matice vystupujúce v QP a potom daný problém riešiť, extrahovať optimálny riadiaci vstup, aplikovať ho na riadený proces a po získaní nových stavov celú procedúru opakovať.

3. Riešenie optimalizačného problému v Matlabe

V kapitole 2.3 sme si ukázali spôsob, akým je možné riešiť problém (2.9) pri obmedzeniach (2.10), resp. (2.11). V praktickej časti sme tento problém riešili pomocou Matlabu a využili sme výhody, ktoré ponúka toolbox Yalmip. V kombinácii s Matlabom bol taktiež použitý výkonný riešiteľ optimalizačných úloh s názvom Gurobi.

Hlavnou výhodou Yalmipu je možnosť zápisu optimalizačných úloh veľmi jednoduchým a intuitívnym spôsobom. To budeme vidieť v ďalšej časti, ktorá je venovaná praktickému riešeniu zostavenej úlohy. Užívateľ Yalmipu nemusí dodržiavať striktnú formu zápisu tak, ako tomu je napríklad pri využívaní funkcií linprog alebo quadprog v Matlabe.

3.1 Definovanie úlohy

V tejto časti si vysvetlíme spôsob vytvorenia súboru *car_platoon_yalmip.m*, ktorý je možné nájsť na priloženom CD. Tento súbor slúži na vytvorenie potrebných premenných a zápis účelovej funkcie s obmedzeniami. Nižšie si uvedieme konkrétne ukážky zdrojového kódu, ktoré si postupne vysvetlíme.

```
A=[1 -1 1; 0 1 0; 0 0 1];
B=[0; 1; 0];
C=eye(3);
D=zeros(3,1);
model = struct('A', A, 'B', B);
```

Tu vidíme zápis stavového opisu systému pomocou matíc (2.8), pričom periódu vzorkovania Δt sme zvolili rovnú jednej sekunde.

```
N = 10;
umin = -8;
umax = 8;
xmin=[0.1; 0; 0];
xmax = [500; 20; 20];
nx = length(xmin);
nu = length(umin);
x = sdpvar(nx, N+1, 'full');
u = sdpvar(nu, N, 'full');
zu = sdpvar(nu, N, 'full');
zx = sdpvar(nx, N+1, 'full');
zd = sdpvar(1, N, 'full');
```

```
Qd = 1;
Qa = 1;
Qzx = 1e3*eye(nx);
Qzu = 1e3*eye(nu);
Qzd = 1e3;
```

V tejto časti sme zapísali predikčný horizont `N`, obmedzenia stavov `xmin, xmax` a obmedzenia vstupov `umin, umax`. Optimalizované premenné `x` a `u` sú v Yalmipe typu `sdpvar`, v zátvorkách sa nastavuje ich rozmer. Môžeme tu vidieť ešte premenné `zx`, `zu` a `zd`, ktoré sa používajú v prípade, keď chceme povoliť tzv. mäkké ohraničenia. Ako posledné sú v tejto časti nastavené váhové matice účelovej funkcie `Qd, Qa`. Vidíme, že váhové matice `Qzx`, `Qzu` a `Qzd` majú nastavené vysoké hodnoty, čím chceme dosiahnuť vysokú penalizáciu, ak sa použijú mäkké ohraničenia.

```
dref = sdpvar(1);
con = [];
obj = 0;
soft_x = false;
soft_u = false;
soft_d = true;

for k = 1:N
    con = con + [x(:, k+1) == A*x(:, k) + B*u(:, k)];
    if soft_x
        con = con + [-zx(:, k)+xmin <= x(:, k) <= xmax+zx(:,k)];
    else
        con = con + [xmin <= x(:, k) <= xmax];
    end
    if soft_u
        con = con + [-zu(:, k)+umin <= u(:, k) <= umax+zx(:,k)];
    else
        con = con + [umin <= u(:, k) <= umax];
    end

    d = x(1, k);
    obj = obj + Qd*(d-dref)^2 + u(:, k)'*Qa*u(:, k);

    if soft_x
        obj = obj + zx(:, k)'*Qzx*zx(:, k);
    end
    if soft_u
        obj = obj + zu(:, k)'*Qzu*zu(:, k);
    end
    if soft_d
        obj = obj + zd(:, k)'*Qzd*zd(:,k);
    end
end
con = con + [ xmin(1) <= dref <= xmax(1) ];
ctrl = optimizer(con, obj, sdpsettings, [x(:, 1); dref], u(:, 1));
```

Tu je najskôr ešte definovaná premenná `dref`, ktorá reprezentuje referenciu pre vzdialenosť medzi autami. Potom nasleduje `con` a `obj`, do nich budeme zapisovať obmedzenia a účelovú funkciu. To vidíme vnútri v cykle „for“, ktorý sa opakuje `N`-krát, kde `N` je dĺžka časového horizontu predikcie. Môžeme si všimnúť napr. zápis nasledujúceho obmedzenia:

```
con = con + [x(:, k+1) == A*x(:, k) + B*u(:, k)];
```

Ide o predikčnú rovnicu a zápis je intuitívny a podobný matematickej formulácii (2.7). Takto jednoducho je možné v Yalmipe zadať celý problém. Vo zvyšku zdrojového kódu sú už zapísané všetky obmedzenia a účelová funkcia. Tu by som ešte chcel poukázať na to, že premenné `soft_x`, `soft_u` a `soft_d` aktivujú jemné ohraničenia v prípade, ak im nastavíme logickú hodnotu `true`. Jemné ohraničenia aktivujeme vtedy, ak nami zadaný problém nemá riešenie. Potom sú k pôvodnému problému pridané nové premenné `zx`, `zu` alebo `zd`, ktoré zväčšujú rozsah obmedzení a sú penalizované veľkými váhovými maticami `Qzx`, `Qzu` a `Qzd`.
Významný je aj posledný riadok súboru *car_platoon_yalmip.m*, kde je vytvorený objekt s názvom `ctrl`. Ten vytvoríme pomocou funkcie `optimizer`, do ktorej ako parametre vstupujú obmedzenia `con`, účelová funkcia `obj`, počiatočná podmienka `x(:, 1)` a referencia `dref`. Výstupom je akčný zásah `u(:, 1)`. Vytvorením objektu `ctrl` sme si vytvorili určitú triedu podobných úloh, ktoré je možné riešiť jednoduchým zavolaním tohto objektu a zadaním príslušných parametrov. Využitie budeme vidieť v kapitole 3.2.

3.2 Simulácia riadenia

V tejto kapitole si na súbore *car_plotn.m* ešte vysvetlíme, akým spôsobom bola realizovaná samotná optimalizácia. Tento súbor predpokladá dva základné simulačné scenáre, ktorých modifikáciou je možné dospieť k všetkým experimentom. Jedným scenárom je situácia, keď prvé auto má zadanú referenčnú trajektóriu rýchlosti, a teda počas simulácie zrýchľuje a spomaľuje a ostatné autá sa snažia dodržiavať zadané vzdialenosti medzi sebou. Druhým scenárom je situácia, keď je zadaná referenčná rýchlosť, ktorá sa nemení, ale mení sa vzdialenosť medzi autami.

```
car_platoon_yalmip
ncars = 4;
showu = false;
scenario = 2;
x0 = [5; 3; 3];
X = cell(1, ncars);
U = cell(1, ncars);
for i = 1:ncars
    X{i} = x0';
end
```

Ako prvé tu voláme súbor `car_platoon_yalmip`, kde máme definovanú účelovú funkciu, obmedzenia a potrebné konštanty. Zadáme počet áut `ncars`, vyberieme simulačný scenár `scenario` a zadáme počiatočnú podmienku `x0`. Do polí `X` a `U` budeme ukladať stavy a akčné zásahy pre jednotlivé autá. Na začiatku je `X` pre každé auto inicializované počiatočnou podmienkou `x0`.

```
speed_ref_target = kron([4; 8; 4; 7; 0], ones(15, 1));
f = tf(1, [1 1]);
speed_ref_target = lsim(f, speed_ref_target, 1:length(speed_ref_target));
distance_ref_f1 = kron([5; 10; 3], ones(25, 1));
distance_ref_other = 8*ones(75, 1);
```

Tu si vo vhodnom tvare pripravíme referenciu pre rýchlosť prvého auta `speed_ref_target`. Tá je použitá v prípade, ak premenná `scenario` nemá hodnotu 3. Takisto sa tu nastavujú referencie pre vzdialenosti medzi autami.

```
if scenario==2
    distance_ref_f1 = 8*ones(75, 1);
elseif scenario==3
    speed_ref_target = 15*ones(75, 1);
end
```

Pomocou vyššie uvedeného kódu sa už len trajektórie žiadaných veličín menia v závislosti od voľby simulačného scenára.

```
for k = 1:length(speed_ref_target)
    xn = cell(1, ncars);
    for i = 1:ncars
        if i==1
            xc = [X{1}(end, 1:2)'; speed_ref_target(k)];
            if isa(ctrl, 'optimizer')
                u = ctrl{[xc; distance_ref_f1(k)]};
            else
                u = ctrl([xc; distance_ref_f1(k)]);
            end

        else
            xc = [X{i}(end, 1:2)'; X{i-1}(end, 2)];
            if isa(ctrl, 'optimizer'),
                u = ctrl{[xc; distance_ref_other(k)]};
            else
                u = ctrl([xc; distance_ref_other(k)]);
            end
        end
        if any(isnan(u))
            error('Problem is infeasible.');
        end


        xn{i} = model.A*xc+model.B*u;
        xn{i}(xn{i}<0)=0;
        U{i} = [U{i}; u];
    end
    for i = 1:ncars
        X{i} = [X{i}; xn{i}'];
    end

end
```

V tejto časti zdrojového kódu vidíme dva vnorené cykly, vonkajší sa opakuje až kým `k` dosiahne hodnotu dĺžky vektora `speed_ref_target`, čo vyjadruje celkový čas simulácie. Vnútorný cyklus sa opakuje pre každé auto. Do pomocnej premennej `xc` si ukladáme aktuálne hodnoty prvých dvoch stavov(vzdialenosť od auta pred nami, našu rýchlosť) a žiadanú hodnotu rýchlosti. Tu rozlišujeme, či ide o prvé auto, alebo ostatné autá. Ak má premenná `i` hodnotu 1, ide o prvé auto, a ako žiadaná hodnota rýchlosti sa použije príslušný prvok vektora `speed_ref_target`. Ak má `i` hodnotu väčšiu ako 1, ako žiadaná rýchlosť sa použije rýchlosť auta idúceho pred nami, čiže vyberieme hodnotu druhého stavu pre auto s indexom `i-1` (`X{i-1}(end, 2)`). V kapitole 3.1. sme si vysvetlili význam objektu `ctrl`, ktorý vznikol použitím funkcie `optimizer`. V tejto časti sa už v každom kroku vypočítava akčný zásah a to tak, že objektu `ctrl` zadáme ako vstupné parametre pomocnú premennú `xc` a žiadanú hodnotu vzdialenosti od predchádzajúceho auta. Žiadaná vzdialenosť je v prípade prvého auta

`distance_ref_f1`, inak je to `distance_ref_other`. Vypočítaná hodnota akčného zásahu je priradená do premennej `u`.

Nový výpočet stavu sa potom vykoná dosadením akčného zásahu do predikčnej rovnice a výsledok je priradený do premennej `xn`. Hodnoty akčných zásahov a stavov sa v každom kroku pridávajú do polí `U` a `X`, ktoré neskôr slúžia na grafické znázornenie priebehov riadenia. Vykresľovanie týchto priebehov zabezpečuje zvyšná časť súboru *car_plotn.m*, to na tomto mieste nie je potrebné hlbšie rozoberať.

