Upravljanje gibanjem trkaćeg bolida koristeći modelsko prediktivno upravljanje

Lovreković, Matej

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 64

UPRAVLJANJE GIBANJEM TRKAĆEG BOLIDA KORISTEĆI MODELSKO PREDIKTIVNO UPRAVLJANJE

Matej Lovreković

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 64

UPRAVLJANJE GIBANJEM TRKAĆEG BOLIDA KORISTEĆI MODELSKO PREDIKTIVNO UPRAVLJANJE

Matej Lovreković

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 64

Pristupnik:	**Matej Lovreković (0036526856)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	akademik prof. dr. sc. Ivan Petrović

Zadatak: **Upravljanje gibanjem trkaćeg bolida koristeći modelsko prediktivno upravljanje**

Opis zadatka:

Cilj je ovog diplomskoga rada implementirati upravljanje gibanjem autonomnog mobilnog vozila u obliku trkaćeg bolida s pogonom na stražnjim kotačima. U radu je potrebno implementirati modelsko prediktivno upravljanje za slijeđenje zadane putanje vozila kroz stazu pri čemu je za validaciju potrebno izvesti analitičku linearizaciju vozila koja bi osigurala da je model uvijek lineariziran oko trenutne radne točke. U model vozila potrebno je uključiti i kašnjenje u sustavu koje je uzrokovano komunikacijom i procesiranjem podataka. Modeliranjem i upravljanjem aktuatora potrebno je ostvariti izravnu kontrolu pozicije ili momenta na aktuatoru za skretanje i kočenje te izravnu kontrolu pogonskih motora. Metoda će biti implementirana unutar Matlab programskog paketa te validirana simulacijom.

Rok za predaju rada: 28. lipnja 2024.

Prvenstveno se zahvaljujem roditeljima i obitelji za veliku podršku tijekom studiranja. Zahvaljujem mentoru, prof. dr. sc. Ivanu Petroviću te asistentima sa studija za pomoć pri defniranju i izvršavanju zadatka.

Također se zahvaljujem kolegama iz FSB Racing Teama na izdvojenom vremenu i savjetima koji su pomogli boljem razumijevanju platforme za koju je namijenjen produkt ovog diplomskog rada.

Sadržaj

1. Uvod 3

2. Stavke definirane pravilnikom 5

2.1. Izgled staze 6

2.2. Komponente vozila 7

3. Opis vozila 9

3.1. Procesne jedinice i komunikacijska struktura 9

3.2. Pogon, skretanje i kočenje 12

3.3. Senzori 16

4. Modelsko prediktivno upravljanje 18

4.1. Matematička pozadina 20

4.2. Princip rada 25

4.3. Tipovi MPC-a 28

4.4. Optimizacijski algoritam 30

4.5. Odabir platforme 34

5. Model vozila 37

5.1. Jednadžbe i varijable stanja 38

5.2. Linearizacija modela 53

5.3. Diskretizacija modela 59

5.4. Testiranje modela 63

6. Implementacija i simulacija adaptivnog MPC-a 71

6.1. Implementacija 72

6.2. Rezultati 79

6.2.1. Simulacija 1 - Agresivno upravljanje . . . 79
6.2.2. Simulacija 2 - Blago upravljanje . . . 81
6.2.3. Simulacija 3 - Rub stabilnosti . . . 82
6.2.4. Nedostatak Adaptivnog MPC-a . . . 84

7. Implementacija i simulacija LTV MPC-a . . . 86
7.1. Implementacija . . . 86
7.1.1. Optimalna trajektorija . . . 88
7.1.2. Analiza šuma . . . 89
7.2. Rezultati . . . 90
7.2.1. Simulacija 4 - Idealno praćenje reference . . . 90
7.2.2. Simulacija 5 - Aktivacija ograničenja . . . 92
7.2.3. Simulacija 6 - Analiza šuma i kašnjenja . . . 95
7.2.4. Simulacija 7 - Optimalna trajektorija . . . 99
7.2.5. Usporedba algoritama optimizacije . . . 101
7.2.6. Ograničenja na ulaze . . . 105

8. Rasprava . . . 107

9. Zaključak . . . 112

Literatura . . . 113

Sažetak . . . 115

Abstract . . . 116

1. Uvod

Moderna vozila sve više koriste sustave koji pomažu vozačima pri vožnji. Gotovo sva vozila koriste elektroničku kontrolnu jedinicu (*engl. Electronic control unit*, ECU) kao centralno računalo. Napretkom upravljačkih algoritama te dostupnošću raznih vrsta senzora te računala za vozila, sve se više pojavljuje autonomna vožnja.

U trendu sa tehnologijom, studentsko natjecanje u inženjerstvu i izradi bolida *Formula Student* je 2017. godine službeno predstavilo novi tip natjecanja - autonomna vožnja. Studenti diljem svijeta natječu se na godišnjoj razini u dizajniranju, konstrukciji te izradi bolida. Natjecanje se dijeli u 4 kategorije:

- Vozila na pogon s unutarnjim izgaranjem,
- Vozila na električni pogon,
- Hibridna vozila (električni pogon i pogon s unutarnjim izgaranjem),
- Autonomna vozila.

Bodovanje na natjecanjima dijeli se prema statičkim ili dinamičkim disciplinama. U statičkim disciplinama vrednuje se razumijevanje inženjeringa vozila, procesa izrade komponenti, upravljanje financijama te strukturom tima. Na dinamičkim disciplinama boduju se same performanse vozila na različitim tipovima staze.

U Europi postoji 70-ak timova s registriranim autonomnim bolidom [1]. Autonomna platforma može se integrirati na električna vozila ili vozila s unutarnjim izgaranjem. Pri počecima autonomnih vozila u *Formula Student* svijetu, predstavljena je disciplina pod imenom *Driverless Cup* u kojoj se boduju statičke i dinamičke discipline vezane samo uz autonomni bolid. U današnjici sve manje natjecanja održavaju *Driverless Cup* te se

bodovi autonomnog vozila izravno spajaju s bodovanjem baznog vozila (električnog ili vozila na unutarnje izgaranje).

Svaki tim koji želi sudjelovati na natjecanju mora se strogo držati pravila definiranih pravilnikom krovne organizacije [2]. Pravila definiraju sadržaj od općenite strukture tima do samog detalja u izradi bolida. Ovaj diplomski rad piše se u sklopu *Formula Student* tima naziva *FSB Racing Team* koji predstavlja Sveučilište u Zagrebu.

Struktura autonomnog procesa je vrlo složena, ali može se podijeliti u osnovne dijelove koji se izvršavaju slijedno:

1. Percepcija,
2. Lokalizacija i mapiranje (*engl. Simultaneous localization and mapping*, SLAM),
3. Planiranje putanje,
4. Upravljanje i aktuacija.

Percepcija se odnosi na prikupljanje podataka u okolini vozila. Nakon toga, pomoću prikupljenih podataka vozilo se lokalizira u prostoru te gradi mapu prostora oko sebe. Koristeći izgrađenu mapu planira se putanja kroz prostor kojom bolid treba odvoziti. Na samom kraju slijede upravljački algoritmi koji pokreću upravljive uređaje vozila da bi vozilo prošlo isplaniranu putanju u zadanom vremenu. Opcionalan korak je i optimizacija trajektorije u kojoj se nakon izgradnje cijele mape optimizira trajektorija po vremenu tako da se idući krugovi odvoze brže.

Tema diplomskog rada temelji se na posljednjem koraku autonomnih procesa - upravljanje i aktuacija. Za algoritam upravljanja koristit će se modelsko prediktivno upravljanje (*engl. Model predictive control*, MPC). Kao što ime algoritma upućuje, za takvo upravljanje potreban je model objekta kojim se upravlja, u ovom slučaju vozila. Za modeliranje vozila i što realnijih simulacija potrebno je znati što više informacija o vozilu i okolini kojom se vozilo kreće, stoga će u nastavku detaljno biti objašnjeni aspekti samog vozila te karakteristika staze kojom se vozilo kreće. U konačnici, performanse razvijenog MPC-a testirane su unutar *Simulink* okruženja uz što realnije uvjete.

2. Stavke definirane pravilnikom

Korisno je znati nekoliko općenitih pravila vezana uz vozilo. Vozilo mora biti jednosjed što znači da može sadržavati samo jednu osobu - vozača. Širina i dužina vozila je propisana kao i minimalna udaljenost od poda. Šasija ima sigurnosne strukture koje u slučaju sudara ili prevrtanja osiguravaju vozača od ozbiljnih ozljeda. Visokonaponski dijelovi auta moraju biti jasno označeni i samo licencirana osoba unutar tima (*engl. Electrical System Officer*, ESO) smije raditi na električnim dijelovima auta za vrijeme natjecanja.

Za vrijeme autonomne vožnje niti jedna osoba ne smije sjediti u bolidu te za autonomno vozilo postoji odgovorna osoba (*engl. Autonomous System Responsible*, ASR) koja u bilo kojem trenutku vožnje može ugasiti i sigurno zaustaviti vozilo. Bitno pravilo je da bolid ne smije posjedovati mogućnost kretanja u natrag, što znači da mu brzina kretanja smije biti samo pozitivna. Staze kojima se vozi, bio to vozač ili autonomno vozilo, označene su čunjevima.

Slika 2.1. Autonomni bolid na službenom natjecanju.

Na prethodnoj slici (slika 2.1.) prikazano je autonomno vozilo koje vozi kroz stazu definiranu pravilnikom. U nastavku rada razmatrati će se pravila vezana samo uz auto-

nomni bolid.

2.1. Izgled staze

Za autonomna vozila lijevi rub staze označen je plavim čunjevima, dok je desni rub staze označen žutim čunjevima. Ovakva raspodjela čunjeva olakšava percepciju i planiranje putanje.

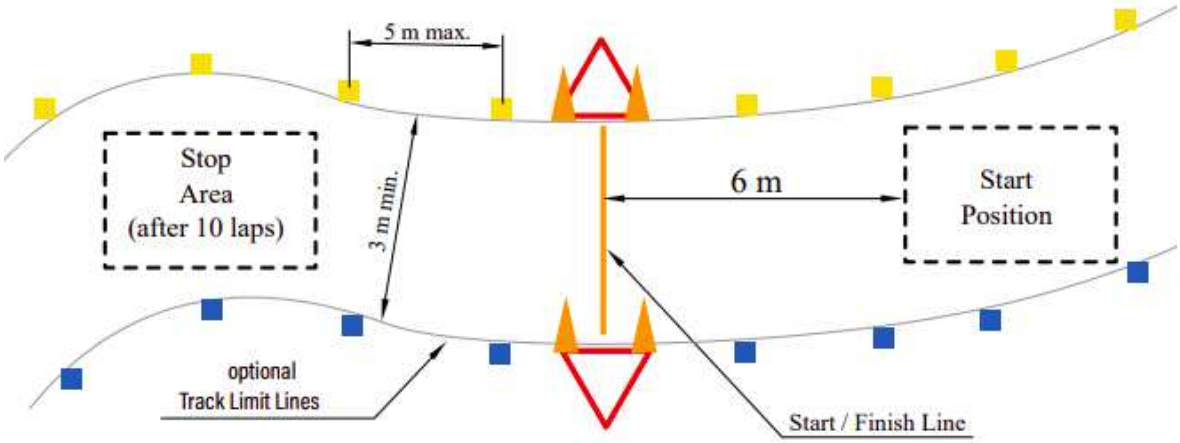
Različiti tipovi staza testiraju vozilo i vozače na razne načine. Dinamičke discipline vezane uz autonomna vozila vrlo su slične onima koje se voze sa vozačem:

- *Acceleration*,
- *Skidpad*,
- *Autocross*,
- *Trackdrive*.

Acceleration je ravna staza duga 75 *m* na kojoj vozilo u što kraćem vremenu mora doći do cilja. Nakon cilja vozilo mora u idućih 75 *m* sigurno zakočiti. Ova disciplina testira kontrolu vozila pri velikim brzinama i akceleracijama. Staza za *Skidpad* disciplinu izgleda kao osmica te je cilj napraviti dva kruga u lijevo i dva kruga u desno od kojih se boduje samo drugi krug. Ovom stazom testira se lateralna kontrola i stabilnost vozila. *Autocross* disciplina sastavljena je od ravnica, uzastopnih zavoja i oštrih zavoja te je ukupna duljina staze od 200 *m* do 500 *m*. Vozi se samo jedan krug u kojem se testira percepcija, lokalizacija i kontrola vozila. *Trackdrive* disciplina je ista kao i *Autocross* disciplina osim toga što se vozi deset krugova. Za ovakvu disciplinu najviše utjecaja ima optimizacija trajektorije te kontrola pri visokim brzinama.

U svim disciplinama bodovanje se dodjeljuje prema najbržim vremenima. Postoje i penali koji se dodjeljuju nakon završetka discipline. Najznačajniji penali su za srušene čunjeve te izlazak van staze sa sva četiri kotača. Na disciplinama *Autocross* i *Trackdrive*, od ukupnog vremena oduzima se 2 *s* za svaki srušeni čunj te 10 *s* za svaki izlazak van staze. Za autonomna vozila takve pojave mogu biti kritične, jer jednom srušeni čunj se više ne vraća na mjesto te to može poremetiti algoritme autonomnog upravljanja. Ču-

njevi mogu završiti unutar staze, van staze ili zaglavljeni ispod vozila. Čunjevi srušeni tako da završe unutar staze su poprilično su opasni jer mogu poremetiti algoritam za planiranje putanje koji se temelji na čunjevima kao rubovima mape.



Slika 2.2. Opća pravila za izgled staze autonomnih disciplina.

Na slici 2.2. može se vidjeti da je staza minimalno široka 3 m, a maksimalna udaljenost između čunjeva u smjeru kretanja je 5 m. Početak i kraj staze označeni su sa četiri velika narančasta čunja. Definiran je i minimalni polumjer zavoja tako da vanjski rub staze u zavoju ima minimalni polumjer 9 m.