4. Simulačné výsledky

V predchádzajúcej kapitole sme si opísali m-súbory, ktoré slúžili na simulácie rôznych situácii, s ktorými sa v praktickom živote stretávame. V nasledujúcich kapitolách si vysvetlíme uvažované scenáre a ukážeme grafické výsledky dosiahnutého riadenia.

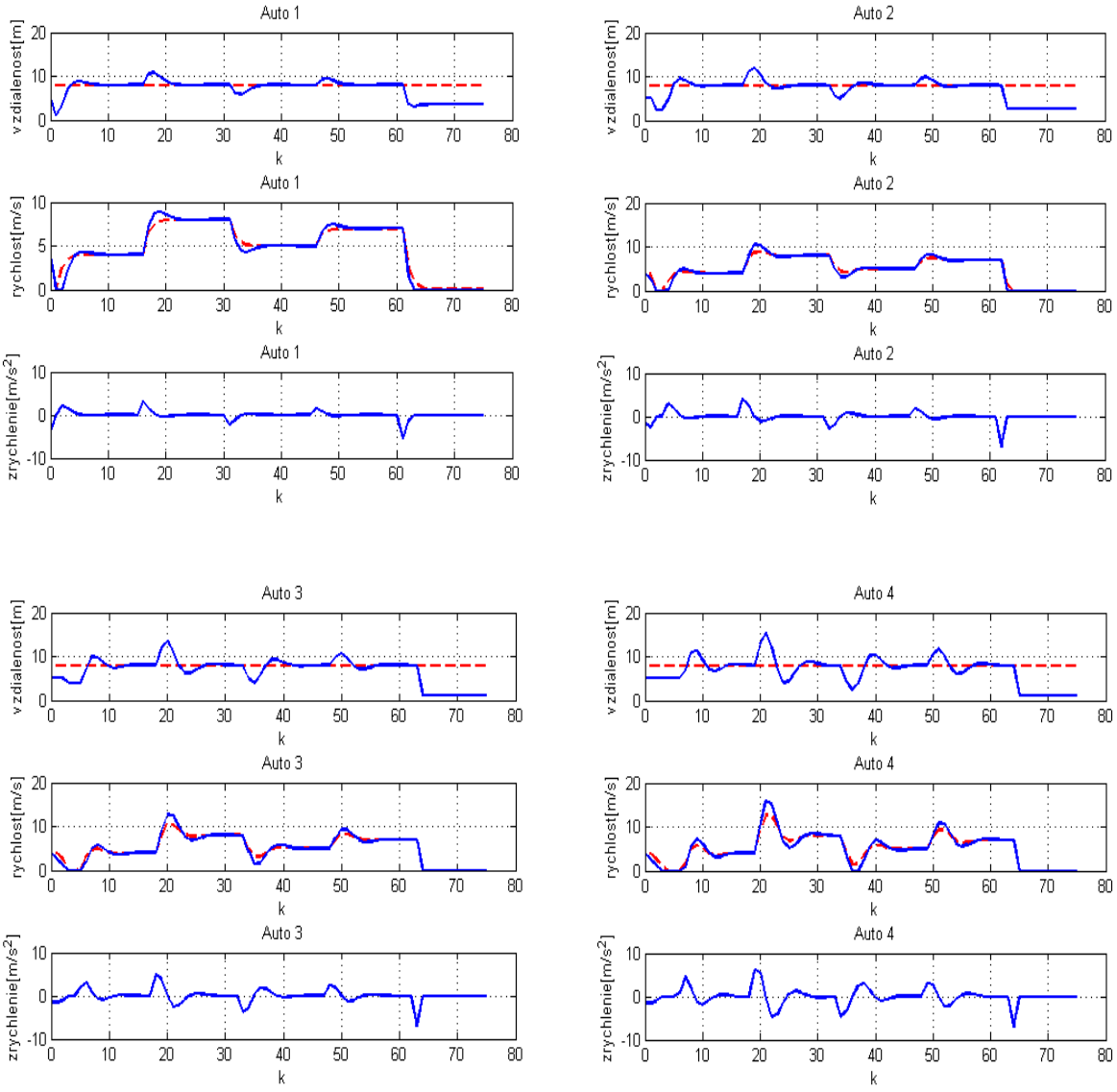
4.1 Scenár 1

Ako prvú predpokladajme situáciu, že riadime 4 autá idúce za sebou. Prvé auto má zadanú referenčnú trajektóriu rýchlosti. Pre každé ďalšie auto je žiadanou rýchlosťou rýchlosť auta idúceho pred ním. Autá majú predpísané žiadané vzdialenosti medzi sebou, takisto vedúce auto má zadanú vzdialenosť od virtuálneho lídra, ktorý reprezentuje neriadené auto idúce pred kolónou. V tabuľke 4.1 sú znázornené všetky referencie pre rýchlosti a vzdialenosti použité v tomto simulačnom scenári. V riadku pre prvé auto vidíme 5 hodnôt rýchlostí, čo znamená, že každá hodnota platí pre 15 simulačných krokov, čiže 15 sekúnd. Tento vektor rýchlostí bol ešte upravený tak, aby sa priebeh rýchlosti nemenil skokovo, ale ako systém opísaný prenosom prvého rádu.

	Referenčná rýchlosť[$m.s^{-1}$]	Referenčná vzdialenosť[m]
Prvé auto	4, 8, 5, 7, 0	8
Ostatné autá	rýchlosť predchádzajúceho	8
Počiatočná podmienka(všetky)	4	5

Tab. 4.1 Vstupné údaje pre prvú simuláciu.

Na obr. 4.1 vidíme grafické priebehy tohto simulačného scenára. Červená čiarkovaná čiara predstavuje žiadanú trajektóriu pre príslušnú veličinu. Modrou čiarou sú znázornené simulačné priebehy. Každé grafické okno na obrázku prislúcha jednému autu a sú v ňom postupne zobrazené dva stavy(vzdialenosť k predchádzajúcemu autu, rýchlosť) a riadiaci zásah (zrýchlenie).

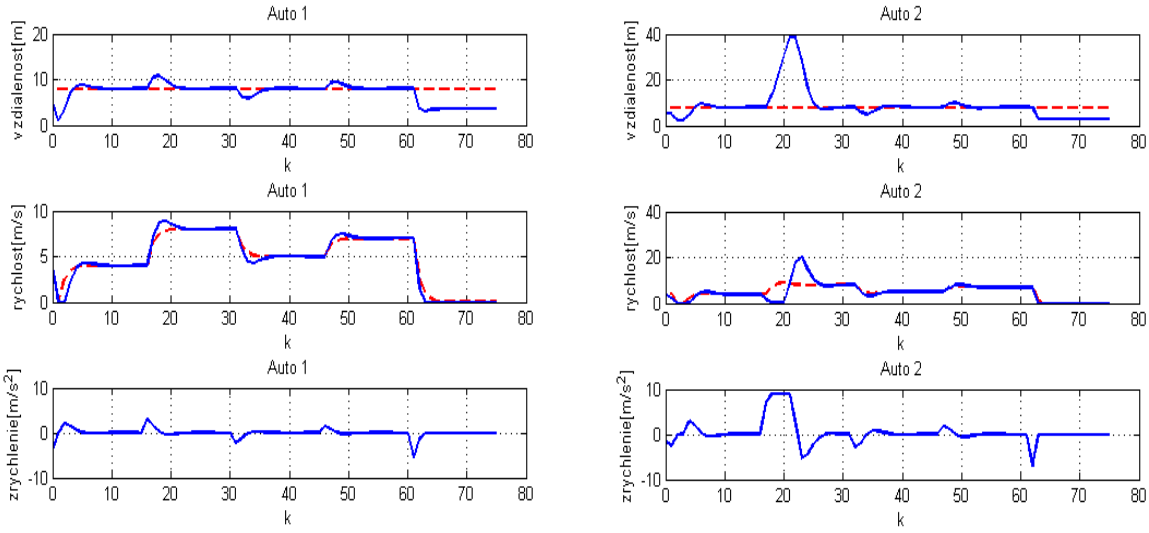


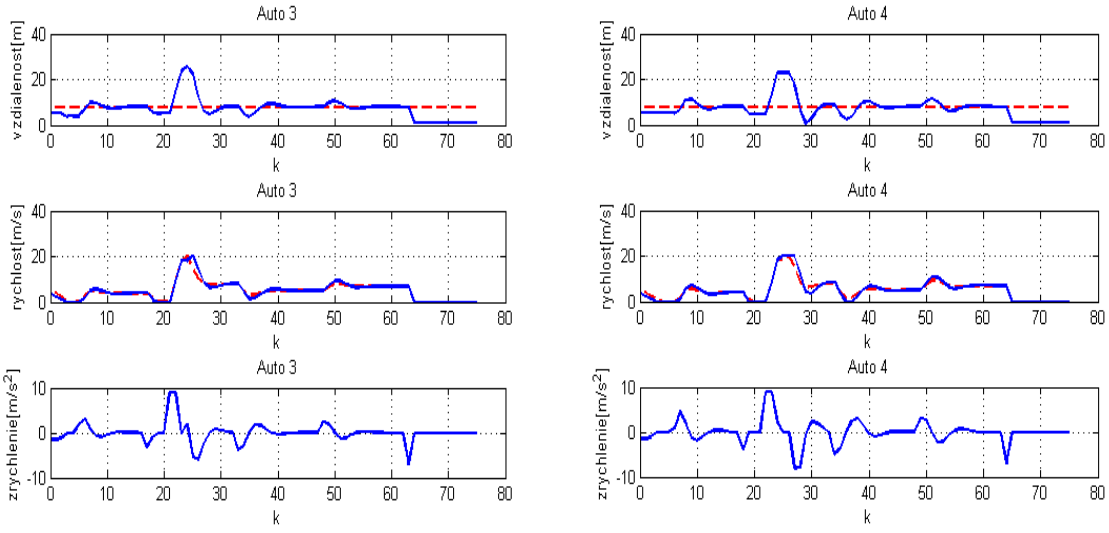
Obr. 4.1 Grafické priebehy prvého simulačného scenára.

Ak chceme zhodnotiť výsledky tejto simulácie, je potrebné určiť kritériá, na základe ktorých budeme simulácie posudzovať. Základným kritériom pre tento prípad bolo, aby pri použití dostupných akčných zásahov autá do seba nenarazili. Vybrali sme tento ilustračný prípad, kde simulujeme jazdu štyroch áut za sebou, pričom sú použité referenčné údaje z tab. 4.1. Výsledky simulácie sú v tomto prípade uspokojivé, autá do seba nenarazili a pomerne dobre sledovali referencie pre rýchlosť.

Tu treba však povedať, že prístup, pri ktorom každé auto má informáciu len o stave auta idúceho ihneď pred sebou, má svoje obmedzenia. Pre každé auto je žiadanou rýchlosťou rýchlosť predchádzajúceho auta, a keďže túto trajektóriu nie je možné kopírovať úplne presne, pri každom aute vzniká určité skreslenie. Toto skreslenie mení krivku pôvodnej trajektórie zadanej pre prvé auto, čo znamená, že pri vyššom počte áut nastane situácia, že do seba narazia. Samozrejme, tento nedostatok je možné eliminovať tým, že autá budú medzi sebou udržovať väčšiu vzdialenosť, a tým pádom sa nezrazia ani pri výraznejších zmenách rýchlosti.

Pre názornejšie zobrazovanie simulačných výsledkov sme vytvorili m-súbor, ktorý získané dáta pretransformuje do jednoduchej animácie. Vytvorenie animácie je podrobnejšie vysvetlené v poslednej kapitole. Pre jednoduché spustenie bez potreby použitia Matlabu sme všetky animácie prerobili do formátu *.avi a nachádzajú sa na priloženom CD. Pre tento konkrétny scenár je potrebné spustiť video s názvom scenar1.avi.





Obr. 4.2 Grafické priebehy prvého simulačného scenára so zastavením auta 2.

Na obr. 4.2 vidíme simulačné priebehy prvého scenára s drobnou modifikáciou. Vstupné údaje pre simuláciu sú zhodné s tabuľkou 4.1, jediným rozdielom je zastavenie druhého auta v čase 15 sekúnd. Tým sme simulovali prípad, keď ide za sebou kolóna áut a jednému z nich(druhé auto) sa v ceste vyskytne prekážka, napr. chodec, a teda auto aj zvyšok kolóny musí zastaviť. Z grafu sa môže na prvý pohľad zdať, že štvrté auto v čase 28 narazilo, v skutočnosti má vzdialenosť od tretieho 0,54 m. Vidíme, že keďže druhé auto musí stáť, do času 20 mu narastá vzdialenosť od prvého auta. Potom keď už prekážku nemá, jeho rýchlosť sa zväčšuje tak, aby opäť dosiahol žiadanú vzdialenosť od prvého. To isté platí aj pre ostatné autá. Takisto je možné na priloženom CD nájsť animáciu tohto prípadu, potrebné je spustiť video s názvom scenar1a.avi.

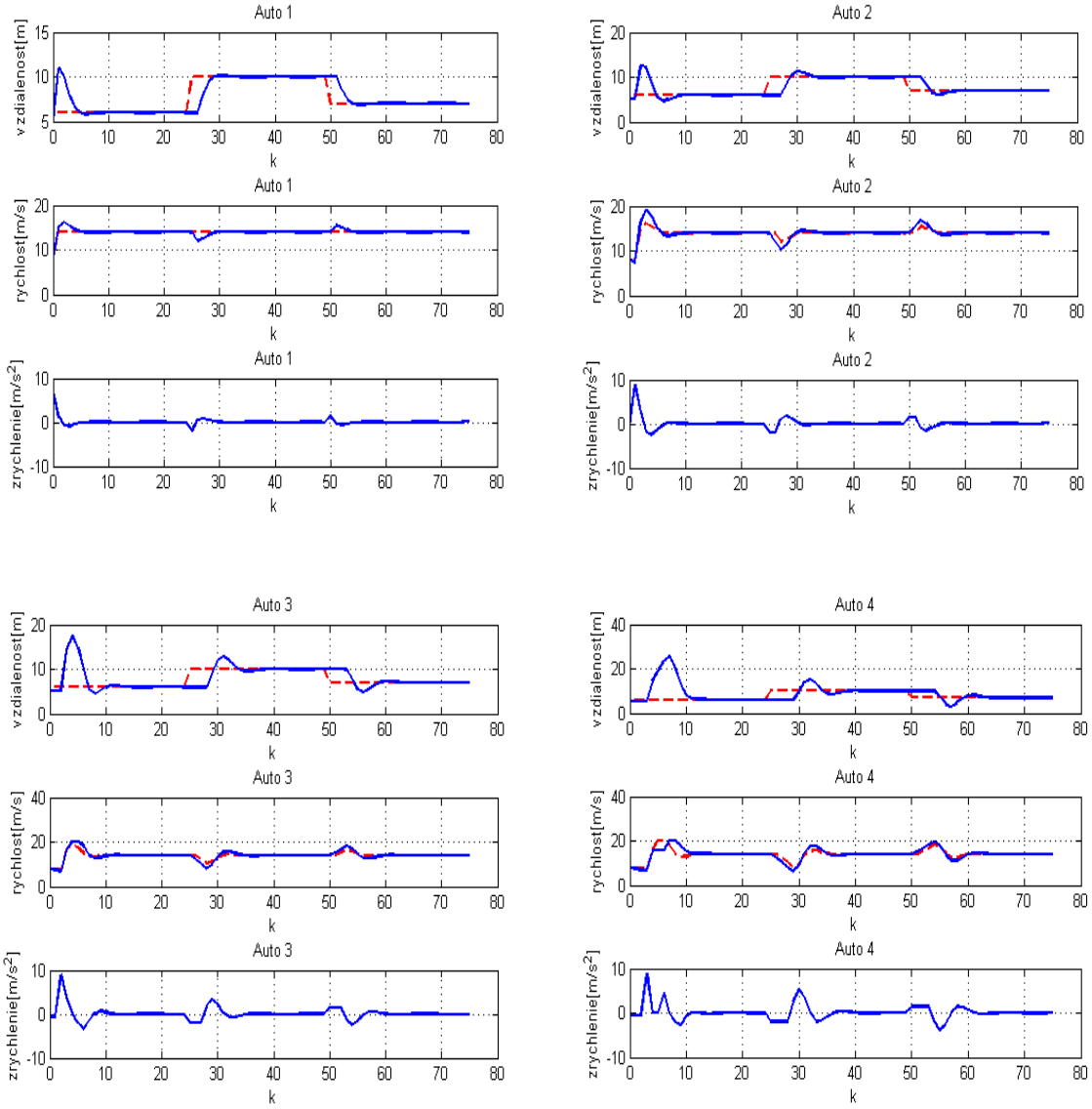
4.2 Scenár 2

Pri simulácii tohto scenára sme predpokladali, že prvé auto bude dodržiavať konštantnú rýchlosť, meniť sa ale budú žiadané vzdialenosti medzi autami. Všetky použité vstupné údaje pre túto simuláciu sú zhrnuté v tabuľke 4.2. Počet áut uvažovaných pre tento prípad bol opäť 4.

	Referenčná rýchlosť[m.s^{-1}]	Referenčná vzdialenosť[m]
Prvé auto	14	6, 10, 7
Ostatné autá	rýchlosť predchádzajúceho	6, 10, 7
Počiatočná podmienka(všetky)	8	5

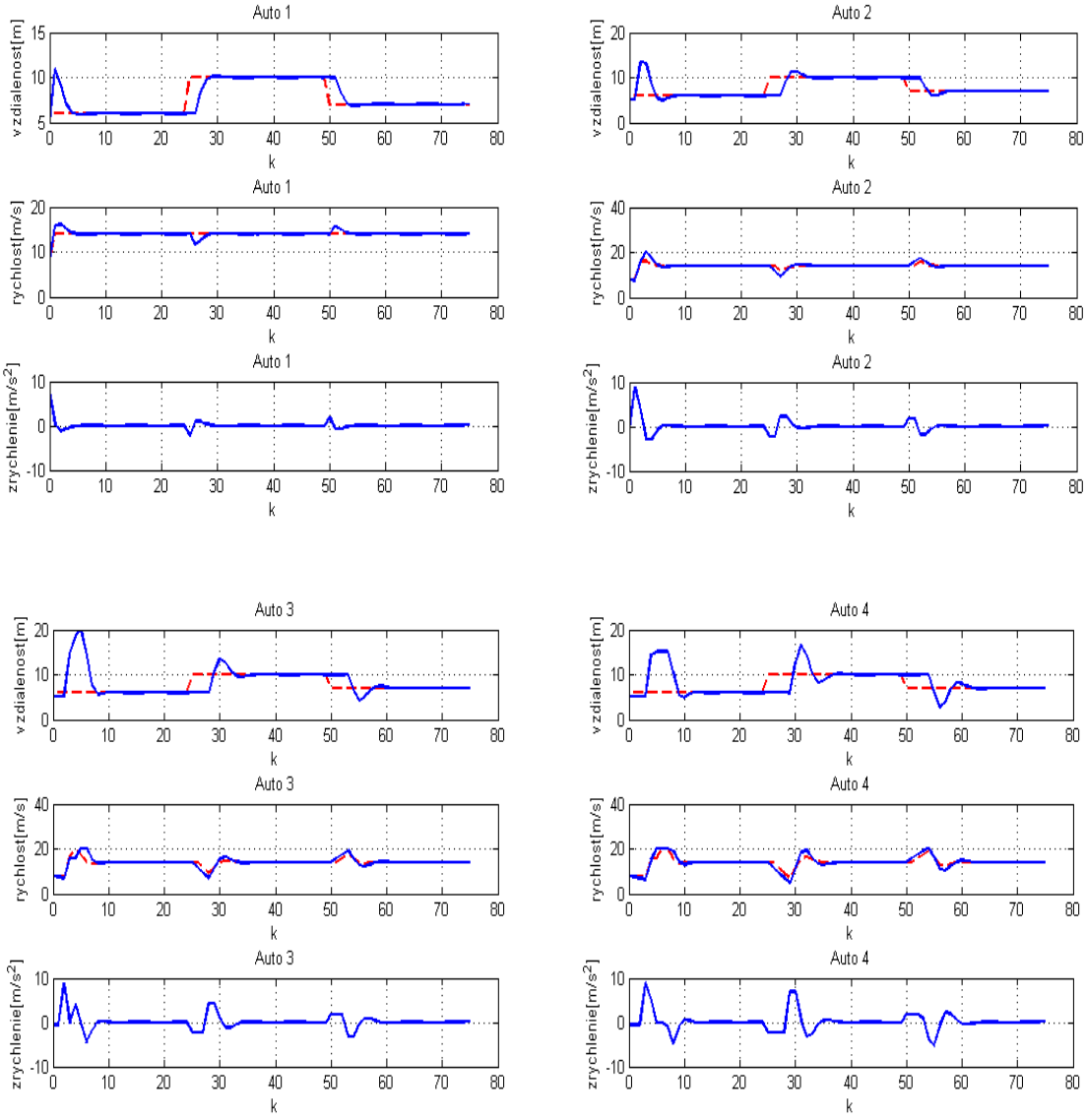
Tab. 4.2 Vstupné údaje pre druhú simuláciu.

Grafické výsledky tejto simulácie sú znázornené na obr. 4.3.