Da bi se spriječila vrlo spora vožnja na *trackdrive* disciplini, uvedeno je pravilo da minimalna prosječna brzina u prva tri kruga mora biti 2.5 m/s, a nakon toga svaki odvoženi krug ne smije imati manju prosječnu brzinu od 3.5 m/s. Zbog toga, opća je praksa među timovima da se prvi krug vozi niskom konstantnom brzinom te centralnom putanjom radi sigurnosti od rušenja čunjeva s ciljem da se napravi dobra karta staze. Ostalih 9 krugova voze se optimalnom putanjom te optimalnom brzinom za timove koji imaju implementiranu optimizaciju trajektorije. Vozilo treba brojati krugove i zaustaviti se nakon posljednjeg kruga.

2.2. Komponente vozila

Autonomno vozilo analizirano u radu je električno, stoga treba spomenuti neka pravila za električna vozila. Maksimalna snaga koja se smije vući iz visokonaponske baterije u bilo kojem trenutku je 80 kW. Naglo kočenje i zadavanje više od 5 kW snage električnim motorima u razdoblju duljem od pola sekunde prekida napajanje prema vozilu.

Za autonomne upravljačke algoritme to znači da se ne smije istovremeno naglo kočiti i zahtijevati pozitivni moment na pogonskim motorima.

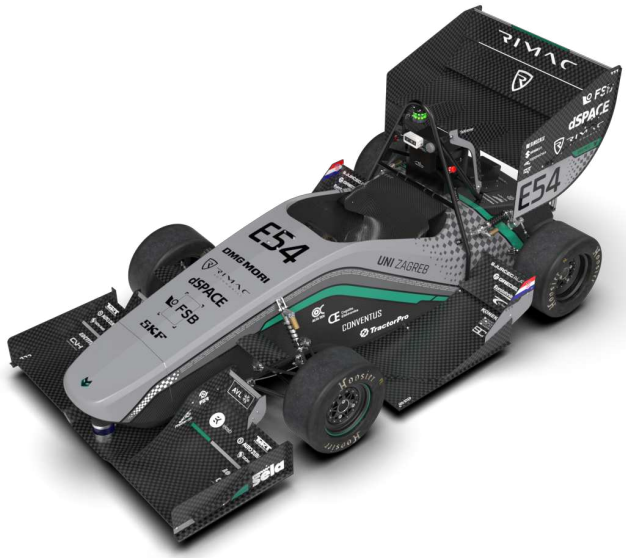
Autonomno vozilo mora imati i nekoliko sigurnosnih komponenti. Glavne sigurnosne komponente su EBS (*engl. Emergency Brake System*) i RES (*engl. Remote Emergency System*). EBS mora biti implementiran tako da vozilo krene naglo kočiti u slučaju gubitka napajanja. RES sustav sastoji se od prijamnika koji se nalazi u vozilu i odašiljača kojeg prilikom vožnje drži odgovorna osoba (ASR). Prilikom uočavanja nezgode ili nepovratnog izlaska sa staze, ASR mora aktivirati odašiljač prilikom čega prijamnik prekida napajanje prema vozilu. Prekidanjem napajanja EBS sustav se aktivira i sigurno zaustavlja vozilo.

Slika 2.3. RES sustav. Odašiljač na lijevoj strani i prijemnik na desnoj.

Za korištenje autonomnih algoritama ne postoje pravila niti restrikcije.

3. Opis vozila

Modelsko prediktivno upravljanje bit će implementirano na vozilu *FSB Racing Teama* službenog imena *FSB-RT06D* ili naziva VulpesD. Vozilo je prvi puta predstavljeno 2019. godine pod službenim imenom *FSB-RT06E* ili Vulpes kao prvo električno vozilo i prvo vozilo sa kompozitnom monokok šasijom. Kroz 2020. i 2021. godinu prolazilo je redizajn fazu pod imenom *FSB-RT06R* ili VulpesR te je konačno 2022. godine upotrijebljeno kao razvojna baza za autonomno vozilo.



Slika 3.1. Autonomni bolid VulpesD.

3.1. Procesne jedinice i komunikacijska struktura

Autonomni i upravljački algoritmi moraju se izvoditi na računalima. Unutar auta nalaze se 2 računala. Električna kontrolna jedinica *MicroAutoBox III* tvrtke *dSpace* te vlastito modificirano računalo sa specifikacijama:

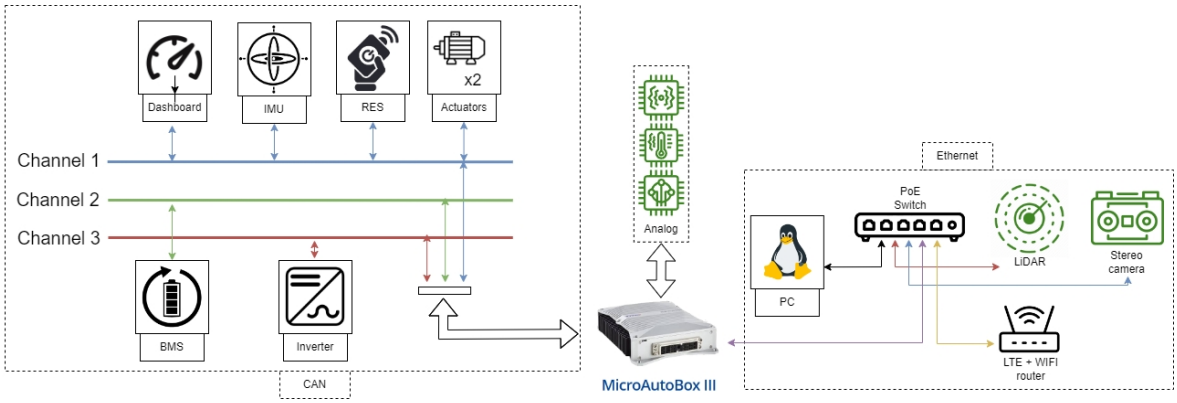
- Procesor: *AMD Ryzen 5 5600X*,
- Grafička kartica: *Nvidia Quadro RTX 4000*,
- Radna memorija: 16 GB 3200 MHz,
- Matična ploča: *ASRock B550M-ITX/ac*,
- SSD (engl. *Solid State Disk*): 512 GB M.2.

Slika 3.2. Vlastito modificirano računalo (lijevo) i *MicroAutoBox III* (desno).

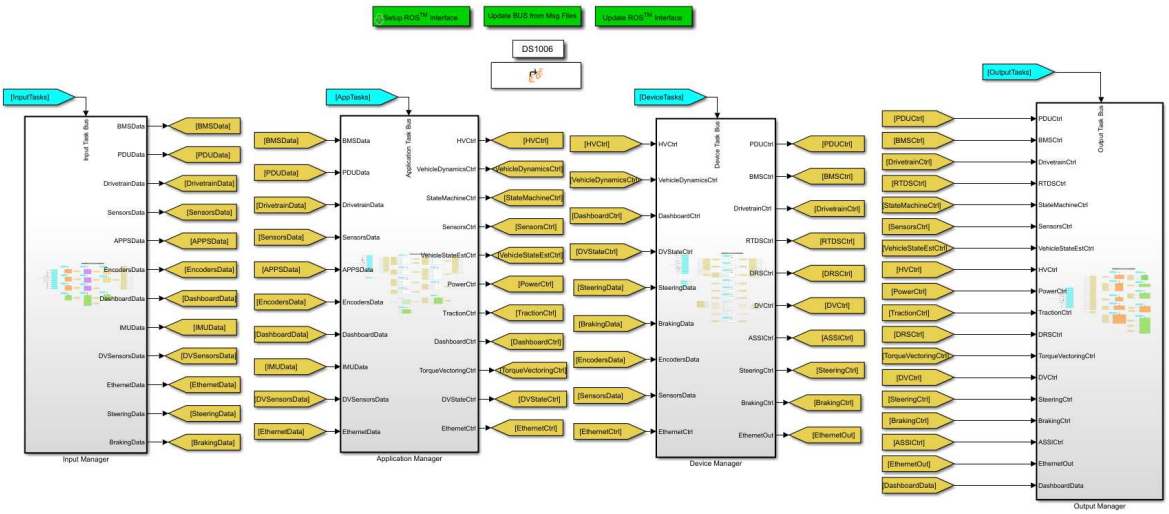
MicroAutoBox III ima četverojezgreni procesor koji radi na 1.4 GHz te ulazno/izlazne jedinice što ga čini centralnim računalom u vozilu. Kroz CAN (*engl. controller area network*) komunikaciju izmjenjuje podatke sa svim komponentama i aktuatorima u vozilu, a sa digitalnim i analognim ulazima/izlazima povezan je na sve senzore unutar vozila. Za modificirano računalo koristit će se naziv autonomno računalo jer se na njemu izvode svi algoritmi autonomnog upravljanja. Autonomno računalo koristi *Ubuntu 20.04*, a ECU (*MicroAutoBox III*) izvršava C kod koji je generiran iz *Simulink* modela. Autonomno računalo i ECU komuniciraju *Ethernet* protokolom (slika 3.3.).

Simulink model izvodi se u stvarnom vremenu na ECU te ima vrijeme izvršavanja 0.1 ms. Cijeli model izvodi se na jednoj jezgri procesora. Glavni zadatak je podijeljen u logičke cjeline izvršavanja koje su prikazane na slici 3.4.

Svi signali i informacije prolaze kroz slojeve nazvane upraviteljima, redom s lijeva na desno. Upravitelj ulaza prikuplja te vrši predobradu ulaznih podataka poput očitanja senzora. Upravitelj aplikacija izvršava postojeće upravljačke algoritme, algoritme



Slika 3.3. Struktura komunikacije unutar vozila.



Slika 3.4. Struktura izvršavanja glavnog zadatka unutar ECU *Simulink* modela.

estimacije te upravljačke logike. Upravitelj uređaja filtrira korisne informacije za uređaje, dok konačno upravitelj izlaza šalje informacije prema svim spojenim uređajima navedenim komunikacijskim protokolima. Svaki upravitelj sastoji se od 10 podzadatka raspoređenih u smislene cjeline. Cijeli model izvršava se odjednom, no radi sigurnosti i sljednosti protoka podataka, ažuriranje podataka vrši se prema rasporedu. U jednome izvođenju modela, dozvoljeno je ažuriranje informacija samo jednoga podzadatka (od ukupno 40 podzadataka), dok se na ostalim podzadacima prosljeđuju stare informacije. Takav pristup stvara kašnjenja na način da očitanja određenog senzora, koja su prikupljena podzadatakom za očitanje mjerenja senzora, neće se gledati s vremenom uzorkovanja 0.1 *ms* nego 4 *ms*. Kašnjenje uzrokovano rasporedom podzadataka ne uzrokuje probleme pri vožnji vozila. Kašnjenja uzrokovana rasporedom izvršavanja zadataka uzet će se u obzir pri simulacijama u potpoglavlju 7.1.2.

Od aktivnih algoritama upravljanja vozila koje treba uzeti u obzir pri razvoju MPC-a

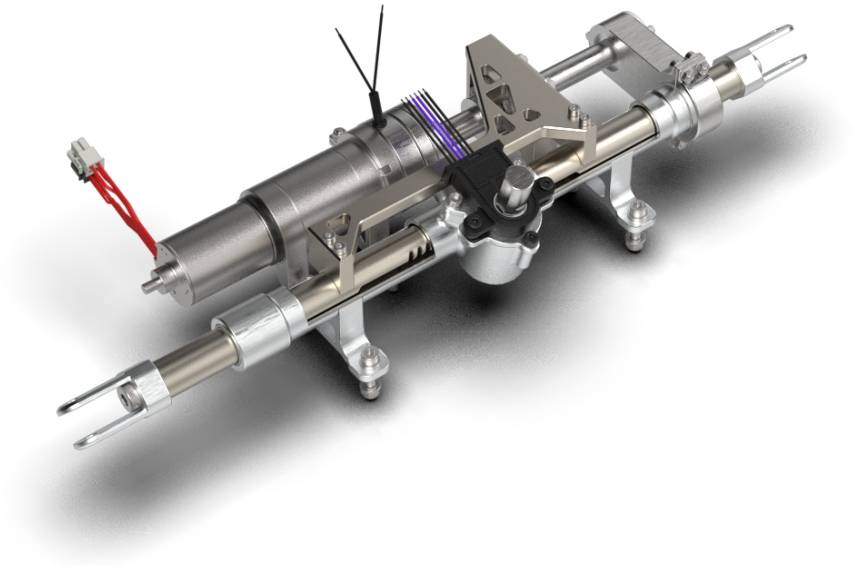
su kontrola proklizavanja (*engl. Traction control*) te vektoriranje momenta pogonskih motora (*eng. Torque vectoring*). Kontrola proklizavanja implementirana je PI regulatorom te oduzima vozaču zahtjev na moment pogonskih kotača ovisno o tome koliko kotači proklizavaju pri ubrzanju vozila. Algoritam vektoriranja momenta implementiran je jednostavnim regulatorom te raspodjeljuje moment na unutarnje i vanjske kotače prilikom skretanja. Unutar glavnog zadatka vozila postoji i mogućnost trajnog linearnog smanjenja momenta na pogonskim motorima izraženog u postotcima (*engl. APPS derating*). Smanjenje momenta često se primjenjuje radi štednje energije baterije pri vožnjama u kojima nije potrebna puna snaga pogonskih motora.

Svi autonomni algoritmi za percepciju, lokalizaciju, mapiranje te planiranje putanje razvijeni su unutar *FSB Racing Teama*. Algoritam koji je od posebne važnosti za MPC je optimizacija trajektorije. Algoritam optimizacije trajektorije preuzet je sa [3] i prilagođen vozilu VulpesD. Algoritam rješava zadatak koji je poznat kao problem minimalnog vremena kruga (*engl. minimum laptime problem*, MLTP). Ulaz u algoritam su podaci o vozilu te izgled konačne izgrađene mape. Izlaz algoritma su optimalna trajektorija, koja uključuje brzine te mnoge druge varijable stanja vozila koje su opisane u odjeljku 5.1. Varijable dobivene rješavanjem MLTP problema služe kao reference MPC algoritmu.

3.2. Pogon, skretanje i kočenje

VulpesD je električni bolid, stoga se za pogon koriste elektromotori. Specifično, 2 sinkrona motora sa trajnim magnetom tipa *Alta Motors TM-40* koji odvojeno kontroliraju stražnje kotače. Maksimalni moment koji mogu razviti je 60 Nm, a maksimalna brzina 14000 okretaja u minuti. Na kotače su spojeni preko prijenosa koja ima prijenosni omjer 8. Upravljački signal za upravljanje motorima je promil maksimalnog momenta u rasponu od 0 do 1000.