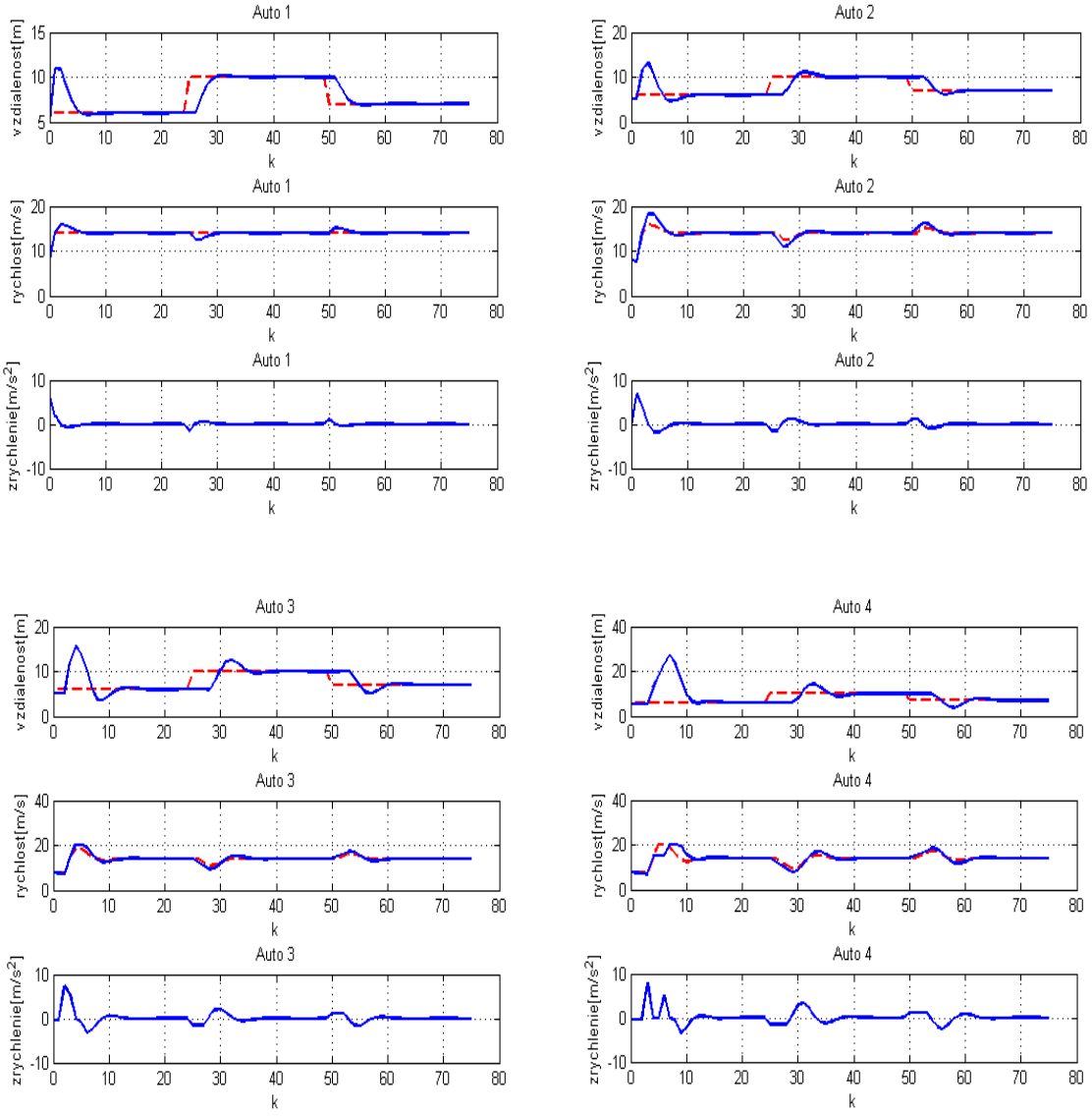
Obr. 4.3 Grafické priebehy pre druhý simulačný scenár, $\boldsymbol{Q_d = Q_a = 1}$.

Pri tejto simulácii sme použili váhové matice účelovej funkcie Q_d a Q_a (kap. 3.1) rovné jednej. Zaujímalo nás, aký vplyv bude mať zmena pomeru váhových matíc na simulačný priebeh a na priebeh riadiacich vstupov. Na obr. 4.4 sú znázornené simulačné priebehy, keď bola váhová matica Q_d dvakrát väčšia ako Q_a.



Obr. 4.4 Grafické priebehy pre druhý simulačný scenár, $\boldsymbol{Q_d = 2}$, $\boldsymbol{Q_a = 1}$.

Aby sme mohli pozorovať vplyv zmien váhových matíc účelovej funkcie na simulačné priebehy, uskutočnili sme ešte jednu simuláciu, kde bola naopak váhová matica Q_d dvakrát menšia v porovnaní s Q_a. Tento priebeh môžeme vidieť na obr. 4.5.



Obr. 4.5 Grafické priebehy pre druhý simulačný scenár, $Q_d = 1$, $Q_a = 2$.

Predpokladali sme, že ak budeme viac penalizovať odchýlku od žiadanej vzdialenosti medzi autami, regulačná odchýlka bude menšia a na jej zmenšenie budú potrebné väčšie akčné zásahy. V tabuľke 4.3 je uvedené porovnanie týchto troch vyššie spomenutých variant druhého scenára.

Q_d, Q_a	Suma kladných u [m.s^{-2}]	IAE
1, 1	95,06	116,07
2, 1	107,40	89,33
1, 2	84,82	119,35

Tab. 4.3 Porovnanie kvality riadenia pre rôzne váhové matice.

Z údajov uvedených v tabuľke 4.3 vyplýva fakt, ktorý sme predpokladali. Ak zväčšíme hodnotu váhovej matice, ktorá penalizuje rozdiel medzi žiadanou a skutočnou vzdialenosťou medzi autami, suma regulačných odchýlok bude menšia, musíme však na to vynaložiť väčšiu snahu v podobe akčných zásahov. Naopak, ak budeme zväčšovať penalizáciu riadiacich vstupov, kvalita riadenia bude horšia, ale na jej dosiahnutie budú postačujúce menšie akčné zásahy.
Z praktického hľadiska môžu mať uplatnenie oba prístupy. Ak budeme požadovať striktné dodržanie žiadaných rozostupov, budú na to potrebné väčšie akčné zásahy, čo sa ale premietne aj do spotreby paliva pri takomto spôsobe jazdy. Menšia penalizácia regulačnej odchýlky môže priniesť jednak nižšiu spotrebu paliva, ako aj väčší komfort jazdy pre pasažierov, keďže sa akčné zásahy nebudú až tak prudko meniť.

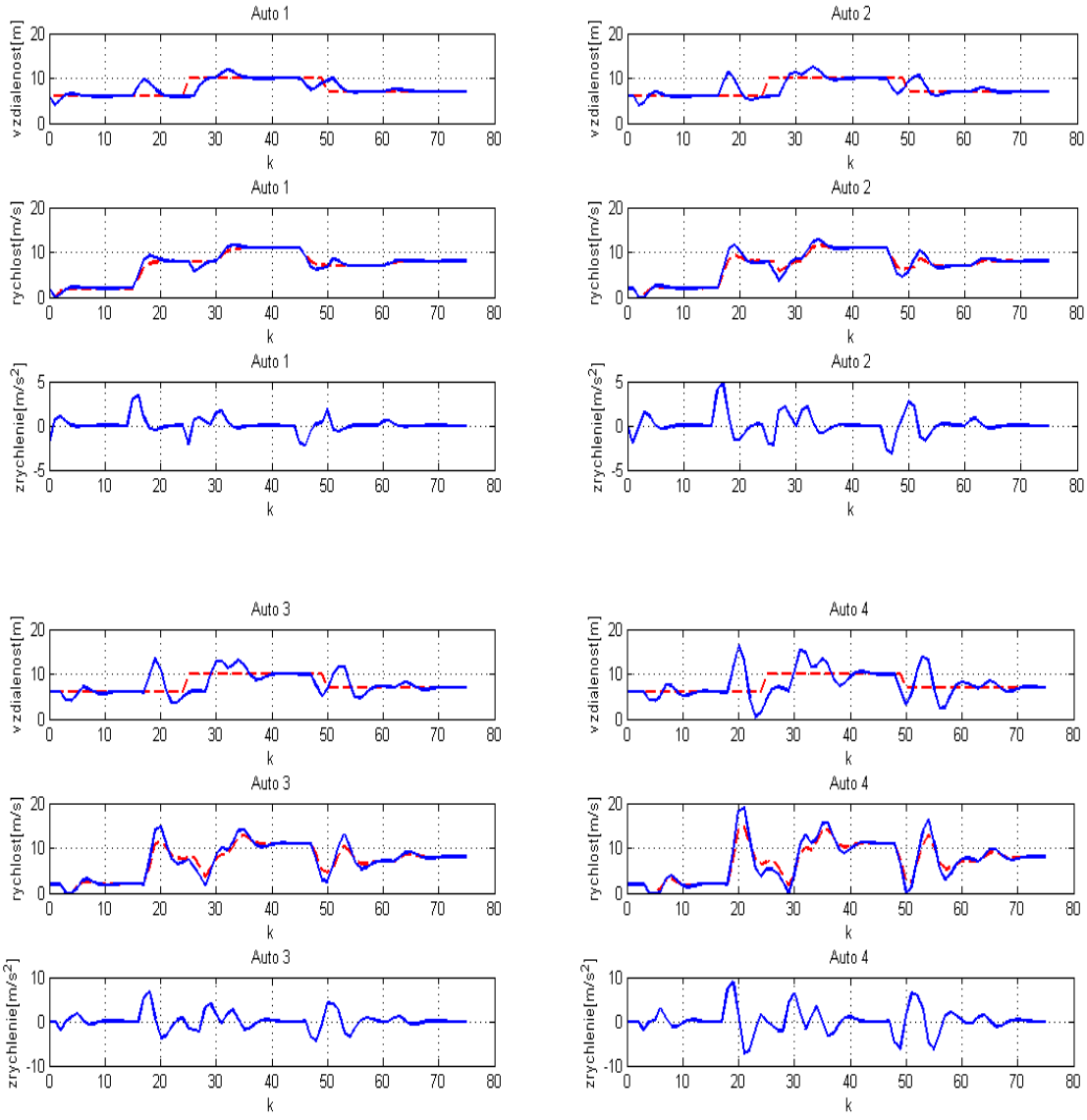
4.3 Scenár 3

Pri tejto simulácii sme uvažovali prípad, pri ktorom sa mení aj rýchlosť vedúceho auta, aj požadované rozostupy. V tabuľke 4.4 sú zhrnuté vstupné údaje použité pri tomto simulačnom scenári.

	Referenčná rýchlosť[m.s^{-1}]	Referenčná vzdialenosť[m]
Prvé auto	2, 8, 11, 7, 8	6, 10, 7
Ostatné autá	rýchlosť predchádzajúceho	6, 10, 7
Počiatočná podmienka(všetky)	2	6

Tab. 4.4 Vstupné údaje pre tretí simulačný scenár.

Na obr. 4.6 sú zobrazené grafické priebehy odpovedajúce tomuto simulačnému scenáru.



Obr. 4.6 Grafické priebehy pre tretí simulačný scenár.

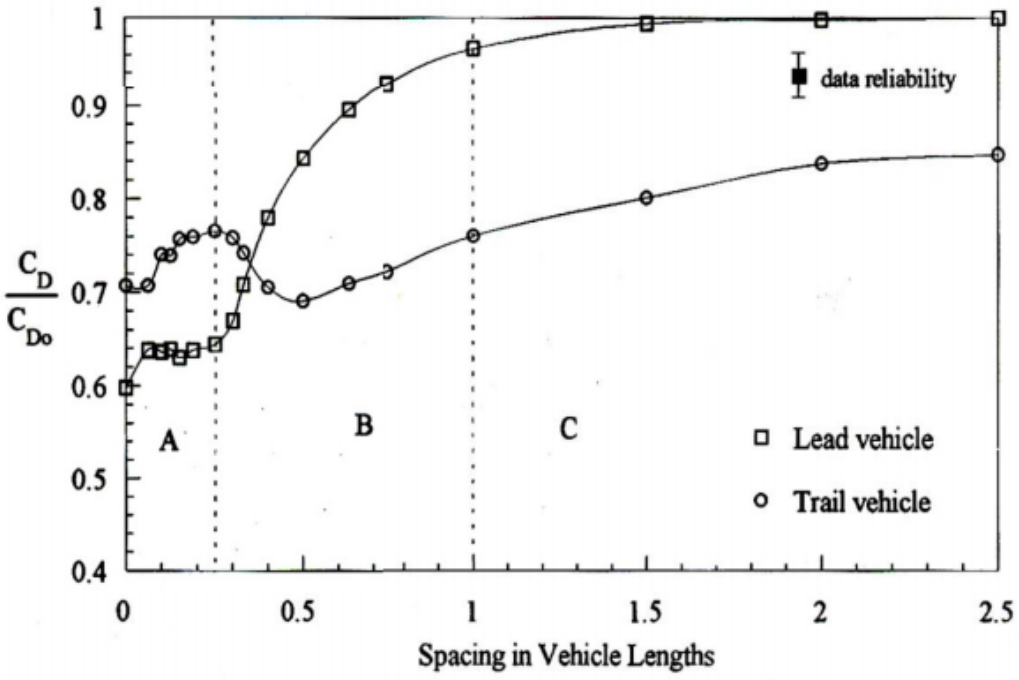
Ako vidíme v tomto prípade, riadenie na meniace sa rozostupy nie je až tak jednoduché, keďže sa mení aj referenčná rýchlosť. Po určitom čase sa ale každému autu podarilo dosiahnuť žiadaný odstup od predchádzajúceho. Samozrejme, riadenie pri takomto scenári si vyžaduje aj pomerne agresívne akčné zásahy, čo vidíme najmä pri pohľade na priebeh riadiacich vstupov pri štvrtom vozidle.