Autonomno skretanje ostvareno je korištenjem bezčetkastog istosmjernog motora te mehaničke konstrukcije koja preko navojnog vretena i vodilice prenosi okret motora u pomak letve volana. Direktnim aktuiranjem letve volana preko navojnog vretena gotovo ne postoji prazan hod aktuatora što uklanja nelinearnost uzrokovanu zračnošću. Dok na primjer, između letve volana i vratila volana postoji zračnost od nekoliko stupnjeva.



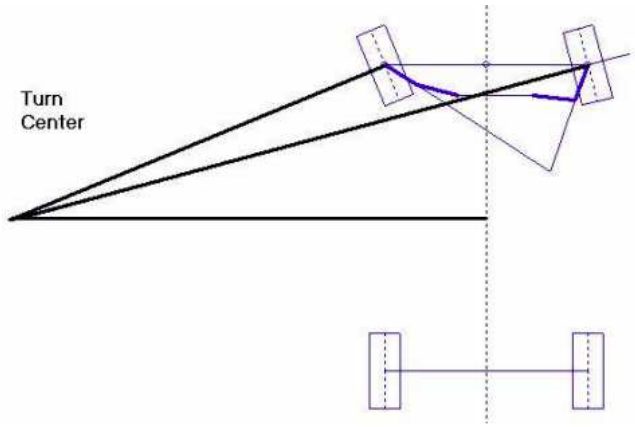
Slika 3.5. Aktuator, senzor za skretanje te mehanička konstrukcije za skretanje.

Odabrani aktuator je proizvod tvrtke *Maxon* te dolazi uz vlastiti mikrokontroler, prijenos te Hall senzor. Zbog navedenih komponenti motor se može upravljati na mnoge načine, od kojih su za vozilo bitni upravljanje pozicijom aktuatora te upravljanje momentom aktuatora.

Za autonomno skretanje očit je izbor upravljanje pozicijom motora, jer se pozicija motora preslikava na poziciju letve volana. Pozicija aktuatora za skretanje zadaje se u inkrementima, diskretnim pomacima motora koji su definirani preciznošću regulatora. Maksimalna vrijednot inkrementa je iznimno velika, stoga će se u nastavku rada koristiti stupnjevi aktuatora koji su linearno povezani sa inkrementima.

Važno je napomenuti da vozilo posjeduje Ackermann geometriju skretanja. Pri skretanju, simetrale kotača sijeku se u centru polumjera skretanja (slika 3.6.). To znači da zakretom volana u jednu stranu, kutevi zakreta vanjskog i unutarnjeg kotača nisu jednaki. Ackermann geometrija više je izražena što su kutevi skretanja oštriji. Također vrijedi napomenuti da su prednji kotači zakošeni prema van (*engl. toe-out*) za nekoliko stupnjeva, a stražnji kotači su za nekoliko stupnjeva zakošeni prema unutra (*engl. toe-in*).

Autonomno kočenje ostvareno je istim aktuatorom kao i skretanje. Sustav autonomnog kočenja sastoji se od koloture koja je montirana na vratilo aktuatora te sajle. Okretanjem aktuatora, sajla se namotava preko koloture te tako povlači pedalu kočnice. Ak-



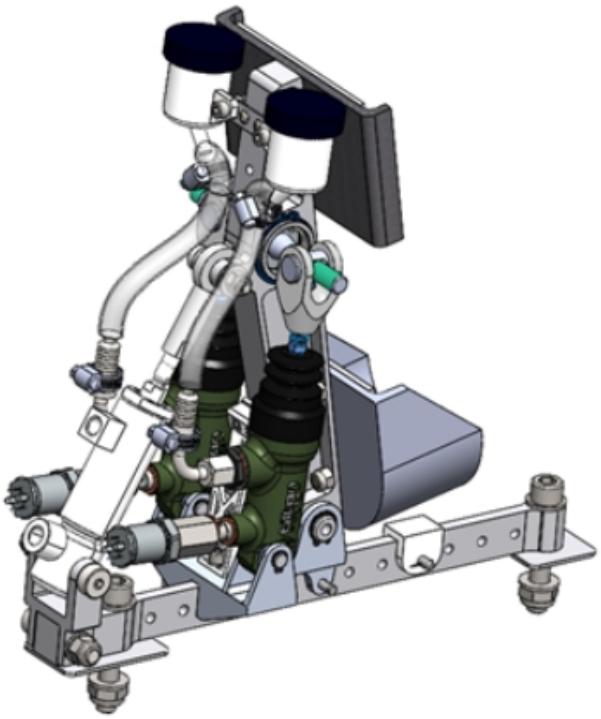
Slika 3.6. Ackermann geometrija skretanja.

tuacijom kočenja može se upravljati pozicijski kao i skretanje ili momentom, pri čemu se moment aktuatora zadaje u postotcima od 0 do 100 u odnosu na maksimalni kontinuirani moment motora.

Mehanički sustavi kočenja i skretanja konstruirani su tako da najbolje iskorištavaju karakteristike aktuatora prema potrebama vožnje.

EBS sustav spomenut u poglavlju 2. izveden je pomoću hidraulike, no potpuno je odvojen od sustava kočenja te stoga nije detaljno obrađen. Jedini način na koji EBS utječe na upravljanje je taj da je glavni cilindar EBS-a direktno spojen na pedalu kočnice (slika 3.8.). Zbog toga cilindar EBS-a stvara dodatni otpor pri kočenju te se moment aktuatora za kočenje treba dodatno trošiti na tlačenje cilindra što otežava matematičko modeliranje kočenja.

Slika 3.7. Sklop pedale kočnice te aktuatora za kočenje.



Slika 3.8. Sklop pedale kočnice te EBS cilindra (bijela boja).

3.3. Senzori

Autonomna vozila najviše se oslanjaju na informacije sa senzora. Stoga je bitno znati kojim senzorima vozilo raspolaže.

Od percepcijskih senzora izdvajaju se:

- LiDAR (*engl. Light Detection and Ranging*),
- Stereokamera.

Percepcijski senzori najviše se koriste za lokaliziranje u prostoru te planiranje putanje stoga nisu detaljno obrađeni.

Bitni propriocepcijski senzori koji se nalaze u vozilu su:

- Enkoder za mjerenje pomaka volana,
- Senzor tlaka u prednjem sustavu kočnica,
- Senzor tlaka u stražnjem sustavu kočnica,
- Vlastito razvijen sustav sa Hall senzorom za mjerenje brzine rotacije kotača na svim kotačima,
- Enkoder za brzinu rotacije i poziciju pogonskih motora,
- Inercijska mjerna jedinica (*eng. inertial measurement unit*, IMU) koja ima akcelerometar i žiroskop, procjenu brzine u svim osima, procjenu orijentacije šasije vozila te estimaciju pozicije pomoću GNSS-a (*engl. Global navigation satellite system*),
- Akcelerometar i žiroskop integriran u ECU,
- Mjerenje struja pogonskih motorima,
- Mjerenje razvijenog momenta, brzine rotacije te pozicije aktuatora,
- Mjerenje temperatura u aktuatorima, pogonskim motorima, te na mnogim drugim mjestima unutar vozila.

Neki senzori poput senzora tlaka i senzora pomaka volana imaju analogna očitanja veličina koja ulaze u ECU, što je podložno elektromagnetskim smetnjama unutar vozila. Šumovita okolina može rezultirati lošijim performansama algoritama. Šumovi će biti detaljno analizirani u potpoglavlju 7.1.2.

Svi nabrojani senzori daju dobar uvid u stanja i ponašanje vozila. No, uspoređivanjem drugih *Formula Student* timova te korištenih senzora i metoda, među stvarima koje VulpesD nema su:

- Senzor za mjerenje brzine gibanja vozila naspram tla (*engl. Ground Speed Sensor*, GSS),
- Dualni postav GPS (*engl. Global Positioning System*) antena (jedna na prednjem kraju vozila, druga na stražnjem) što pridonosi redundanciji i dodatnim mogućnostima estimacije bočnog kuta klizanja vozila.

4. Modelsko prediktivno upravljanje

Kao zadnji korak autonomnog procesa dolazi aktuacija i upravljanje. Algoritam planiranja putanje ili optimizacije trajektorije pronalazi putanju kojom se vozilo treba kretati u prostoru, a brzina slijeđenja putanje zadana je proizvoljnom logikom ili optimizatorom trajekorije. Glavni zadatak upravljanja vozilom je aktuirati upravljive komponente unutar vozila kako bi se vozilo provezlo zadanom putanjom uz zadanu brzinu.

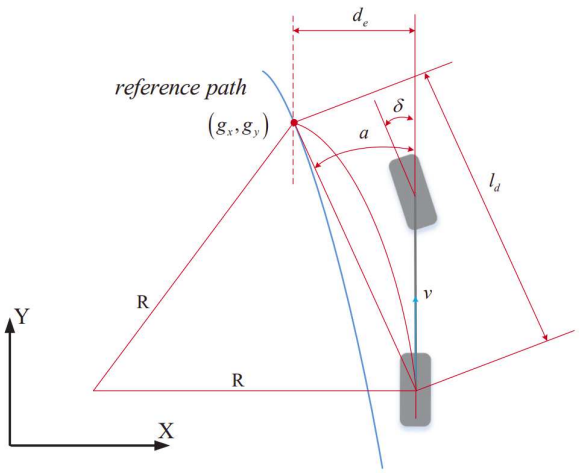
Algoritmi upravljanja moraju se izvršavati aktivno dok auto vozi u stvarnom vremenu (*engl. online*). Rad u stvarnom vremenu dodaje mnogo problema za upravljačke algoritme. Zbog komunikacija i procesiranja dolazi do kašnjenja mnogih signala. Vrijeme izvršavanja upravljačkih algoritama mora biti iznimno brzo inače dolazi i do kašnjenja upravljačkog signala. Posljedice kašnjenja signala i izvršavanja u stvarnom vremenu analizirane su kroz simulaciju u potpoglavlju 6.2.3.

Nekoliko općenitih algoritama upravljanja vozila te algoritama koje koriste drugi timovi su:

- Algoritam čistog slijeđenja (*engl. Pure pursiut control*),
- *Stanley* regulator,
- MPC regulator.

Algoritam čistog slijeđenja [4] koristi kinematički model vozila sa dva kotača (slika 4.1.). Uz model vozila te podesivog parametra udaljenosti predviđanja (*engl. lookahead distance*) dobije se izraz za kut skretanja $\delta = \frac{2L\sin(\alpha)}{l_d}$ koji minimizira udaljenost d_e. Udaljenost predviđanja l_d definira točku na referentnoj putanji (g_x, g_y) u kojoj bi se trebala naći stražnja osovina vozila pri stalnom kretanju u smjeru izračunatoga kuta skretanja δ.

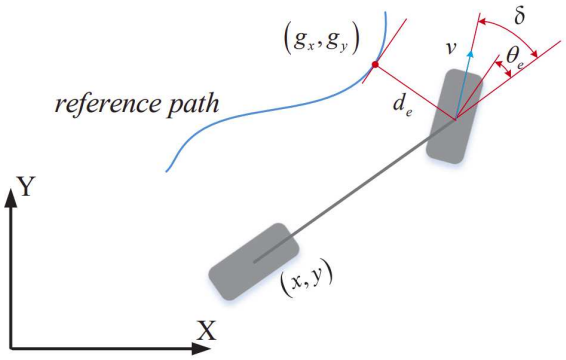
Kut δ služi kao referenca uz traženu brzinu vozila izračunatu nezavisnim algoritmom. Ostaje samo upotrijebiti algoritam koji će u stvarnom vremenu pratiti zadane reference. Najčešće su upotrijebljena dva PID (*engl. Proportional – Integral – Derivative*) regulatora radi jednostavnosti, jedan za praćenje reference skretanja, a drugi za praćenje reference brzine. Parametar predviđanja l_d najčešće se skalira prema brzini vozila. Pri niskim brzinama vozila l_d parametar je vrlo mali zbog čega je skretanje agresivnije, a pri velikim brzinama l_d je velik te za posljedicu vozilo siječe stazu u korist stabilnosti sustava.



Slika 4.1. Shema algoritma čistog slijeđenja.

Stanley regulator [5] sličan je algoritmu čistog slijeđenja. Umjesto da računa referencu kuta skretanja prema stražnjoj osovini, kut skretanja računa se prema prednjoj osovini (slika 4.2.). Uz minimizaciju udaljenosti d_e također se radi i korekcija skretanja prema zakrivljenosti referentne putanje u točki (g_x, g_y). Što znači da ako je vozilo blizu referentne putanje ali ima grešku u orijentaciji, *Stanley* regulator će zadati referencu skretanja tako vozilo prati nagib putanje, dok će algoritam čistog slijeđenja gledati samo da se vrati na putanju što ponovo može rezultirati prekoračenjem putanje. Generalno, *Stanley* regulator je stabilniji od algoritma čistog slijeđenja jer uzima u obzir i korekciju orijentacije vozila prema referentnoj putanji.

Algoritam čistog slijeđenja i *Stanley* regulator su samo načini zadavanja referenci, dok regulatori poput PID, LQR (*engl. linear–quadratic regulator*) ili MPC zapravo upravljaju aktuatorima. Reference su već dovoljno specifične poput kuta volana ili zahtijevane brzine. U nastavku rada razvijen je MPC na proširenom modelu vozila koji može pratiti i kompliciranije reference poput udaljenosti vozila od referentne putanje ili orijentacije vozila naspram referentne putanje.



Slika 4.2. Shema algoritma *Stanley* regulatora.

MPC ima bolje performanse od ostalih vrsta regulatora zbog mogućnosti predviđanja. Usporedba PID regulatora i MPC regulatora već je odrađena u [6].

Najveća prednost MPC-a naspram ostalih upravljačkih algoritama je prediktivno djelovanje. U sadašnjem trenutku može se djelovati na stanja ili smetnje koje su predviđene u budućnosti. Pri kontroli vozila to bi značilo da vozilo može predvidjeti zavoj te poduzeti akcije u sadašnjem trenutku (kočenje, skretanje) tako da u budućnosti uđe u zavoj pod optimalnom brzinom i orijentacijom. Algoritam čistog slijeđenja isto generira reference na temelju parametra predviđanja, no MPC ima integrirano predviđanje cijelim modelom u samom radu algoritma.

4.1. Matematička pozadina

Modelsko prediktivno upravljanje nalazi se u kategoriji optimalnih regulatora. Uz MPC tu je i LQR regulator. Karakteristika optimalnih regulatora je ta što za upravljanje koriste optimizacijsku funkciju. Jednostavniji primjer je linearni vremenski nepromjenjivi LQR regulator koji za optmizacijsku funkciju koristi funkciju:

$$J = \int_0^\infty \left(\mathbf{x}^\top(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^\top(t)\mathbf{R}\mathbf{u}(t)\right) dt. \tag{4.1}$$

Gdje se J zove funkcija troška. Model upravljanog procesa u obliku prostora stanja ima oblik:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \tag{4.2}$$

Gdje su:

- $\mathbf{x}(t) \in \mathbb{R}^n$ je vektor varijabli stanja,
- $\mathbf{u}(t) \in \mathbb{R}^m$ je vektor ulaznih varijabli (upravljačkog signala),
- $\mathbf{A} \in \mathbb{R}^{n \times n}$ je matrica stanja,
- $\mathbf{B} \in \mathbb{R}^{n \times m}$ je matrica ulaznih varijabli,
- $\mathbf{Q} \in \mathbb{R}^{n \times n}$ je matrica težina na varijable stanja,
- $\mathbf{R} \in \mathbb{R}^{m \times m}$ je matrica težina na ulaze.

Skalarna vrijednosti n je broj varijabli stanja, a vrijednost m je broj ulaza u proces (upravljačkih signala procesa).

Proces je u ovom slučaju vozilo, varijable stanja su vrijednosti karakteristične za vozilo (više o tome u poglavlju 5.1.), a ulazne varijable su ulazi u proces (signali kojima se proces upravlja) poput momenta na kotačima, zakreta volana itd.

Cilj regulatora je pronaći ulazne varijable $\mathbf{u}(t)$ tako da funkcija troška J bude minimalna. Funkcija troška ima kvadratni oblik što znači da kada bi se sustav sveo na jednu dimenziju, funkcija bi imala oblik kvadratne funkcije. Kvadratna funkcija ima samo jedan minimum koji je ujedno i globalni i lokalni minimum što olakšava traženje minimuma.

Integralni član u funkciji troška označava da se optimizacija odvija kroz određeni vremenski horizont koji je u ovome slučaju beskonačan. Ili za diskretne sustave, funkcija troška je suma svih stanja u nekom diskretnom horizontu koji je u ovome slučaju beskonačan. Podesive vrijednosti su matrice **Q** i **R** koje uvelike utječu na karakteristike upravljanja. Povećavanjem vrijednosti matrice **Q** kažnjavaju se velike vrijednosti varijable stanja, što bi značilo da će regulator kroz vrijeme nastojati držati varijable stanja što nižim kako bi funkcija gubitka bila što niža. Na isti način, povećavanjem matrice **R** kažnjavaju se velike vrijednosti ulaznih signala, što znači da će regulator kroz vrijeme nastojati primjenjivati što niže vrijednosti ulaznih signala. Na nižim vrijednostima matrice **R** regulator će upravljati procesom agresivno sa velikim ulaznim veličinama, dok

će na malim vrijednostima matrice **R** regulator preferirati glatko upravljanje malim aktuacijskim signalima.

LQR regulator može se implementirati i na kontinuiranim i na diskretnim sustavima, također može imati beskonačni optimizacijski horizont te može garantirati stabilnost sustava. Nedostatak LQR regulatora je što se ne mogu definirati granice na ulazne signale te varijable stanja.

Za razliku od LQR regulatora, MPC može raditi samo sa diskretnim sustavima. Mora koristiti model procesa kako bi predvidio varijable stanja i ulaze te najčešće koristi konačni predikcijski horizont. Stabilnost regulatora nije zagarantirana. Povećavanjem predikcijskog horizonta povećava se stabilnost regulatora jer se daje više vremena da regulator dođe u ravnotežno stanje. MPC koristi načelo pomičnog horizonta (*engl. Receding horizon control*, RHC) za nalaženje optimalnog ulaza u proces.

Prvi korak načela pomičnog horizonta nad linearnim vremenski nepromjenjivim sustavom je predikcija varijabli stanja te ulaza pomoću diskretnog modela sustava:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \tag{4.3}$$

gdje su:

- $\mathbf{x}_k \in \mathbb{R}^n$ vektor varijabli stanja u diskretnom trenutku k,
- $\mathbf{u}_k \in \mathbb{R}^m$ vektor ulaza u diskretnom trenutku k,

a matrice **A** i **B** su diskretizirane iz kontinuiranog modela.

Drugi korak je sastavljanje optimizacijakog problema nad definiranim predikcijskim horizontom:

$$J = \sum_{k=0}^{N-1} \left(\mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^\top \mathbf{R}\mathbf{u}_k\right) + \mathbf{x}_N^\top \mathbf{S}\mathbf{x}_N, \tag{4.4}$$

te vrijedi:

- N je veličina predikcijskog horizonta,

- $\mathbf{Q}$ i $\mathbf{R}$ su matrice težina,
- $\mathbf{S} \in \mathbb{R}^{n \times n}$ je matrica težina na posljednje stanje.

Gdje su nad varijablama stanja i ulazima definirana ograničenja

$$\mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max}, \tag{4.5}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}. \tag{4.6}$$

Nakon optimizacije dobije se N optimalnih ulaznih varijabli za svaki od predikcijskih koraka. Za načelo pomičnog horizonta specifično je da se za upravljački signal u trenutnom koraku uzme samo prva optimalna upravljačka varijabla $\mathbf{u}_0$.

Konačni izgled optimizacijskog problema izgleda ovako:

$$\min_{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}} J. \tag{4.7}$$

uz uvjete:

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \quad k = 0, \ldots, N-1 \\
\mathbf{x}_0 &= \mathbf{x}_{\text{est}} \\
\mathbf{x}_{\min} &\leq \mathbf{x}_k \leq \mathbf{x}_{\max}, \quad k = 0, \ldots, N \\
\mathbf{u}_{\min} &\leq \mathbf{u}_k \leq \mathbf{u}_{\max}, \quad k = 0, \ldots, N-1
\end{aligned}$$

Jednadžba 4.7 tvrdi da se računaju varijable $\mathbf{u}_0$ do $\mathbf{u}_{N-1}$ takve da se minimizira funkcija J. Uz to da je sljedeće stanje sustava zadano jednadžbom varijabli stanja, a moraju se ispuniti kriteriji da su varijable stanja $\mathbf{x}_k$ i $\mathbf{u}_k$ unutar definiranih granica. Također mora biti zadano i početno stanje sustava ($\mathbf{x}_0 = \mathbf{x}_{\text{est}}$) pomoću kojega se može obaviti predikcija.

Zadana kvadratna funkcija troška uz definirana ograničenja može se riješiti kvadratnim programiranjem. U radu se neće detaljno prolaziti teorija algoritama za rješavanje ovakvih problema.

Izraz $\mathbf{x}_N^\top \mathbf{S} \mathbf{x}_N$ unutar funkcije troška naziva se terminalni izraz. Pošto stabilnost MPC-

a nije zagarantirana, dodaje se terminalni izraz koji kažnjava varijable stanja u zadnjem koraku predikcije koje nisu u ravnotežnom stanju. Na ovaj način MPC će u ograničenom horizontu probati što više ustabiliti varijable stanja zadnjeg koraka čime se povećava stabilnost regulatora.

Prethodne jednadžbe vrijede za promatranje sustava u povratnoj vezi bez reference (referenca je 0). Za praćenje reference, kao što je slučaj u ovome radu, pogodniji izraz za funkciju troška je:

$$J = \sum_{k=0}^{N-1} \left(\mathbf{e}_k^\top \mathbf{Q} \mathbf{e}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \right) + \mathbf{e}_N^\top \mathbf{S} \mathbf{e}_N, \tag{4.8}$$

gdje vrijedi:

- $\mathbf{r}_k \in \mathbb{R}^n$ je referentni signal u trenutku k,
- $\mathbf{e}_k = \mathbf{x}_k - \mathbf{r}_k \in \mathbb{R}^n$ je vektor pogreške u trenutku k.

Vidi se razlika naspram prethodne jednadžbe. Umjesto korištenja varijabli stanja $\mathbf{x}_k$ koristi se vektor pogreške $\mathbf{e}_k$. Može se uočiti da će funkcija troška stvarno biti minimalna kada je vektor pogreške jednak nuli.

Ovo je standardni izraz funkcije troška i ograničenja na varijable, no funkcija troška može sadržavati i druge izraze kao što su promjena ulaznih signala ili bilo koje proizvoljno definirane uvjete.

Ograničenja navedena u jednadžbi 4.7 zovu se tvrda ograničenja. Algoritam za rješavanje problema nikada neće prekršiti ograničenja, a prelazak ograničenja rezultira greškom u optimizaciji. U nekim fizikalnim procesa nema potrebe za tvrdim ograničenjima pa se može dozvoliti mali prelazak varijabli izvan ograničenja. Tada tvrda ograničenja postaju meka ograničenja.

Meka ograničenja unutar MPC-a mogu se realizirati uvođenjem dodatnog vektora ϵ [7]. Stavljanjem varijable ϵ u kriterijsku funkciju i ograničenja mogu se realizirati meka ograničenja.

$$J = \sum_{k=0}^{N-1} \left(\mathbf{e}_k^\top \mathbf{Q} \mathbf{e}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \right) + \mathbf{e}_N^\top \mathbf{S} \mathbf{e}_N + \epsilon^\top \lambda \epsilon^\top, \tag{4.9}$$

$$\begin{aligned} \mathbf{x}_{\min} - \epsilon \leq \mathbf{x}_k \leq \mathbf{x}_{\max} + \epsilon, \quad & k = 0, \ldots, N \\ \mathbf{u}_{\min} - \epsilon \leq \mathbf{u}_k \leq \mathbf{u}_{\max} + \epsilon, \quad & k = 0, \ldots, N-1 \\ \epsilon \geq 0. & \end{aligned}$$

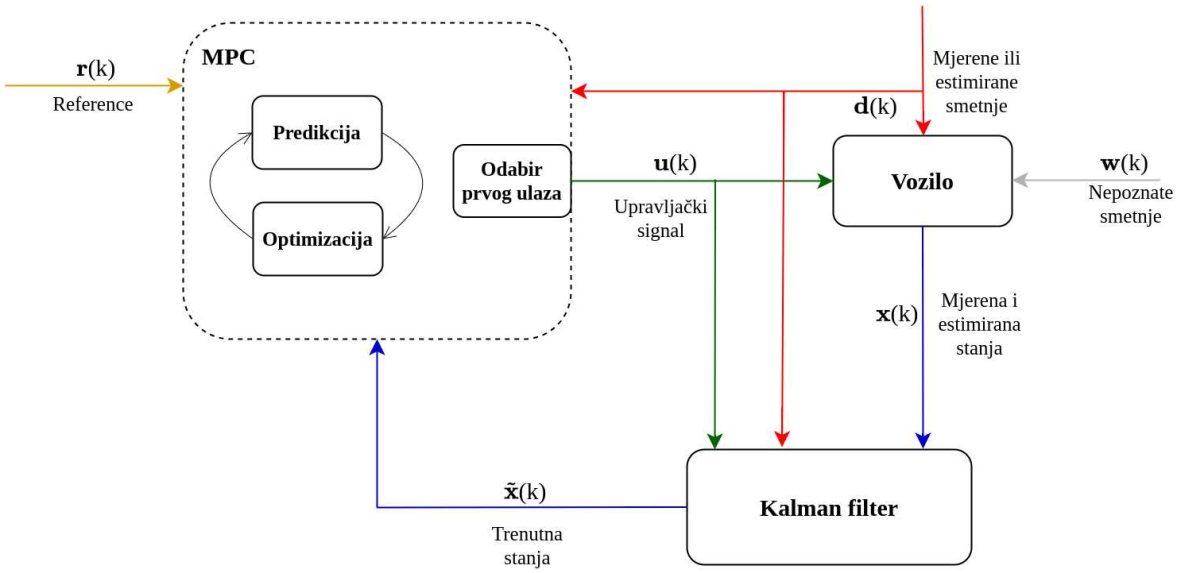
Varijabla ϵ će se povećati za iznos prelaska ograničenja što će utjecati na minimum kriterijske funkcije. Skalar λ je težina koja odlučuje koliko će se penalizirati izlazak van ograničenja. Varijabla λ može se smatrati regularizacijskim faktorom.

Kroz prethodne jednadžbe, vektor izlaza $\mathbf{y}_k$ u kontekstu jednadžbe izlaza $\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k$ je zanemaren, tj. smatra se da su izlazi upravo varijable stanja što čini matricu $\mathbf{C}$ jediničnom matricom.

4.2. Princip rada

Način kako bi se modelsko prediktivno upravljanje trebalo koristiti u zatvorenoj petlji upravljanja vozila prikazan je na slici 4.3.

Ulazi u MPC su reference koje treba pratiti, trenutna stanja upravljanog sustava te trenutne mjerene ili estimirane smetnje (ako su modelirane). Na temelju tih ulaza MPC računa upravljački signal koji upravlja vozilom s ciljem praćenja reference. Trenutna stanja koja ulaze u MPC trebaju biti što točnija stoga se obično koristi Kalman filter nad mjerenjima. U sklopu ovoga rada nije implementiran Kalman filter, već se mjerena stanja vozila prosljeđuju direktno u MPC ili prolaze kroz druge filtre za uklanjanje šuma. Nepoznate smetnje nisu uzete u obzir, ali mogu se modelirati u Kalman filtru ili samoj predikciji MPC-a pri čemu se unosi statistička interpretacija signala.



Slika 4.3. Modelsko prediktivno upravljanje u zatvorenoj petlji upravljanja.