4.4 Určenie optimálneho rozostupu medzi dvoma autami

Jedným z dôležitých kritérií riadenia môžu byť prevádzkové náklady, konkrétne spotreba paliva. Je zrejmé, že spotreba paliva je v našom prípade úmerná sume kladných akčných zásahov. Spotrebu môžeme v zásade ovplyvňovať dvomi spôsobmi. Ak ju chceme znížiť, je to možné pomocou váhových matíc v účelovej funkcii. Zväčšením hodnoty váhovej matice, ktorá penalizuje akčný zásah dosiahneme zníženie spotreby, avšak kvalita riadenia sa zhorší.
Druhým smerom, ktorým sa možno uberať, je analýza spotreby paliva v súvislosti s aerodynamickým odporom pri pohybe vozidla. Je zrejmé, že jednou zo síl ktoré musí motor pri pohybe auta prekonať, je sila aerodynamického odporu. Pre výpočet tejto sily platí nasledovný vzťah:

$$F_o = \frac{1}{2} CSqv^2, \tag{4.1}$$

kde S je čelný prierez telesa, C je koeficient aerodynamického odporu, q je hustota prostredia a v je rýchlosť telesa. [8] Zo vzťahu (4.1) je vidieť, že sila aerodynamického odporu je úmerná koeficientu aerodynamického odporu, a teda čím nižší bude tento koeficient, tým nižšia bude spotreba paliva. Bolo vypracovaných niekoľko štúdií, ktoré sa zaoberali meraním hodnoty tohto koeficientu pre autá idúce tesne za sebou. Bol zistený fakt, že aerodynamický odpor závisí od toho, aké sú rozostupy medzi autami [6]. Na obr. 4.7 sú namerané závislosti aerodynamického odporu od rozostupu pre dve vozidlá idúce za sebou. Tieto hodnoty boli získané na základe meraní v tzv. veternom tunely. Rozostup je meraný v dĺžkach vozidla a na druhej osi je pomer koeficientu aerodynamického odporu pre prípad izolovanej jazdy a jazdy s druhým vozidlom.

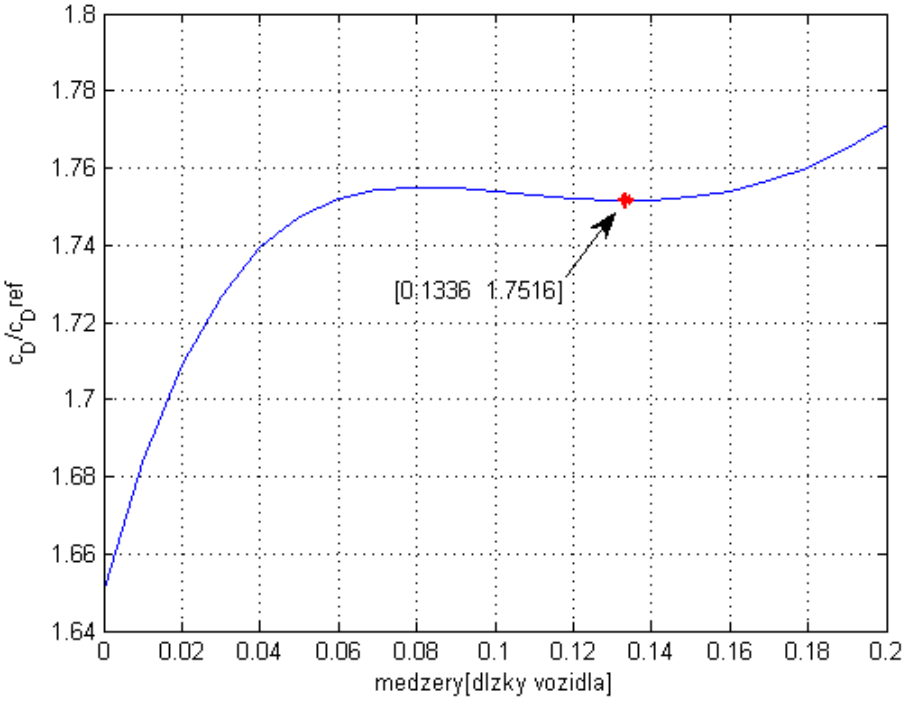


Obr. 4.7 Relatívny koeficient aerodynamického odporu v závislosti od rozostupu pre 2 autá. [6]

Keď spravíme súčet priebehov na obr. 4.7, zistíme, že výsledná funkcia má rastúci priebeh. Ak by sme požadovali úsporu paliva pre dve autá ako celok, znamená to, že by mali mať medzi sebou čo najmenší rozostup. To by ale znamenalo, že vedúce vozidlo by šetrilo palivo na úkor vozidla idúceho za ním. Ďalším problémom je potom bezpečnosť, pretože by vozidlá museli ísť za sebou s takmer nulovým rozostupom. Preto je z praktického hľadiska významný rozostup 0,35 dĺžky vozidla, pričom obe autá majú rovnakú hodnotu koeficientu aerodynamického odporu. V kapitole 4.6 si ukážeme ďalší simulačný scenár, ktorý je založený na týchto úvahách.

4.5 Určenie optimálneho rozostupu medzi troma autami

Z práce [4], v ktorej sa vedci zaoberali meraním aerodynamického odporu vo veternom tunely sme získali údaje pre tri autá idúce tesne za sebou. Mali sme k dispozícii grafické závislosti relatívneho koeficientu aerodynamického odporu od vzdialenosti medzi vozidlami pre každé auto. Výsledná závislosť zložená zo všetkých troch je zobrazená na obr. 4.8.



Obr. 4.8 Výsledná závislosť relatívneho koeficientu aerodynamického odporu od medzier medzi vozidlami pre 3 autá.

Červená hviezdička vyznačená v grafe predstavuje optimálny rozostup pre 3 autá. Vidíme, že relatívny koeficient aerodynamického odporu nadobúda nižšie hodnoty, ak idú autá ešte bližšie pri sebe, ale riadenie na tak malé hodnoty rozostupov nie je prípustné. Preto sme hľadali extrém na intervale $x > 0{,}1$. V kap. 4.7 sa budeme teda zaoberať simulačným scenárom, pri ktorom budeme za optimálnu vzdialenosť medzi vozidlami považovať 0,1336 dĺžky vozidla.

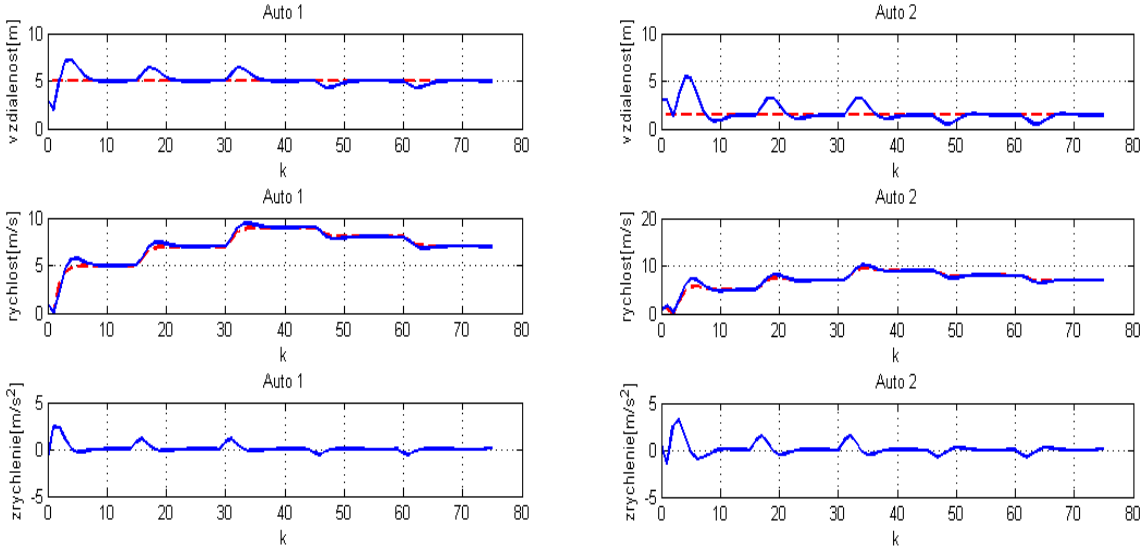
4.6 Scenár 4

V tomto príklade sme uvažovali prípad jazdy dvoch vozidiel za sebou s takou vzdialenosťou medzi nimi, aby boli ich náklady na jazdu čo najnižšie. Pri tejto simulácii sme predpokladali, že prvé vozidlo pôjde za virtuálnym lídrom, od ktorého dodržuje istú vzdialenosť a počas jazdy sa mení jeho rýchlosť. Druhé auto sa snaží udržiavať od prvého vzdialenosť 0,35 z dĺžky vozidla. Všetky vstupné údaje uvažované pri simulácii sú zhrnuté v tabuľke 4.5.

	Referenčná rýchlosť[m.s^{-1}]	Referenčná vzdialenosť[m]
Prvé auto	5, 7, 9, 8, 7	5
Druhé auto	rýchlosť predchádzajúceho	0,35*dĺžka vozidla(0,35*4)
Počiatočná podmienka	1	3

Tab. 4.5 Vstupné údaje pre štvrtú simuláciu.

V tomto príklade je opäť dôležité, aby sa autá nezrazili, keďže simulujeme veľmi malú vzdialenosť medzi nimi. Grafický výsledok tejto simulácie máme k dispozícii na obr. 4.9.



Obr. 4.9 Grafické priebehy štvrtého simulačného scenára.

Tu je potrebné pripomenúť, že žiadaná hodnota vzdialenosti(červená čiarkovaná čiara) pre druhé auto je rovnako výhodná pre obidve vozidlá, pretože majú výrazne nižšiu hodnotu koeficientu aerodynamického odporu, ako keby išli samostatne, alebo s väčším rozostupom. Takisto chcem zdôrazniť, že takáto malá vzdialenosť medzi autami v prípade drastickejších zmien rýchlosti lídra nie je prípustná a fyzikálne nie je možné v takom prípade autá bezpečne riadiť. V reálnom živote si však môžeme ľahko predstaviť situáciu, kedy idú dve vozidlá na komunikácii mimo mesta alebo na diaľnici a premávka nie je natoľko zhustená, aby bolo nutné prudkým spôsobom meniť rýchlosť jazdy. V takom prípade by bolo možné aplikovať aj riadenie na tak malý odstup.

Ďalším významným príkladom využitia môže byť nákladná preprava v noci. V noci by mohli kamióny využívať takýto štýl jazdy, aby ušetrili náklady na dopravu. Štúdia, ktorú robili v Kalifornii v súvislosti so znížením spotreby v závislosti od vzdialenosti medzi idúcimi autami hovorí o úspore 6 - 7 % paliva pri rozostupe medzi autami 0,8 dĺžky vozidla. [2] Aj pre túto simuláciu sa animácia nachádza na priloženom CD, názov videa je scenar4.avi.