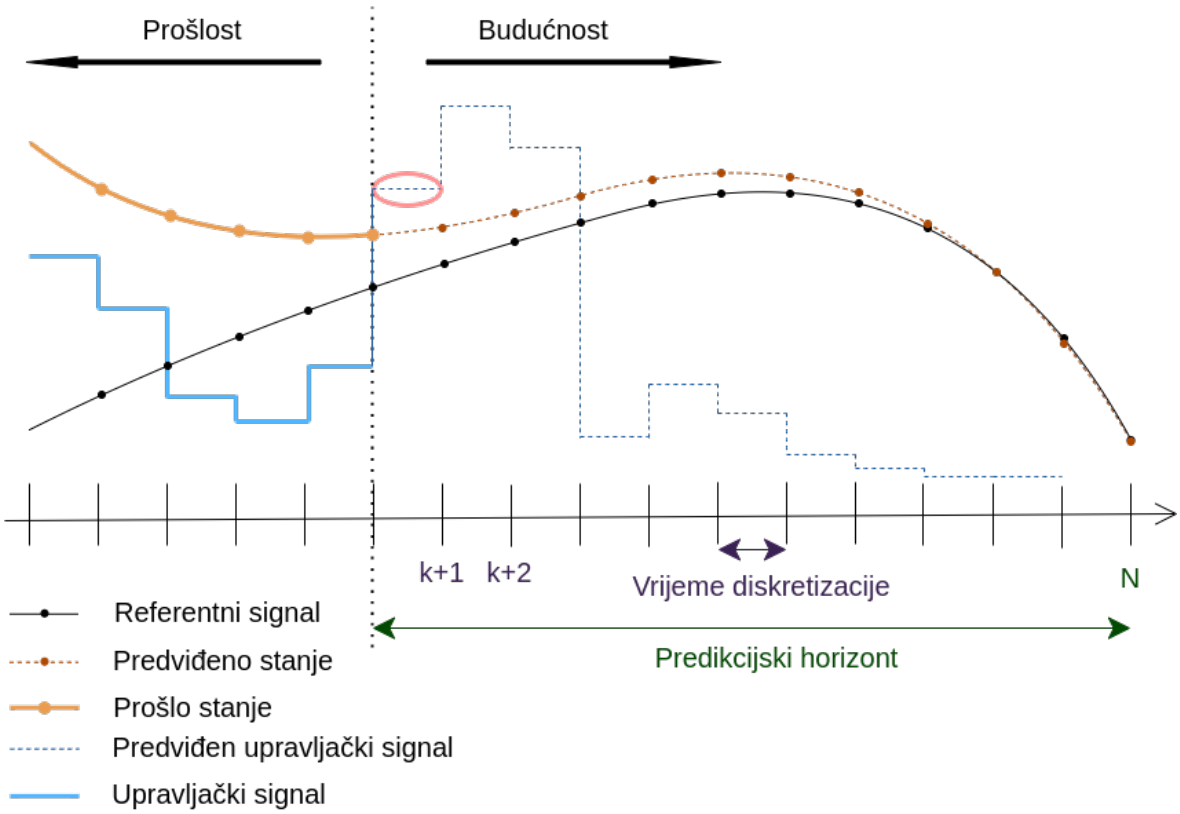
Generalni princip rada jednog izvršavanja modelsko prediktivnog upravljanja prikazan je na slici 4.4. Prikazan je proces kojem je cilj pratiti referencu određenog stanja. Proces ima jedan ulazni signal kojim se upravlja. Zadani su vrijeme diskretizacije i predikcijski horizont. Na temelju trenutnog stanja i poznate reference sastavlja se optimizacijski problem. Optimizator će vratiti optimalna stanja u budućnosti kao i optimalan upravljački signal s kojim se došlo u takva stanja, poštujući sva definirana ograničenja. Po načelu pomičnog horizonta, za idućih N koraka kontrole procesa, neće se uzeti dobivenih N optimalnih upravljačkih signala već se uzima samo prvi upravljački signal (zaokružen crveno na slici). Pri idućem pozivanju MPC-a ponavlja se cijeli proces.

Vrijeme predikcije (t_{pred}) računa se prema:

$$t_{\text{pred}} = TN, \tag{4.10}$$

gdje T predstavlja vrijeme diskretizacije modela zadano u sekundama, a N predstavlja širinu predikcijskog horizonta.

Vrijeme izvršavanja MPC-a i vrijeme diskretizacije ne moraju biti jednaki. Na primjer, MPC se može izvršavati svakih pola sekunde, dok je vrijeme diskretizacije 50 ms. Ako je zadano da je predikcijski horizont 5, MPC će predvidjeti optimalne kontrolne signale za idućih 0.25 s. U tom slučaju primjenjivati će prvi optimalni signal idućih pola



Slika 4.4. Princip rada modelsko prediktivnog upravljanja.

sekunde dok se MPC ponovo ne pozove. Ovakva kontrola bi davala jako loše rezultate, a rješenje bi bilo povećati vrijeme diskretizacije, povećati predikcijski horizont ili povećati frekvenciju izvršavanja MPC-a. Povećavanje vremena diskretizacije, ovisno o modelu, sve se lošije opisuje model te u nekim slučajevima može učiniti model nestabilnim. Povećavanje predikcijskog horizonta može produljiti vrijeme izvršavanja MPC-a jer je problem kompliciraniji, a maksimalna frekvencija pozivanja MPC-a ograničena je samim vremenom izvršavanja.

Uz pojam predikcijski horizont postoji i pojam upravljački horizont. Predikcijski horizont odnosi se na predikciju stanja ili izlaza, dok se upravljački horizont odnosi na predikciju upravljačkih varijabli. Često se definira da je upravljački horizont kraći od predikcijskog, pošto upravljački signali u daljoj budućnosti nisu bitni. Na taj način smanjuje se veličina optimizacijskog problema, a time i vrijeme izvršavanja optimizacije. U nastavku rada upravljački horizont biti će jednak predikcijskom horizontu.

Svako pozivanje MPC-a zahtjeva podatke o trenutnom stanju modela ($\mathbf{x}_{\text{est}}$). Predikcija algoritma je točnija što je model točnije opisan te što je trenutno očitano ili estimi-

rano stanje procesa bliže stvarnom stanju procesa. Stanja procesa inače dolaze iz očitanja senzora ili estimacija nekih varijabli koje se ne mogu direktno mjeriti. Da bi se ostvarilo upravljanje u potpunom zatvorenom krugu, potrebno je u svakom trenutku znati potpuno stanje sustava. Neka stanja su jednostavno nemjerljiva pa se ne može znati potpuno stanje modela. Takvi slučajevi su česti, a tada se za trenutno stanje sustava koje nije mjereno koristi estimacija predikcije iz prošlog koraka. Takva estimacija stanja sustava može driftati kroz vrijeme ili čak postati nestabilna ako model nije dovoljno točan. Najbolji način za izbjeći takve stvari je pametno modelirati sustav, definirajući što više mjerljivih varijabli stanja.

Dodatna prednost MPC algoritama nad ostalim regulacijskim algoritmima je što MPC može direktno upravljati modelima sustava sa više ulaza i više izlaza (*engl. multiple input multiple output*, MIMO). Na primjer, PID regulator može upravljati samo jednim izlazom. Što znači da ako se želi upravljati modelom sa više izlaza mora se upotrijebiti više PID regulatora.

4.3. Tipovi MPC-a

Kontrola vozila je kompliciran problem zbog toga što je model prilično kompleksan i nelinearan. Takve optimizacije moraju se izvršavati aktivnim izvođenjem algoritma u stvarnom vremenu. Nasuprot tome postoje jednostavni sustavi koji su lako predvidljivi te se optimalne upravljačke varijable izračunaju unaprijed (*engl. offline*). Za kontrolu u stvarnom vremenu tada se samo uzmu predefinirane vrijednosti upravljačke varijable ovisno o trenutnom stanju.

MPC izvođen u stvarnom vremenu dijeli se na:

- Linearni MPC,
- Nelinearni MPC.

Razlika je u tome što linearni MPC za dinamiku procesa koristi standardni linearni zapis varijabli stanja $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$, dok nelinearni MPC i koristi nelinearnu funkciju za opis dinamike procesa $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$. Algoritmi rješavanja optimizacijskih problema nisu isti za linearne i nelinearne sustave. Nelinearni sustavi zahtijevaju glo-

balni optimizator, jer u slučaju nelinearne optimizacije može postojati više lokalnih minimuma uz globalni minimum. Tada globalni optimizator mora pronaći globalni minimum funkcije troška, što je teže jer rješenje ne smije konvergirati u lokalni minimum. Linearni kvadratni problemi imaju jedan lokalni minimum funkcije troška koji je ujedno i globalni minimum, što čini optimizaciju puno jednostavnijom.

Za linearni MPC vrijedi da mu funkcija troška smije biti zadana kvadratnom funkcijom koja je nelinearna. Osim funkcije troška koja smije biti kvadratna, sve ostale jednakosti i nejednakosti moraju biti linearne, uključujući dinamiku procesa.

Linearni MPC može rješavati i nelinearne probleme ako se model prije toga linearizira. Takvi algoritmi su:

- Adaptivni MPC,
- Linearni vremenski ovisni (*engl. Linear Time-Varying*, LTV) MPC.

Adaptivni MPC radi na istom principu kao i linearni MPC, osim toga što se pri svakom pozivanju MPC-a matrica stanja **A** ažurira prema trenutnoj radnoj točki. Korak predikcije koristi LTI (*engl. Linear Time-Invariant*) sustav, što znači da je kroz predikciju matrica **A** konstantna.

LTV MPC ne samo da ažurira matricu **A** pri svakom pozivanju algoritma, nego ju ažurira i za vrijeme predikcije. To znači da predikcija koristi LTV sustav. Problematika ove metode je što moramo predvidjeti u kojem stanju će se nalaziti proces kroz predikcijski horizont kako bi unaprijed mogli definirati odgovarajuće matrice stanja. Praksa je da se pri trenutnom pozivu predikcije koriste matrice **A** linearizirane oko predikcija prošlog koraka.

Adaptivni MPC vrlo je dobar za nelinearne sustave koji sporo mijenjaju radnu točku i pravi je izbor za većinu procesa. Međutim, za procese koji iz koraka u korak naglo mijenjaju radnu točku LTV MPC je pravi odabir. Ako se matrice stanja **A** dobro predvide za korak predikcije, LTV MPC ima performanse gotovo kao nelinearni MPC [8].

Obje vrste MPC-a su implementirane i razlike među performansama analizirane su u poglavljima 6. i 7.

Među ostalim MPC regulatorima ističu se:

- Robusni MPC,
- Stohastički MPC.

Robusni MPC koristi se za upravljanje sustavima u vrlo nepredvidivoj okolini sa puno smetnji. Može vrlo dobro pratiti referencu uz vrlo velike smetnje. Zahtijeva jako točan model sustava te se zbog načina zadavanja problema vrijeme izvršavanja povećava eksponencijalno sa širinom predikcijskog horizonta.

Stohastički MPC uvodi nesigurnost u model predikcije $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + w_k$, gdje w_k može biti modeliran kao *Gaussov* šum.

4.4. Optimizacijski algoritam

Najčešća praksa pri konstruiranju optimizacijskog problema modelsko prediktivnog upravljanja je pisanje funkcije troška u kvadratnom obliku. Vrlo čest oblik osim kvadratnog oblika je linearni oblik funkcije troška. Linearni oblici funkcije troška rješavaju se linearnim programiranjem. Linearno programiranje brže rješava problem te je pogodno i za vrlo velike optimizacijske probleme. Kvadratno programiranje je preferirano jer rezultira glatkim upravljačkim akcijama te je podešavanje težinskih parametra funkcije troška intuitivnije [9]. Korisno je napomenuti da kvadratno programiranje može riješiti i problem koji se sastoji od kvadratnih i linearnih članova istovremeno.

Globalni optimizatori za rješavanje nelinearnih problema su sporiji od kvadratnih i linearnih problema, a pogotovo za velike optimizacijske probleme poput upravljanja vozilom. Na primjer, za vrlo pojednostavljen problem upravljanja vozilom, nelinearni problem definiran u radu [10] riješen je u 200 ms. Pri velikim brzinama kretanja vozila i zahtjevima za brzo djelovanje u stvarnome vremenu takvo kašnjenje može biti kritično. Napretkom algoritama za globalnu optimizaciju vremena izvođenja su sve brža, no za velike optimizacijske probleme poput problema koji će biti opisan u ovome radu, globalna optimizacija je i dalje potencijalno spora.

Zbog dovoljno velike brzine izvođenja u stvarnome vremenu te glatkih upravljačkih akcija, problem modelsko prediktivnog upravljanja realiziran je u kvadratnom obliku.

Postoje razne tehnike za rješavanje kvadratnog programiranja, što rezultira velikim brojem algoritama za kvadratno programiranje. U nastavku su navedeni algoritmi i rješenja dobivena mjerenjem performansi nad *Maros-Meszaros* testnim setom [11]. *Maros-Meszaros* testni set sadrži primjere kvadratnog programiranja iz mnogo izvora te se smatra vrlo teškim problemom za riješiti što čini dobar set za testiranje performansi.

Korišteni algoritmi za rješavanje kvadratnog programiranja korišteni u testiranju su:

Tablica 4.1. Popis algoritama za rješavanje kvadratnog programiranja.

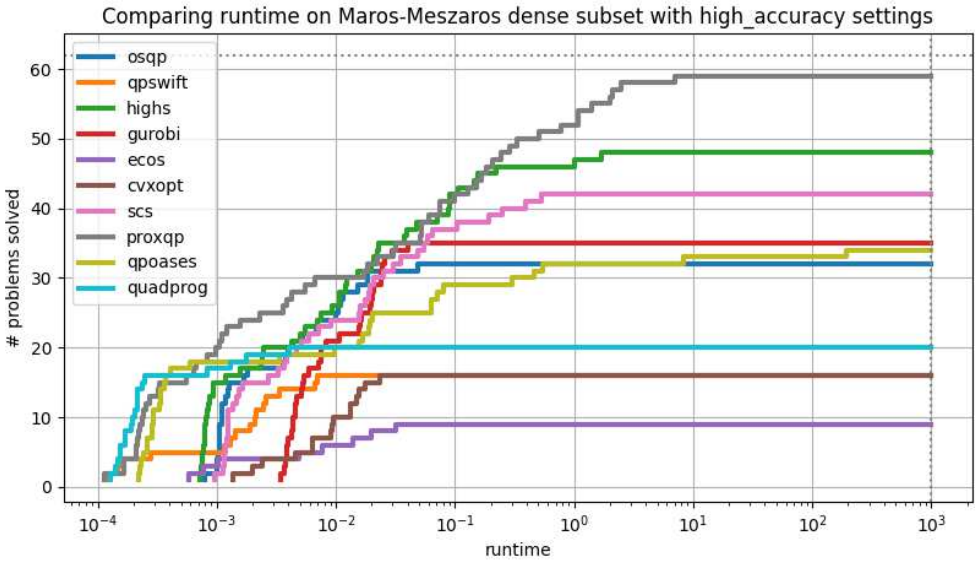
Naziv	**Algoritam**	**Licenca**
Clarabel	Traženje unutarnje točke	Apache-2.0
CVXOPT	Traženje unutarnje točke	GPL-3.0
DAQP	Aktivni skup	MIT
ECOS	Traženje unutarnje točke	GPL-3.0
Gurobi	Traženje unutarnje točke	Komercijalna
HiGHS	Aktivni skup	MIT
HPIPM	Traženje unutarnje točke	BSD-2-Clause
MOSEK	Traženje unutarnje točke	Komercijalna
NPPro	Aktivni skup	Komercijalna
OSQP	Douglas–Rachford	Apache-2.0
PIQP	Proksimalna točka	BSD-2-Clause
ProxQP	Prošireni Lagranžijan	BSD-2-Clause
QPALM	Prošireni Lagranžijan	LGPL-3.0
qpOASES	Aktivni skup	LGPL-2.1
qpSWIFT	Traženje unutarnje točke	GPL-3.0
quadprog	Goldfarb-Idnani	GPL-2.0
SCS	Douglas–Rachford	MIT

Mnogi algoritmi za rješavanje kvadratnog programiranja su licencirani proizvodi. Licence algoritama prikazanih u tablici su:

- Apache-2.0 licenca - dozvoljeno je korištenje, modificiranje i distribuiranje licencirane programske podrške.

- GPL-2.0/3.0 licenca - (*engl. General Purpose License*) je licenca za besplatno korištenje programske podrške.
- LGPL-2.1/3.0 - (*engl. GNU Lesser General Public License*) dozvoljava besplatno korištenje programske podrške.
- BSD-2-Clause - dozvoljava besplatno korištenje programske podrške kao kopiranje i distribuiranje.
- MIT licenca - dozvoljeno je korištenje, modificiranje i distribuiranje licencirane programske podrške.
- Komercijalna licenca - zahtijeva kupovinu licence. Međutim većina komercijalnih algoritama su dostupni za studente.

Performanse navedenih algoritama nad *Maros-Meszaros* setom prikazani su na idućoj slici.



Slika 4.5. Performanse algoritama za rješavanje kvadratnog programiranja testiranih nad *Maros-Meszaros* setom [11].

Vidljivo je da su za male optimizacijske probleme najbrži algoritmi quadprog i qpoases, dok je algoritam koji pokazuje općenito najbolje performanse proxqp. No ovakav test je vrlo specifičan te za različite probleme razni algoritmi imaju drugačije performanse. U potpoglavlju 7.2.5. biti će analizirane performanse različitih optimizacijskih algoritama nad definiranim problemom za modelsko prediktivno upravljanje.

Većina algoritama rješava problem iterativno tražeći minimum funkcije, što zahtijeva početnu pretpostavku za minimum funkcije. Ovisno o tome koliko je početna pretpostavka blizu traženog minimuma funkcije troška, algoritam može brže riješiti problem uz manje iteracija. Zbog toga je u mnogim algoritmima dopušteno da se definira početna pretpostavka iz koje će krenuti optimizacija (*engl. warm-start*). Kod problema modelsko prediktivnog upravljanja praksa je za početnu pretpostavku uzeti prošlo rješenje, tj. minimum izračunat u prošloj optimizaciji. Ako nije uključena opcija korištenja početne pretpostavke, što je zadana postavka svih algoritama, algoritam sam bira početnu točku (*engl. cold-start*).

Standardni problem kvadratnog programiranja za modelsko prediktivno upravljanje, koji zahtijevaju svi algoritmi za rješavanje, ima vektorski oblik [9]:

$$\min_{\mathbf{v}} \left(\frac{1}{2} \mathbf{v}^T \tilde{\mathbf{H}} \mathbf{v} + \mathbf{c}^T \mathbf{v} \right), \tag{4.11}$$

uz uvjete:

$$\mathbf{L}\mathbf{v} \leq \mathbf{b}. \tag{4.12}$$

Vektor $\mathbf{v}$ je vektor slobodnih varijabli u optimizaciji, tj. vektor varijabli čija rješenja se traže, dok je $\tilde{\mathbf{H}}$ hesijan matrica, a vektor $\mathbf{c}$ je linearni dio funkcije troška. Matrica $\mathbf{L}$ i vektor $\mathbf{b}$ opisuju linearna ograničenja. Gornja jednadžba je samo vektorski zapis jednadžbe 4.8 prilagođen problemu kvadratnog programiranja. Hesijan matrica i linearni dio funkcije troška računaju se na ovaj način:

$$\tilde{\mathbf{H}} = \hat{\mathbf{B}}^T \hat{\mathbf{Q}} \hat{\mathbf{B}} + \hat{\mathbf{R}}, \tag{4.13}$$

$$\mathbf{c}^T = \chi_0^T \hat{\mathbf{Q}} \hat{\mathbf{B}}, \tag{4.14}$$

a matrica χ_0^T se računa iz:

$$\chi_0 = \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{N-1} \\ \mathbf{A}^N \end{bmatrix}}_{\hat{\mathbf{A}}} \mathbf{x}_0 + \underbrace{\begin{bmatrix} \mathbf{B} & 0 & \cdots & 0 & 0 \\ \mathbf{AB} & \mathbf{B} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}^{N-2}\mathbf{B} & \mathbf{A}^{N-3}\mathbf{B} & \cdots & \mathbf{B} & 0 \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \cdots & \mathbf{AB} & \mathbf{B} \end{bmatrix}}_{\hat{\mathbf{B}}} (\mathbf{u}_{\text{ref}} - \mathbf{x}_{\text{ref}}). \tag{4.15}$$

Pri čemu je N širina predikcijskog horizonta, $\mathbf{x}_0$ trenutne varijable stanja, $\mathbf{u}_{\text{ref}}$ i $\mathbf{x}_{\text{ref}}$ referentne vrijednosti upravljačkog signala i varijabli stanja ako su zadane, a matrice $\mathbf{A}$ i $\mathbf{B}$ su standardne matrice zapisa modela u prostoru stanja: $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$.

Proširene težinske matrice na varijable stanja i kontrolne signale dobiju se slaganjem težinskih matrica za svaki korak horizonta po dijagonali:

$$\hat{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{Q} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathbf{Q} & 0 \\ 0 & 0 & \cdots & 0 & \mathbf{S} \end{bmatrix}_{NxN}, \quad \hat{\mathbf{R}} = \begin{bmatrix} \mathbf{R} & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{R} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathbf{R} & 0 \\ 0 & 0 & \cdots & 0 & \mathbf{R} \end{bmatrix}_{NxN}. \tag{4.16}$$

Linearna ograničenja iz svakog koraka predikcije slažu se u retke te se na taj način dobiju matrica $\mathbf{L}$ i vektor $\mathbf{b}$.

4.5. Odabir platforme

Modelsko prediktivno upravljanje mora biti implementirano na vozilu VulpesD. Zbog kompleksnosti zadatka, MPC razvijen u ovom radu mora se izvršavati u stvarnom vremenu. Sklopovlje dostupno za implementaciju unutar vozila je autonomno računalo te

ECU.

Računalno i vremenski najzahtjevniji korak modelsko prediktivnog upravljanja je rješavanje optimizacijskog problema. Radi jednostavnosti definiranja i rješavanja optimizacijskih problema, odlučeno je koristiti alate otvorenog koda koji su posrednici među optimizacijskim algoritmima i korisnikom. Najpoznatiji alati su *CasADi* te YALMIP.

CasADi je dostupan u jezicima *Python*, *Matlab* te *C++*. Korištenje MPC-a na autonomnom računalu moglo bi biti lako implementirano pomoću jezika *Python* i *C++* u već postojeću ROS (*engl. Robot operating system*) strukturu. Za integriranje MPC-a u sklopu postojećeg ECU *Simulink* modela, YALMIP je dobro podržan za korištenje unutar *Simulinka* te omogućuje generiranje potrebnog C koda.

Autonomno računalo ima bolje sklopovlje te bi moglo izvršavati MPC u kraćem vremenu nego ECU. No prednost ECU-a je što se jedna cijela jezgra može posvetiti izvršavanju algoritma, dok će autonomno računalo raspodjeljivati resurse zadatku. Najveća prednost izvršavanja MPC-a na ECU je izravna dostupnost svih podataka zbog iznimne centraliziranosti sustava. Implemetacijom MPC-a na autonomnom računalu rezultiralo bi puno dužim vremenom zatvorenog ciklusa komunikacije nego implementacijom na ECU. Pogotovo iz razloga što ECU i autonomno računalo komuniciraju uz podršku ROS operacijskog sustava, koji koristi komunikacijski protokol nepogodan za izvršavanje u stvarnom vremenu. Dok je ROS 2 više razvijen u smislu izvršavanja u stvarnom vremenu, ROS je relativno spor. Mjereni primjer je slanje ECU podatka na autonomno računalo, gdje je komunikacija dostigla najbržu brzinu od 100 Hz, što je kašnjenje podatka od 10 ms u najboljem slučaju.

Za MPC primjene u ostalim vozilima, uz naglasak na izvođenje u stvarnome vremenu, tipične frekvencije rada algoritma su od 50 Hz do 100 Hz [12]. To bi značilo da se cijeli MPC treba izvršiti unutar 20 ms, ako se uzme u obzir frekvencija rada od 50 Hz.

Potpuni krug upravljanja u slučaju implementacije MPC-a na autonomnom računalu (slika 3.3.) bio bi:

1. Prikupljanje podataka sa senzora i osvježavanje informacija na ECU (~5 ms),

2. Slanje podataka na računalo autonomnog upravljanja (~10 ms),

3. Izvršavanje MPC-a i dobivanje upravljačkog signala (~20 *ms*),

4. Slanje podataka na ECU (~10 *ms*),

5. Osvježavanje informacija na ECU za izvedbu komande (~4 *ms*).

Zbog velike prednosti izbjegavanja popriličnog kašnjenja, a malog nedostataka potencijalno sporijeg vremena izvršavanja MPC algoritma, odlučeno je implementirati MPC koji će se izvršavati na ECU.

Korištenje alata YALMIP pogodno je za integraciju MPC-a u postojeći *Simulink* model. Osim toga što omogućuje programiranje u *Simulinku* te generiranje C koda, YALMIP koristi simboličke varijable te vrlo jednostavnu logiku pri sastavljanju optimizacijskog problema. Optimizacijski problem je vrlo intuitivan jer se može zadati u općenitoj formi (jednadžba 4.7), a pozivanje optimizatora i prebacivanje problema u formu definiranu jednažbom 4.11 izvršava se u pozadini. Cijela interakcija sa optmizatorima izvedena je kroz YALMIP, što znači da se vrlo jednostavno mogu mijenjati vrste optimizatora korištene pri rješavanju MPC problema.

5. Model vozila

Predikcija modelsko prediktivnog upravljanja točnija je što je matematički model sličniji dinamici i vladanju stvarnoga procesa. U ovome poglavlju odabire se model vozila koji zahtijeva linearizaciju te diskretizaciju zbog uporabe linearnog MPC-a. Konačna jednadžba mora biti u obliku $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$.

Cilj je što bolje upravljati vozilom stoga je potrebno odabrati model vozila koji će se koristiti. Matematički model vozila može biti opisan na dva načina:

- Kinematički model vozila - modeliraju vozilo na principu jednadžbi gibanja. Ne uzimaju u obzir sile i momente na vozilo.
- Dinamički model vozila - modelira sve sile i momente koji djeluju na vozilo.

Dinamički model smatra se puno točnijim modelom jer su opisane sve fizikalne interakcije. Radi točnosti modela, dinamički model vozila odabran je kao predikcijski model. Pri modeliranju u obzir se mogu uzeti i iduća dva načina za modeliranje kotača.

- Model vozila sa 4 kotača
- Model vozila sa 2 kotača - bicikl model

Model vozila sa 2 kotača pojednostavljuje model na način da modelira samo jedan kotač po osovini. Bicikl model često je korišten u raznim algoritmima jer uvodi značajno pojednostavljenje, a performans pri MLTP problemu je vrlo sličan onome koji koristi model sa 4 kotača [3]. Radi potencijalnog integriranja aktivnih upravljačkih algoritama u razvijeni MPC za autonomnu vožnju, a pogotovo kod vozila sa pogonom na sva 4 kotača, odabran je model vozila sa 4 kotača.