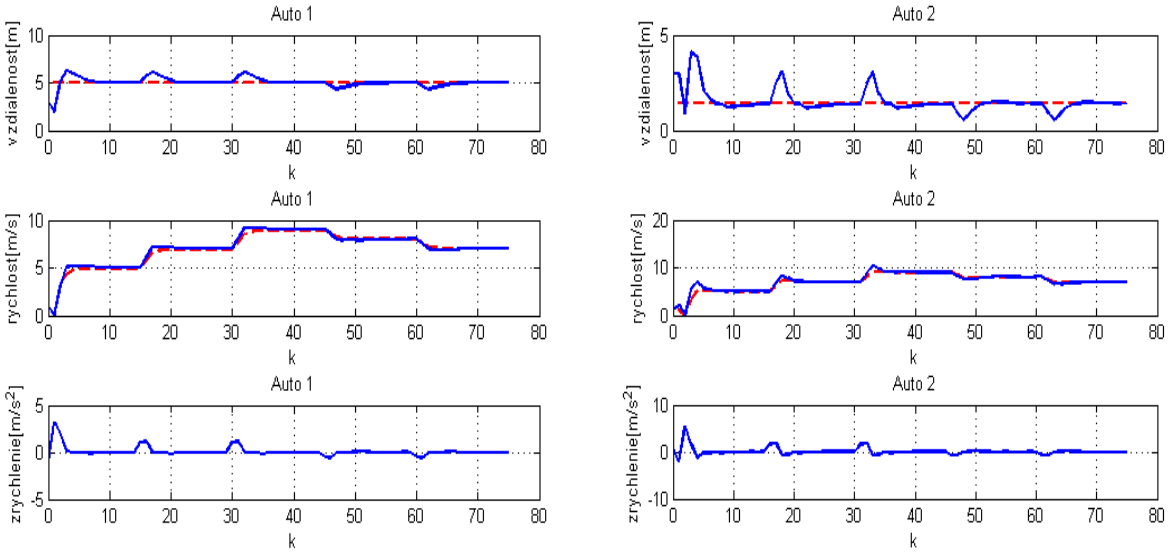
Pri tomto simulačnom scenári bolo použité ešte jedno riadenie, ktoré by mohlo priniesť ďalšiu úsporu paliva, npar. pri riadení nákladnej prepravy. Pôvodne bolo na výpočet optimálneho akčného zásahu použité kritérium (2.9), pri ktorom sa minimalizuje kvadrát rozdielu aktuálneho a žiadaného rozostupu a akčnej veličiny. V ďalšom skúmaní bolo požité nové kritérium, ktoré už nepenalizuje akčný zásah, ale pomocnú premennú *E*, pre ktorú platí:

$$E = \max\{0, u\}, \tag{4.2}$$

kde u vystupovalo v pôvodnej účelovej funkcii a predstavuje akčný zásah. Znamená to, že použitím takejto účelovej funkcie už penalizujeme len kladné akčné zásahy, čiže akceleráciu a nie brzdenie. Nová účelová funkcia má teda podobu (4.3), pribudli k nej 2 obmedzenia:

$$\min \sum_{k=0}^{N-1} \left((d_{t+k} - d_{ref,k})^T Q_d (d_{t+k} - d_{ref,k}) + Q_E E_{t+k} \right). \tag{4.3}$$
$$E \geq 0, \qquad E \geq u$$

S použitím takejto alternatívy sme urobili simuláciu, ktorá bola inak realizovaná pri tých istých podmienkach, ktoré sú uvedené v tabuľke 4.5. Cieľom bolo porovnanie súm kladných akčných zásahov, ako jedného z kritérií určujúcich spotrebu paliva. Zároveň nás zaujímala aj kvalita riadenia, tú sme porovnali na základe sumy absolútnych regulačných odchýlok IAE (z angl. Integral absolute value of error). Na obr. 4.10 je výsledok takto modifikovaného riadenia, a v tabuľke 4.6 sú zhrnuté spomínané kritériá.



Obr. 4.10 Grafické priebehy modifikovaného štvrtého simulačného scenára.

Po porovnaní súm kladných akčných zásahov v tabuľke 4.6 môžeme konštatovať, že zavedením novej optimalizovanej premennej *E* by sme mohli dosiahnuť ďalšiu úsporu paliva, keďže zrýchľovanie priamo súvisí so spotrebou. Rozdiel je približne 6,2 %. Keď sa pozrieme na hodnoty IAE ako ukazovateľa kvality, takisto možno konštatovať lepší výsledok.

	Suma kladných u[m.s^{-2}]	IAE
Pôvodná účelová funkcia	27,49	33,51
Nová účelová funkcia	25,74	23,31

Tab. 4.6 Porovnanie kritérií riadenia pre pôvodnú a modifikovanú účelovú funkciu.

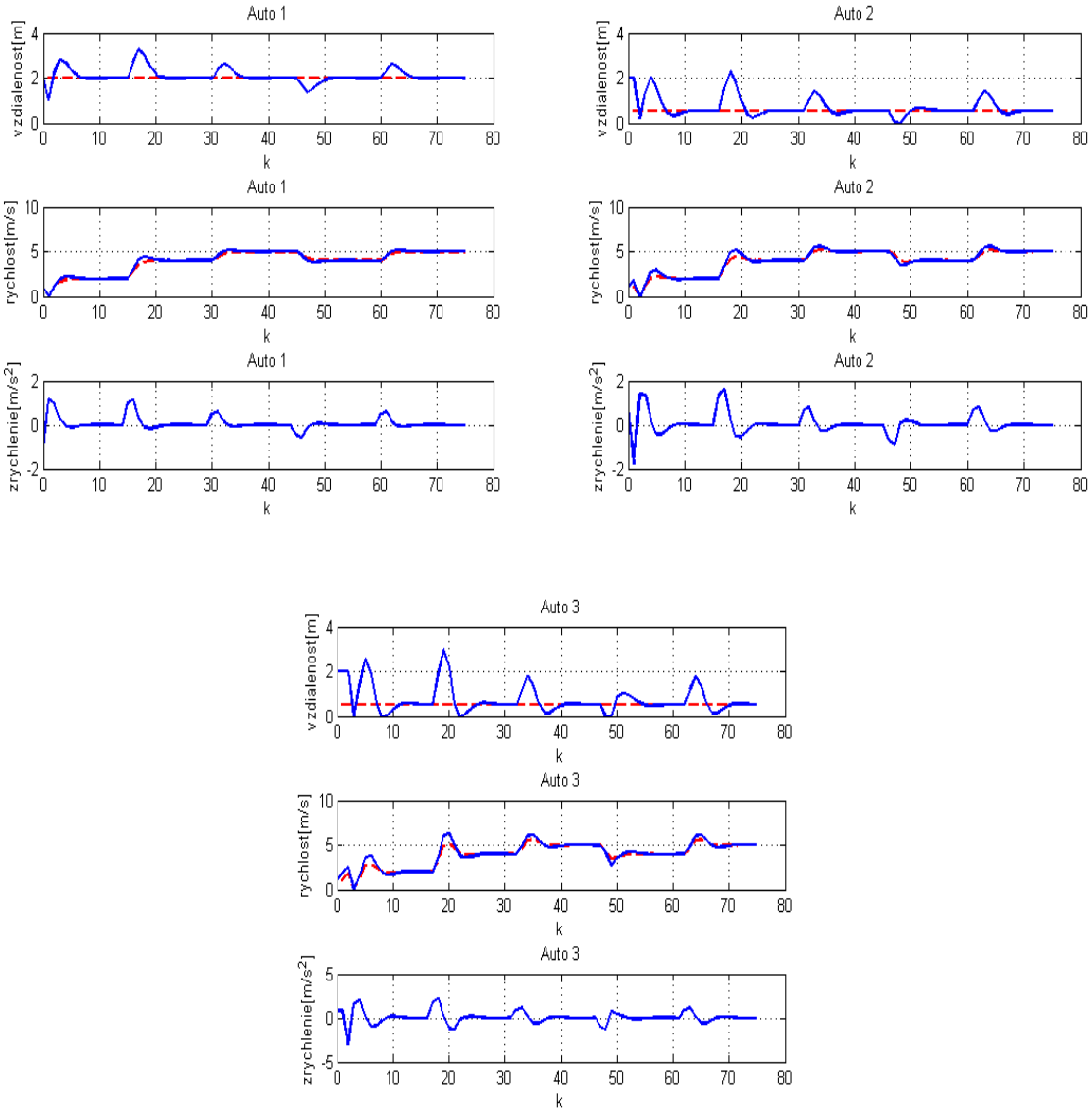
4.7 Scenár 5

V tomto prípade sme sa snažili simulovať riadenie troch áut, pričom rozostupy medzi nimi mali byť podľa kap. 4.5 0,1336 z dĺžky vozidla a túto dĺžku sme predpokladali 4 metre. V tabuľke 4.7 sú zhrnuté ostatné údaje, ktoré sme použili pri tejto simulácii.

	Referenčná rýchlosť[$m.s^{-1}$]	Referenčná vzdialenosť[m]
Prvé auto	2, 4, 5, 4, 5	2
Ostatné autá	rýchlosť predchádzajúceho	0,1336*dĺžka vozidla(0,1336*4)
Počiatočná podmienka	1	2

Tab. 4.7 Vstupné údaje pre piatu simuláciu.

Keď sa pozrieme na žiadanú rýchlosť pre prvé auto, vidíme, že hodnoty sú pomerne nízke a menia sa len veľmi málo. Je to preto, že sme sa snažili nájsť taký simulačný scenár, aby sme dostali priebeh bez zrážky. Ako však ale môžeme vidieť na obr. 4.11, ani pri takýchto podmienkach sa nám to nepodarilo. Vidíme, že druhé ani tretie auto sa pri tak malých odstupoch nedokážu vyhnúť zrážke. Takýmto spôsobom sme odsimulovali viaceré rýchlostné profily, ale vždy neúspešne. Preto môžeme vysloviť záver, že v reálnej premávke nie je možné skupinu áut riadiť tak, aby udržiavali medzi sebou rozostup len 0,1336 z dĺžky vozidla. Teoreticky by sa dalo nad takýmto scenárom uvažovať, ale len v tom prípade, že by sme mohli autá riadiť centralizovane. Ak by mali autá vždy informácie o celej kolóne, mohli by v rovnakom čase používať tie isté akčné zásahy a tým pádom sledovať tie isté trajektórie rýchlosti. V našom prípade nastáva veľké skreslenie pôvodnej referenčnej trajektórie rýchlosti, keďže každé auto okrem prvého má za svoju žiadanú hodnotu rýchlosti rýchlosť auta idúceho pred ním. A keďže už prvé auto nesleduje referenciu úplne presne, ale s istou odchýlkou, takisto aj druhé, a teda tretie už úplne stráca informáciu o tom, čo sa deje vpredu. Môžeme si to všimnúť, keď sa pozrieme na obr. 4.11 a porovnáme rýchlostné profily prvého a tretieho auta. Sú dosť odlišné na prvý pohľad, preto nie je možné riadiť kolónu, v ktorej idú autá tak tesne za sebou.