5.1. Jednadžbe i varijable stanja

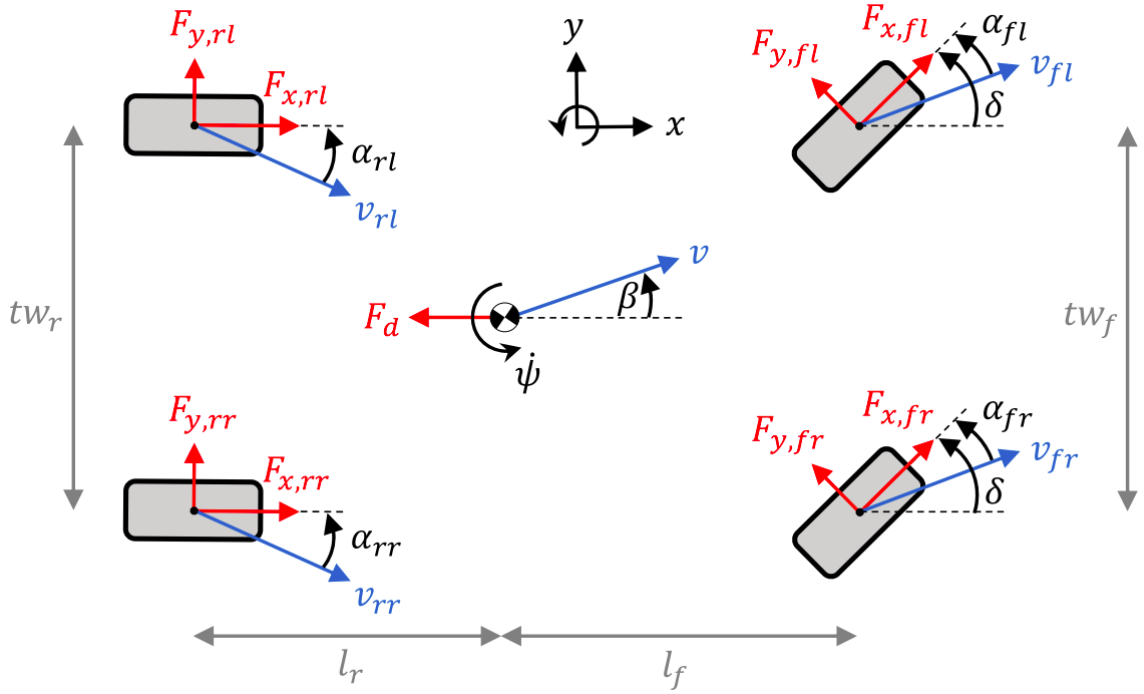
Pošto korišteni optimizator trajektorije također koristi puni dinamički model vozila za generiranje referenci MPC-u [3], za model vozila koristiti će se identične jednadžbe. Jednadžbe vozila su:

$$\begin{aligned}
\dot{v} = \frac{1}{m}\Big[&\big(F_{x,rl} + F_{x,rr}\big)\cos(\beta) + \big(F_{x,fl} + F_{x,fr}\big)\cos(\delta - \beta) \\
&+ \big(F_{y,rl} + F_{y,rr}\big)\sin(\beta) - \big(F_{y,fl} + F_{y,fr}\big)\sin(\delta - \beta) - \frac{1}{2}c_d \rho A v^2 \cos(\beta)\Big], \qquad (5.1)\\
\dot{\beta} = -\omega_z + \frac{1}{mv}\Big[&\big(F_{x,rl} + F_{x,rr}\big)\sin(\beta) - \big(F_{x,fl} + F_{x,fr}\big)\sin(\delta - \beta) \\
&+ \big(F_{y,rl} + F_{y,rr}\big)\cos(\beta) + \big(F_{y,fl} + F_{y,fr}\big)\cos(\delta - \beta) + \frac{1}{2}c_d \rho A v^2 \sin(\beta)\Big], \qquad (5.2)\\
\dot{\omega}_z = \frac{1}{J_{zz}}\Big[&-\big(F_{x,rr} - F_{x,rl}\big)\frac{tw_r}{2} - \big(F_{y,rl} + F_{y,rr}\big)l_r \\
&+ \Big(\big(F_{x,fr} - F_{x,fl}\big)\cos(\delta) + \big(F_{y,fl} - F_{y,fr}\big)\sin(\delta)\Big)\frac{tw_f}{2} \\
&+ \Big(\big(F_{y,fl} + F_{y,fr}\big)\cos(\delta) + \big(F_{x,fl} + F_{x,fr}\big)\sin(\delta)\Big)l_f\Big]. \qquad (5.3)
\end{aligned}$$

Jednadžbe opisuju derivaciju ukupne brzine ($\dot{v}$), derivaciju bočnog kuta klizanja ($\dot{\beta}$) te derivaciju kutne brzine skretanja vozila ($\dot{\omega}$). Indeksi x i y unutar jednadžbi odnose se na smjerove sila definirane na slici 5.1. Za označavanje kotača koriste se parovi indeksa. Na prvome mjestu nalaze se indeksi koji označavaju kotače prednje osovine (*engl. front* - f) te kotače zadnje osovine (*engl. rear* - r). Na drugome mjestu su indeksi za razlikovanje lijevog kotača (*engl. left* - l) te desnog kotača (*engl. right* - r). Definicije konstanti navedene su u tablici 5.1.

Koordinatni sustav vozila poštuje pravila desne ruke. Svi kutevi korišteni u modelu imaju pozitivne vrijednosti pri orijentaciji koja je u lijevo u odnosu na nultu poziciju, a negativne vrijednosti u suprotnome.

Uz model vozila, model za predikciju proširen je i jednadžbama koje su bitne za autonomnu vožnju:



Slika 5.1. Opis sila i varijabli dinamičkog modela vozila [3].

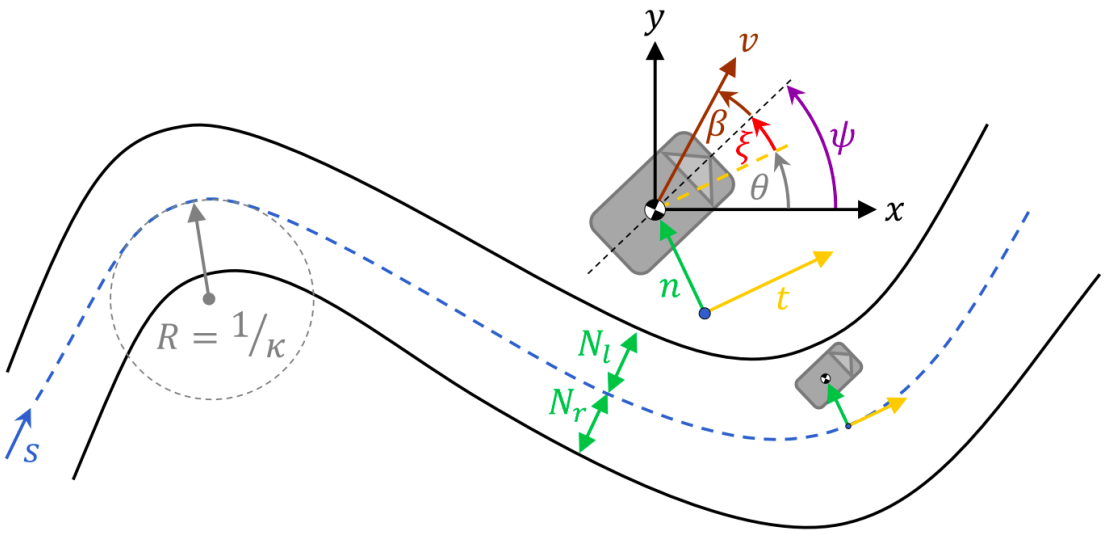
$$\dot{s} = \frac{v\cos(\xi+\beta)}{1-n\kappa}, \tag{5.4}$$

$$\dot{n} = v\sin(\xi+\beta), \tag{5.5}$$

$$\dot{\xi} = \omega_z - \kappa\frac{v\cos(\xi+\beta)}{1-n\kappa}. \tag{5.6}$$

Jednadžbe opisuju odnos vozila prema stazi izražene u krivolinijskom sustavu. Prednost krivolinijskog sustava je jednostavnije matematičko modeliranje naspram koordinatnog sustava te intuitivnije izražavanje varijabli bitnih za stazu i vozilo. Varijabla s predstavlja prijeđeni put, dok n označava udaljenost vozila od referentne putanje na trenutnom prijeđenom putu. Kut ξ označava orijentaciju vozila naspram tangente na referentnu putanju ili kut deklinacije vozila u odnosu na referentnu putanju (slika 5.2.).

Na slici 5.2. definirane su vrijednosti N_l i N_r koje predstavljaju udaljenost granica staze od referentne putanje na lijevu i desnu stranu. Vrijednost κ definira zakrivljenost referentne putanje u svakoj točki puta. Sve definirane udaljenosti i varijable definiraju se naspram centra mase vozila.



Slika 5.2. Opis vrijednosti koje predstavljaju odnos vozila i putanje [3].

Na općenitoj razini, spomenute jednadžbe opisuju vladanje samog vozila te odnos vozila prema stazi kojom se vozi. Na ovaj način, predikcija modelsko prediktivnog upravljanje uključuje puno ponašanje autonomnog vozila.

Kotači su jedini kontakt vozila sa površinom po kojoj vozi stoga je bitno uzeti i dobar model gume. Neki od poznatijih modela gume su:

- Vlaknasti model gume,
- Model gume definiran "magičnom" formulom.

Vlaknasti model gume pripada fizikalnoj kategoriji modela gume te opisuje gumu sa beskonačno mnogo mikroskopskih vlakana. Prednost vlaknastog modela gume je opisivanje jednostavnim formulama što ga čini vrlo primjenjivim u aktivnim upravljačkim algoritmima. Model gume definiran "magičnom" formulom pripada empirijskoj kategoriji modela gume. Model gume opisan je formulom čiji su koeficijenti dobiveni aproksimacijom krivulje nad testnim podacima. Takav model vrlo dobro opisuje vladanje gume, ali jednadžba je vrlo nelinearna te simulacije vozila sa takvim modelom zahtijeva iznimno nisko vrijeme uzorkovanja. Radi velike točnosti modela, u nastavku se koristi model gume definiran "magičnom" formulom.

Lateralna dinamika vozila opisana je silama na kotače uz korištenu pojednostavljenu "magičnu" formulu za model gume:

$$F_{y,ij} = \mu_{ij} F_{z,ij} D_i \sin\left(C_i \arctan\left(B_i \alpha_{ij}\right)\right). \tag{5.7}$$

Koeficijent μ predstavlja koeficijent trenja između kotača i podloge. Koeficijent je različit za svaki kotač te je ovisan o vrsti podloge po kojoj se vozi. U [3] izrađena je mapa dinamičkih koeficijenata trenja po stazi pri rješavanju MLTP optimizacije. Na *Formula Student* natjecanjima takva trenja se ne mogu mjeriti ili znati u naprijed pa će koeficijent μ biti jednaka konstanta za sva 4 kotača i neće se mijenjati kroz vrijeme. Koeficijenti D, C i B su specifični koeficijenti gume dobiveni aproksimacijom krivulje modela gume definiranog "magičnom" formulom. Na trkaćim vozilima obično se koriste različiti parovi guma na prednoj i stražnjoj osovini pa je bitno uzeti u obzir da gume na stražnjoj osovini imaju različite koeficijente od onih na prednjoj. Radi jednostavnosti simulacije pretpostavljeno je da su prednje i stražnje gume jednake. Kut α predstavlja bočni kut klizanja kotača te je različit za prednje i zadnje kotače zbog geometrije skretanja:

$$\alpha_{fl/fr} = \delta - \arctan\left(\frac{l_f \omega_z + v \sin(\beta)}{v \cos(\beta) \mp \frac{1}{2} t w_f \omega_z}\right), \tag{5.8}$$

$$\alpha_{rl/rr} = \arctan\left(\frac{l_r \omega_z - v \sin(\beta)}{v \cos(\beta) \mp \frac{1}{2} t w_r \omega_z}\right). \tag{5.9}$$

Nužna pretpostavka ovakvog modela jest da je $v\cos(\beta) >> \left|\frac{1}{2} t w_f \omega_z\right|$. Pri vrlo malim brzinama brojnik se dijeli sa vrlo malim brojem što rezultira iznimno visokim bočnim kutevima klizanja što je nemoguće. Zbog toga ovakav model se ne može koristiti pri vrlo niskim brzino što uvelike utječe na primjenu MPC-a. Funkcija $arctan$ ima sliku funkcije u prvom i četvrtom kvadrantu. Vrijednost u nazivniku neće biti negativna zbog korištenja modela samo pri velikim brzinama, stoga će slika funkcije biti uvijek u prvom i četvrtom kvadrantu. Da to nije slučaj morala bi se koristiti $arctan2$ funkcija kojoj slika funkcije ovisi predznaku vrijednosti u nazivniku. Ona se ne može jedinstveno modelirati te bi trebalo stvoriti dva potpuna modela vozila koja bi se izmjenjivala ovisno o predznaku u nazivniku.

Pojednostavljena vertikalna sila na kotač u izrazu 5.7 definirana je formulom:

$$F_{z,fj} = \frac{1}{2}mg\frac{l_r}{l_f + l_r} + \frac{1}{4}c_{l,f}\rho A v^2, \tag{5.10}$$

$$F_{z,rj} = \frac{1}{2}mg\frac{l_r}{l_f + l_r} + \frac{1}{4}c_{l,r}\rho A v^2, \tag{5.11}$$

gdje prvi član predstavlja statičku vertikalnu silu na kotač uzrokovanu gravitacijom, a drugi silu aerodinamičkog tlaka na svaki kotač. Uvedeno je značajno pojednostavljenje jer je izostavljena dinamika pri skretanju koja ima najveći utjecaj na vertikalne sile. Dinamika pri skretanju računa se pomoću lateralne akceleracije koja ima kompliciran izraz za derivaciju. Akceleracija se stoga obično raspisuje preko sila, a u ovom slučaju to bi stvorilo algebarsku petlju te je zbog toga izostavljena.

Pri računanju vertikalne sile također je zanemaren dinamički prijenos težišta vozila koji se događa prilikom ubrzavanja ili usporavanja.

Ostaje još modelirati longitudinalne sile vozila $F_{x,ij}$ te vladanje kuta zakreta kotača δ pri skretanju. Ponašanje tih veličina specifične su za vozilo te načine aktuiranja. Da bi se vjerno modeliralo ponašanje vozila pri vožnji bitno je uzeti u obzir načine aktuacije vozila VulpesD.

Glavne varijable koje djeluju na longitudinalne sile vozila su kočenje te pogonski motori, a na kut zakreta kotača aktuatori za skretanje. Longitudinalne sile mogu se izraziti preko longitudinalnih momenata uzevši u obzir polumjer kotača prema formuli $M_{x,ij} = r_{\text{ef}}F_{x,ij}$, pri čemu je r_{ef} efektivni polumjer kotača [1]. Momenti na kotačima mogu se koristiti kao upravljački signali koje MPC računa što je slučaj u [6]. Takav pristup je često korišten kod vozila koji imaju pogon na sva 4 kotača te regenerativno kočenje. MPC bi zadao svakom pogonskom motoru moment koji mora ostvariti, a uzevši u obzir veliku brzinu odziva pogonskih motora takva komanda bi se mogla smatrati trenutnom. U slučaju vozila VulpesD koje ima pogon na stražnje kotače bez regenerativnog kočenja te odvojenu logiku za kočenje, takvo modeliranje unosi problematiku. Tada bi momenti

[1] Efektivni polumjer kotača definiran je pri punom opterećenu vozila te je nešto manji od neopterećenog polumjera kotača.

dobiveni modelsko prediktivnim upravljanjem služili kao reference. Bilo bi potrebno razviti regulator niže razine koji bi dobivenu referencu momenta na kotačima pretvarao u upravljačke signale poput postotka momenta na aktuatoru za kočenje opisanog u odjeljku 3.2. Dodatne komplikacije javljaju se pri definiranju ograničenja na kontrolne signale. MPC može uzeti u obzir ograničenja na razvijene momente na kotaču, no ne može uzeti u obzir stvarna ograničenja na aktuatore. Ta ograničenja morala bi biti definirana u kontroli na nižoj razini, što nije optimalno jer regulatori poput PID mogu postati nestabilni uz ograničavanje upravljačkog signala. Zbog navedenih problematika, generalni model vozila proširiti će se modelima aktuatora pri čemu će MPC upravljati izravno aktuatorima.