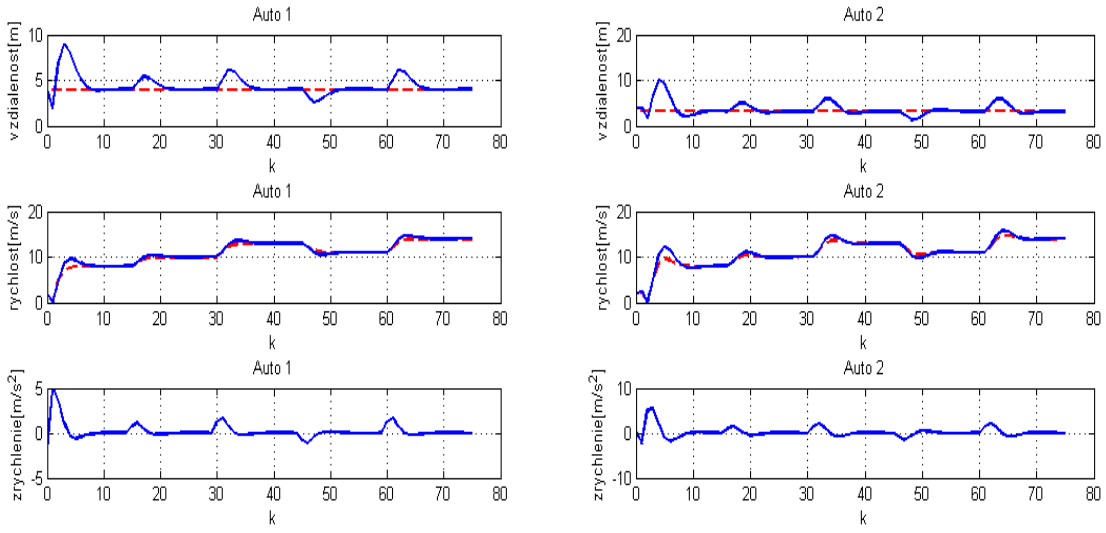
Obr. 4.11 Grafické výsledky piatej simulácie s rozostupmi 0,1336 dĺžky auta.

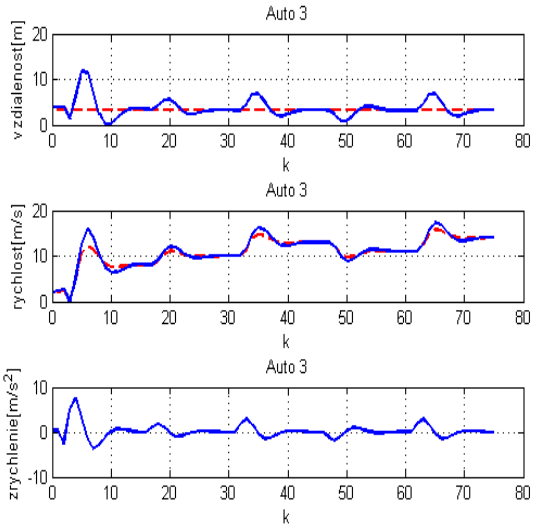
Podľa výskumu [2], ktorý bol zameraný na redukciu aerodynamického odporu a spotreby paliva pri jazde troch áut v kolóne s odstupom 0,8 dĺžky vozidla, je v tomto prípade priemerná úspora paliva 6-7 %. Preto sme ďalšiu pozornosť venovali simuláciám rôznych situácií, pri ktorých by bolo možné kolónu riadiť na takéto rozostupy. V nasledujúcej tabuľke sú zhrnuté údaje, pre simuláciu s rozostupmi 0,8 dĺžky auta, pri ktorých bolo ešte možné kolónu riadiť tak, aby sa autá nezrazili.

	Referenčná rýchlosť[m.s^{-1}]	Referenčná vzdialenosť[m]
Prvé auto	8, 10, 13, 11, 14	4
Ostatné autá	rýchlosť predchádzajúceho	0,8*dĺžka vozidla(0,8*4)
Počiatočná podmienka	2	4

Tab. 4.8 Vstupné údaje pre piatu simuláciu, rozostupy 0,8 dĺžky vozidla.

Na obr. 4.12 sú znázornené priebehy tejto simulácie. Máme možnosť vidieť, že dosiahnuť spomínanú úsporu paliva, a teda dodržiavať rozostupy 0,8, je bez kolízie možné pri zmenách rýchlosti prvého vozidla o cca 2-3 m.s^{-1}.





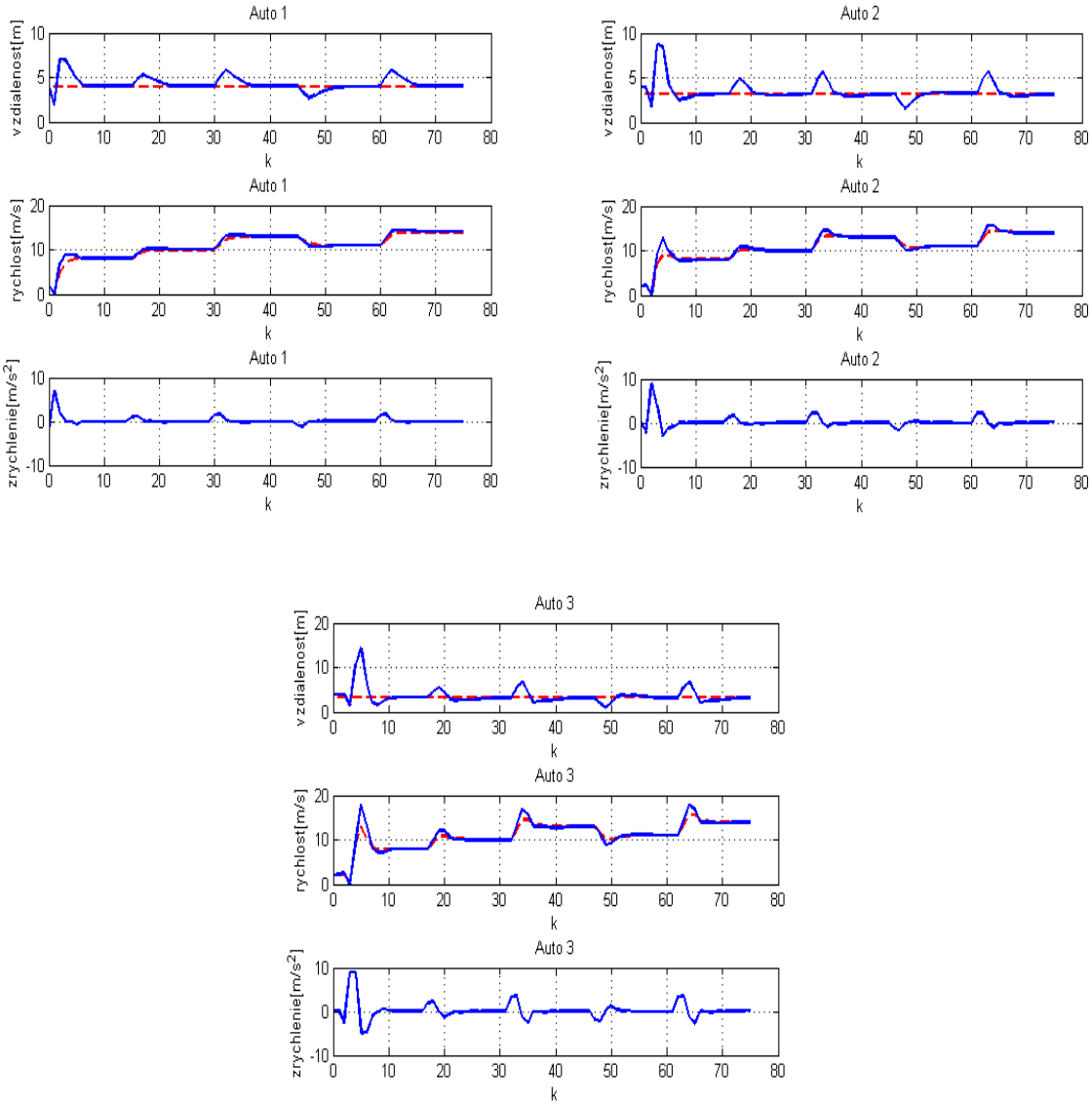
Obr. 4.12 Grafické výsledky piatej simulácie s rozostupmi 0,8 dĺžky auta.

Podobne ako tomu bolo v kap. 4.7, aj tu sme vykonali simuláciu s modifikovanou účelovou funkciou, v ktorej nevystupoval riadiaci vstup *u*, ale nová premenná *E*. Použitím takéhoto kritéria sme opäť penalizovali len kladné akčné zásahy. Porovnanie súm kladných *u* a *IAE* pre simuláciu s pôvodnou a modifikovanou účelovou funkciou je zobrazené v tabuľke 4.9.

	Suma kladných u[m.s^{-2}]	IAE
Pôvodná účelová funkcia	87,11	149,38
Nová účelová funkcia	84,53	119,96

Tab. 4.9 Porovnanie kritérií riadenia pre pôvodnú a modifikovanú účelovú funkciu.

Znova máme možnosť vidieť, že suma kladných akčných zásahov zavedením novej premennej *E* mierne klesla(cca 3%). Pritom sme dosiahli aj zníženie absolútnej hodnoty regulačnej odchýlky. Ukázalo sa teda, že zavedenie kritéria (4.3) pri takýchto úlohách je vhodné a malo by nám priniesť zníženie spotreby paliva pri súčasnom zlepšení regulačných priebehov. Na obr. 4.13 sú grafické výsledky pre tento scenár s použitím účelovej funkcie (4.3).

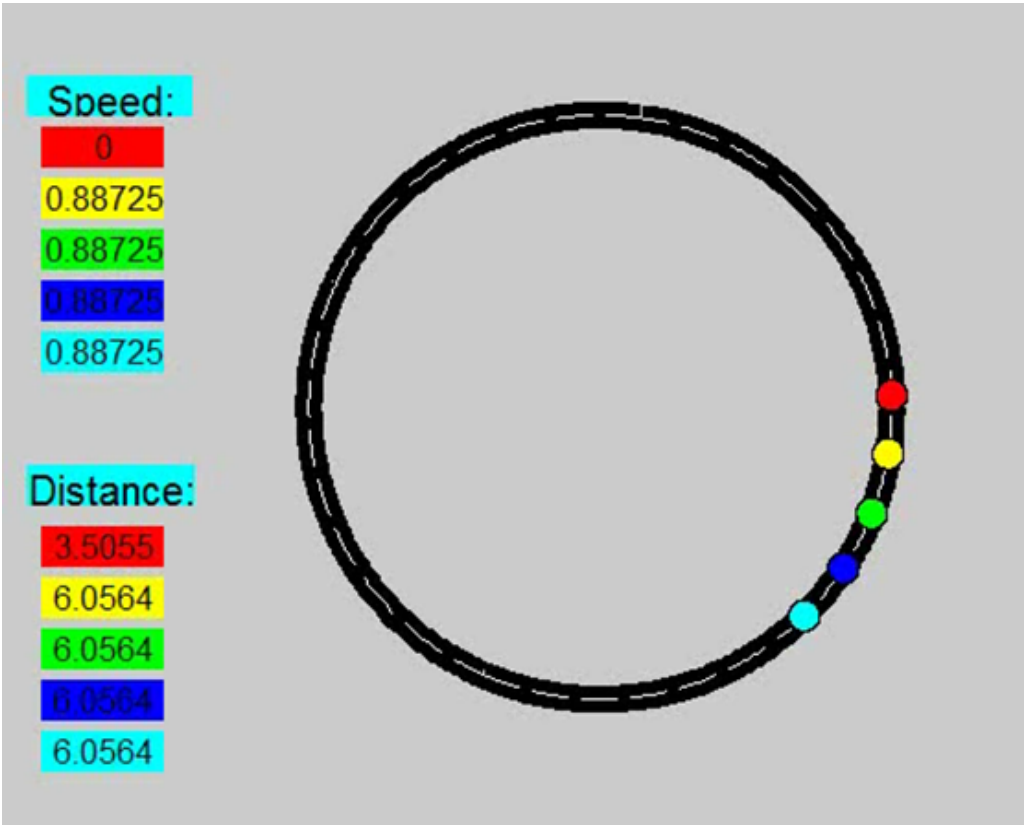


Obr. 4.13 Grafické výsledky piatej simulácie s rozostupmi 0,8 dĺžky auta a zavedením E.

4.8 Vytvorenie animácie

Aby bolo možné názornejšie posúdiť dosiahnuté simulačné výsledky, ďalším cieľom bolo vytvorenie animácie. Na tento účel bol vytvorený ďalší m-súbor s názvom *moveComplete.m*, ktorý spracováva údaje získané optimalizáciou. Pre spustenie tohto súboru je nutné najskôr spustiť *car_plotn.m*. Po spustení tohto súboru sa zobrazí okno, ktorého vzhľad je možné vidieť na obr. 4.14. Automobily sú reprezentované farebnými krúžkami, ktoré sa pohybujú po kružnici.

V ľavej časti okna sú zobrazované aktuálne rýchlosti každého auta a jednotlivé vzdialenosti medzi nimi, pričom jednotlivé textové okná sú príslušne farebne rozlíšené, aby zodpovedali každému automobilu (farebnému krúžku).



Obr. 4.14 Vzhľad grafického okna pri priebehu animácie.

V nasledujúcej časti si stručne opíšeme fungovanie takejto animácie.

```
F=figure;
col={[1 0 0],[1 1 0],[0 1 0],[0 0 1],[0 1 1],[1 0 1],[0 0 0],[1 1 1]};
dist_init=x0(1);
v=cell(ncars);
alfa=cell(ncars);
TextBox_speed=cell(ncars+1);
TextBox_distance=cell(ncars+1);
```

V úvodnej časti sa vytvorí prázdny obrázok a definujú sa niektoré premenné. Vektor `col` je v ďalšej časti použitý ako zásobník RGB farieb pre jednotlivé autá, `dist_init` je počiatočná vzdialenosť medzi autami.

```
for i=1:ncars
    v{i}=cumsum(X{i}(:,2));
    c=v{1}(end);
    r=c/(2*pi);
    r_car=r/20;
    alfa{i}=((v{i}-(i-1)*(2*r_car+dist_init))*2*pi)/c;

    TextBox_speed{i} = uicontrol('style','text');
    set(TextBox_speed{i}, 'Position',  get(TextBox_speed{i},...
        'Position')+[8, 315-(i-1)*25, 0, 0], 'BackgroundColor',col{i});
    TextBox_distance{i} = uicontrol('style','text');
    set(TextBox_distance{i}, 'Position',  get(TextBox_distance{i},...
        'Position')+[8, 315-((ncars+i)*25+5), 0, 0],...
        'BackgroundColor',col{i});

end
```

V tejto časti zdrojového kódu vidíme cyklus, ktorý sa opakuje `ncars`-krát, kde `ncars` predstavuje počet áut, ktoré sme zadali v súbore *car_plotn.m*. Jednotlivé polia `v{i}`, vzniknú ako tzv. kumulované sumy druhého stĺpca stavového vektora `X{i}(:,2)`, na tomto mieste sa nachádzajú aktuálne rýchlosti áut. Keď uvažujeme periódu vzorkovania rovnú 1 sekunde, vektory `v{i}` v sebe obsahujú informáciu o celkovej prejdenej dráhe v každom kroku simulácie. Premenná `c` je potom rovná dráhe, ktorú prešlo prvé auto. Keď uvažujeme, že sa autá pohybujú po kružnici, `c` bude jej obvod a `r` polomer. Autá sa v animácii zobrazujú ako krúžky s polomerom `r_car`. Dôležité premenné sú vypočítané vo vektoroch `alfa{i}`, pretože ide o uhol kruhového pohybu, o ktorý sa každé auto v danom kroku posunie. Tento uhol nám v ďalšej časti slúži na vypočítavanie súradníc *x* a *y*, po ktorých sa autá v priebehu animácie pohybujú.

Druhá polovica ukážky zdrojového kódu slúži na vytvorenie dvoch textových polí pre každé auto, a to `TextBox_speed{i}` a `TextBox_distance{i}`.Súradnice sú vypočítané tak, aby sa v ľavej časti okna animácie zobrazovali pod sebou rýchlosti (`TextBox_speed{i}`) pre všetky autá a pod nimi jednotlivé vzdialenosti (`TextBox_distance{i}`) medzi nimi.

```
TextBox_speed{ncars+1} = uicontrol('style','text');
set(TextBox_speed{ncars+1}, 'Position',  get(TextBox_speed{ncars+1},...
    'Position')+[0, 340, 22, 0],'BackgroundColor','c');
set(TextBox_speed{ncars+1},'string','Speed:','FontSize',15);
TextBox_distance{ncars+1} = uicontrol('style','text');
set(TextBox_distance{ncars+1}, 'Position',...
    get(TextBox_distance{ncars+1}, 'Position')+...
    [0, 315-(ncars*25+5), 22, 0],'BackgroundColor','c');
set(TextBox_distance{ncars+1},'string','Distance:','FontSize',15);
```

Vyššie uvedený kód slúži len na označenie rýchlostí ako „Speed“ a vzdialeností medzi autami ako „Distance“. Umiestnenie týchto prvkov je možné vidieť na obr. 4.14.

```
circle=0:0.01:2*pi;
circleX=r+r*cos(circle);
circleY=0+r*sin(circle);
plot(circleX, circleY,'k','LineWidth',10)
hold on
plot(circleX, circleY,'w--','LineWidth',1)
```

Tu pomocou príkazu `plot` vykresľujeme kružnicovú dráhu s polomerom `r`. V animácii je táto dráha zobrazená ako pomyselná cesta, po ktorej jazdia jednotlivé autá za sebou v kolóne.

```
axis([-20 2*r+10 -(r+15) r+15]);
axis square;
axis off;
```

Príkaz `axis` slúži na definovanie a škálovanie osí. Najskôr nastavíme minimálne a maximálne hodnoty pre osi x a y, potom pomocou `axis square` nastavíme rovnakú dĺžku a veľkosť kroku. Keďže v priebehu animácie nie je potrebné osi zobrazovať, použijeme `axis off`.

```
car_init=0:0.01:2*pi;
car_initX=r+r_car*cos(car_init);
car_initY=0+r_car*sin(car_init);
h=cell(ncars);

for i=1:ncars
    h{i}=patch(car_initX,car_initY,col{i});
end
```

V tejto časti už vytvoríme jednotlivé autá (krúžky) s polomerom `r_car`. Vektory `car_initX` a `car_initY` obsahujú súradnice bodov, ktoré tvoria krúžky. Príkaz `patch` pospája jednotlivé body do mnohostenu, v našom prípade vytvorí krúžky, pričom každému je pridelená jedna farba z vektora `col{i}`.

```
carX=cell(ncars);
carY=cell(ncars);
for j=1:length(X{1})

    for i=1:ncars
        set(TextBox_speed{i},'string',num2str(X{i}(j,2)),'FontSize',12);
        set(TextBox_distance{i},'string',num2str(X{i}(j,1)),...
            'FontSize',12);

        if j==length(X{1})
            set(TextBox_speed{i},'string','0')
        end

        carX{i}=car_initX+r*cos(alfa{i}(j));
        carY{i}=car_initY+r*sin(alfa{i}(j));
        set(h{i},'XData',carX{i});
        set(h{i},'YData',carY{i});
    end

    pause(0.2);
end
```

Vo vyššie uvedenej ukážke kódu môžeme vidieť dva vnorené cykly, ktoré už zabezpečujú pohyb áut. Vonkajší cyklus sa opakuje až kým premenná `j` nedosiahne hodnotu `length(X{1})`, čo zodpovedá počtu krokov simulácie. Vnútorný cyklus sa opakuje pre každé auto. Pomocou príkazu `set` tu nastavujeme hodnoty aktuálnych rýchlostí a vzdialeností v textových oknách. Tu si môžeme všimnúť, že číselné hodnoty uložené v jednotlivých stavoch boli prevedené na dátový typ *string* (napr. `num2str(X{i}(j,2))`), aby bolo možné ich zobraziť v animácii.

Ďalej sú v každom kroku vypočítavané nové súradnice `carX{i}` a `carY{i}` pre každý automobil. Potom sa tieto súradnice nastavujú jednotlivým autám `h{i}`, a tým, že sa autá prekresľujú v každom kroku, dosiahli sme ich pohyb po kružnici.

Záver

V závere môžeme konštatovať splnenie vytýčených cieľov. Diplomová práca obsahuje v úvodných kapitolách stručný teoretický prehľad základných aspektov prediktívneho riadenia, na ktorých boli postavené praktické simulačné úlohy. Pre potreby experimentálnej časti bola zostavená matematické formulácia úlohy riadenia automobilových skupín. Táto formulácia bola potom implementovaná v Matlabe. Výraznú pomoc v podobe možnosti jednoduchého a intuitívneho zápisu optimalizačných úloh nám v tomto prípade poskytol toolbox Yalmip.

V experimentálnej časti bolo uvažovaných niekoľko simulačných scenárov, ktoré by v reálnom živote mohli nastať. Takisto bolo skúmaných niekoľko možností, ktorými by bolo možné znížiť prevádzkové náklady vozidiel v bežnej premávke a boli simulované niektoré hraničné možnosti súvisiace so znížením spotreby paliva.

Keď zhrnieme dosiahnuté výsledky, môžeme konštatovať potenciálnu možnosť zavedenia niektorých foriem MPC použitých v tejto práci do skutočnej premávky. Na druhej strane je potrebné aj povedať, že takýto spôsob riadenia, kde majú vozidlá informáciu len o stavoch auta idúceho v kolóne pred nimi, má svoje nedostatky. Tie sa prejavujú najmä pri väčšom počte áut v kolóne, prípadne ak chceme medzi autami dodržiavať malé rozostupy. Napriek týmto nedostatkom sme však v práci poukázali na niektoré možnosti použitia takéhoto prístupu. Bezpochyby lepšou alternatívou by bolo centralizované riadenie, pri ktorom by boli riadené všetky autá naraz a na základe informácií o stavoch každého z nich. To si však momentálne vieme len ťažko predstaviť, pretože rovnaký riadiaci systém by musel byť zdieľaný každým vozidlom jazdiacim v kolóne.

V závere práce je ešte vysvetlený spôsob, akým bola vytvorená jednoduchá animácia slúžiaca na doplnenie grafických výsledkov. Pre každý simulačný scenár nájdeme na priloženom CD krátke videá vo formáte *.avi, ktoré slúžia na lepšiu interpretáciu vykonaných simulácií.

Zoznam použitej literatúry

[1] Boyd, S. – Vandenberghe, L.: *Convex Optimization*, ISBN 0 521 83378 7, 2004.

[2] Browand, F.: *Reducing Aerodynamic Drag and Fuel Consumption*, Advanced Transportation Workshop, October 10-11, 2005.

[3] Karas, A. – Rohaľ-Ilkiv, B.: *Praktické aspekty prediktívneho riadenia*, STU Bratislava, 2007, ISBN 978-80-89316-06-9.

[4] Marcu, B. – Browand, F.: *The Aerodynamic Forces on Misaligned Platoons*, California PATH Research Report UCB-ITS-PRR-98-4, Berkely, California, 1998.

[5] Rossiter, J.A.: *Model-based predictive control: a practical approach*, ISBN 0-8493-1291-4, Taylor & Francis e-Library, 2005.

[6] Zabat, M. – Stabile, N – Frascaroli, S. – Browand, F.: *Drag Forces Experienced by 2, 3 & 4 Vehicle Platoons at Close – Spacings*, SAE paper No. 950632, 1995.

[7] http://en.wikipedia.org/wiki/File:MPC_scheme_basic.svg

[8] http://sk.wikipedia.org/wiki/Odporov%C3%A1_sila