Radi jednostavnosti, preporuka je modelirati dinamiku aktuatora PT1 članom [13].Izgled prijenosne funkcije PT1 člana u Laplaceovoj domeni ima oblik:

$$H_{\mathrm{PT1}}(s) = \frac{K}{Ts+1}, \tag{5.12}$$

gdje:

- K predstavlja koeficijent pojačanja sustava,
- T predstavlja vremensku konstantu sustava.

Pri definiranju aktuatora kočenja, dostupna mjerenja su moment na aktuatoru za kočenje te tlak u sustavu kočenja. Sila na pedali kočnice proporcionalna je momentu kočenja na kotačima prema [14], kao što je i tlak u kočnicama proporcionalan momentu kočenja izravno iz karakteristika diskova kočenja. Korištenjem jednadžbi mogu se izravno povezati mjerene veličine poput tlaka u sustavu kočenja te momenta aktuatora sa silom kočenja kotača. Mjerenje momenta aktuatora može se preslikati na silu pedale kočnice preko polumjera koloture. Sila na pedalu kočnice i sila kočenja kotača tada su linearno povezani jednadžbom. Međutim takvo računanje ima puno međukoraka i ovisnosti, a uostalom veza aktuatora i pedale kočnice nije kruta veza jer se koristi sajla. Ovakav princip ima nekoliko nesigurnosti jer ovisi o pretpostavci da se pedala kočnice aktuira idealno, a u stvarnom slučaju sva sila se troši dodatno na gubitke trenja i skupljanje cilindra EBS-a. Druga metoda je značajno bolja jer se izbjegava cijeli sklop pedale i

mjeri se izravno tlak u sustavu kočenja koji se može vrlo točno preslikati na silu kočenja, stoga će model biti proširen mjerenjem tlaka. Veza momenta aktuatora te tlaka u sustavu kočenja tada se ne modelira nego estimira PT1 članom. Iz [14] uvrštavanjem izraza za tlak u izraz na razvijenu silu na pedalama može se dobiti sljedeća ovisnost sile kočenja o tlaku sustava kočenja:

$$F_{b,f} = 2\mu_{\mathrm{DO}}(\frac{1}{r_{\mathrm{ef}}})A_f R_{\mathrm{m,f}} 0.1p, \tag{5.13}$$

$$F_{b,r} = 2\mu_{\mathrm{DO}}(\frac{1}{r_{\mathrm{ef}}})A_r R_{\mathrm{m,r}} 0.1p. \tag{5.14}$$

$F_{b,f}$ i $F_{b,r}$ označavaju silu kočenja na jednom prednjem kotaču i jednom stražnjem kotaču. Pretpostavka je da su sile na kotačima pojedine osovine jednaki. Koeficijent μ_{DO} predstavlja koeficijent trenja između diska i kočne obloge, $A_{r/f}$ su površine klipova stražnjih i prednjih čeljusti, a $R_{m,r/f}$ je srednji promjer stražnje i prednje obloge. Koeficijent 0.1 predstavlja promjenu mjerne jedinice iz N/mm^2 u bar.

U [14] također je proračunat i maksimalni tlak pri kojemu dolazi do blokiranja kotača. Za prednju osovinu to je 54.68 $bara$, dok je za stražnju 33.78 $bara$. Pošto u autu ne postoji razvijen algoritam protiv proklizavanja pri kočenju (*engl. Anti-Lock Braking System*, ABS), tlak u kočnicama biti će ograničen na 33 $bara$. I vrlo malo proklizavanje pri kočenju može zarotirati auto van upravljivog područja stoga je iznimno bitno da nema proklizavanja pri kočenju. Tlak u prednjem i zadnjem sustavu kočenja su dva zasebna sustava, no balans kočnica u VulpesD je 50 % na prednjim kočnicama i 50 % na zadnjim što znači da se podjednako aktuira prednji i zadnji sustav kočnica. Analiziranjem podataka vožnje, tlakovi u prednjim i zadnjim kočnicama ponašaju se slično do vrlo visokih vrijednosti kočenja od 30 $bara$. Stoga u se u jednadžbi iznad koristi samo jedan tlak radi pojednostavljenja.

Jednadžbe iznad preslikavaju tlak u kočnicama izravno u silu na kotaču bez imalo dinamike. Da se uvede dinamika vladanja, PT1 članom modelirati će se dinamička veza između upravljačkog signala u aktuator te tlaka u kočnicama. To znači da će se vremenska ovisnost od ulaza upravljačkog signala u aktuator do djelovanja aktuatora na pedalu i

povećavanja tlaka modelirati jednostavnim članom. Korištenje linearnog člana je opravdano jer je cijeli sustav linearan. Aktuator kočenja upravljan momentom je potpuno linearni sustav. Hod pedale kočnice je iznimno mali pa su nelinearnosti uzrokovane različitim kutem hvatišta sajle na pedalu zanemarene. Tlačenje fluida u glavnom cilindru je također linearno. Koeficijent pojačanja sustava K je statički član koji skalira ulazne vrijednosti prema izlazu, a vremenska konstanta sustava T označava vrijeme u kojem sustav dosegne približno 63 % konačne vrijednosti pri odzivu na step funkciju. Pretpostaviti će se da je aktuator za kočenje dimenzioniran tako da pri zahtjevu od 100 % momenta razvije tlak u kočnicama od 33 $bara$. Koeficijent pojačanja sustava tada će biti $\frac{33}{100}$ jer se ulaz od 100 preslikava na tlak od 33. Iz dobivenih mjerenja ili iskustveno može se odrediti koliko vremena treba aktuatoru da na step odziv od 100 dovede tlak do 33 $bara$. Takva mjerenja nisu obavljena pa će se pretpostaviti da aktuatoru za kočenje treba jedna sekunda za opisanu akciju. Konstanta sustava u tome slučaju biti će $T = 0.63 \cdot 1s$.

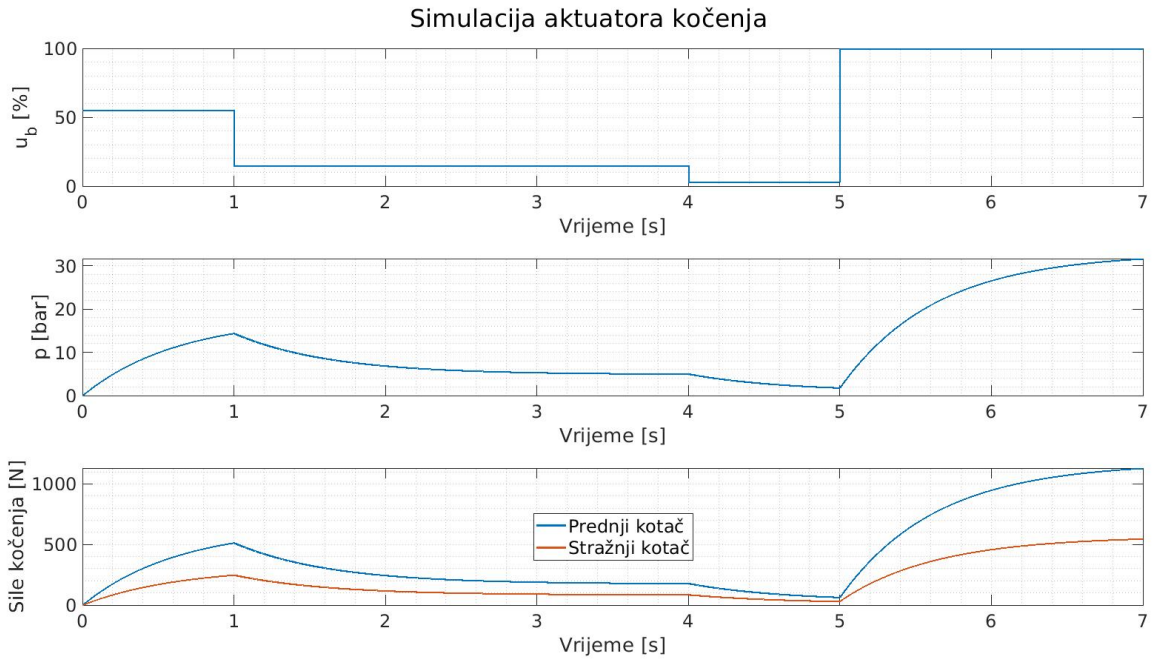
Prijenosna funkcija ima oblik:

$$H_{\text{PT1}}(s) = \frac{\frac{33}{100}}{0.63s + 1}. \tag{5.15}$$

Da bi se koristila dalje pri modeliranju mora se prebaciti u diferencijalni oblik. Inverznom Laplaceovom transformacijom dobije se diferencijalna jednadžba:

$$\dot{p} = -1.5823p + 0.5221u_b, \tag{5.16}$$

gdje u_b predstavlja upravljački signal aktuatora za kočenje. Simulacija otvorenog kruga upravljanja za aktuaciju kočenja napravljenja je u *Simulinku*. Korišteni su nasumični ulazni step signali. Odzivi se mogu vidjeti na slici 5.3.



Slika 5.3. Simulacija otvorenog kruga aktuatora kočenja.

Uz kočenje, na longitudinalnu dinamiku utječe i zadavanje momenta pogonskim kotačima. Pogonski motori kontroliraju se promilom maksimalnog ostvarivog momenta motora. Zbog velike brzine odziva pogonskih motora na moment može se pretpostaviti da moment djeluje trenutno. No to je veliko pojednostavljenje, a pošto u vozilu VulpesD postoji mjerenje struja motora, motor će se modelirati preko struja. Moment na pogonskom motoru određen je strujom i_q te i_d u jednadžbi:

$$T_m = \frac{3}{2} pp \left[\lambda_m i_q + \left(L_d - L_q \right) i_d i_q \right], \tag{5.17}$$

gdje:

- pp predstavlja broj pari polova,
- λ_m je iznos toka permanentnih magneta,
- L_d i L_q induktiviteti u d i q osima motora.

Struja i_q je vrlo mala ili jednaka nuli stoga će se zanemariti struja i_d što rezultira pojednostavljenom linearnom ovisnošću struje i_q te momenta motora:

$$T_m = \frac{3}{2} pp\lambda_m i_q. \tag{5.18}$$

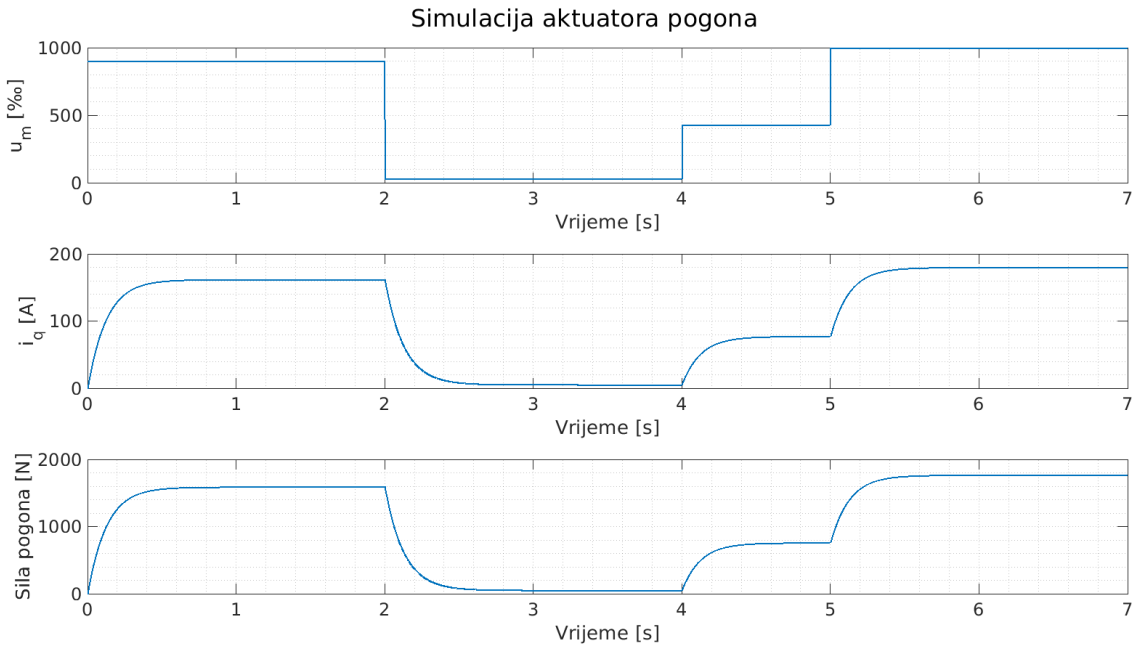
Da bi se dobio moment na kotačima , treba se uzeti u obzir i redukcijski omjer prijenosa motora i. Pošto se u dinamičkom modelu vozila koriste sile na kotačima, dobiveni moment treba se podijeliti sa dinamičkim polumjerom kotača. Konačna jednadžba utjecaja pogonskih motora na longitudinalne sile je:

$$F_m = T_m i \frac{1}{r_{din}} = \frac{3}{2} pp\lambda_m i \frac{1}{r_{din}} i_q. \tag{5.19}$$

Na isti način kao i za kočenje, dobivena je statička relacija longitudinalne sile u ovisnosti o varijabli stanja. Za modeliranje dinamike koristi se PT1 član. Najveća vrijednost struje i_q koju pogonski moment može postići je 180 A uz ulaz od 1000 promila. Vrijednost K PT1 člana tada će biti $\frac{180}{1000}$. Iz mjerenih podataka, i_q struje motora postignu maksimalnu vrijednost na step pobudu u otprilike 200 ms. Stoga vrijednost T PT1 člana iznosi $0.63 \cdot 0.2s$. Inveznom Laplaceovom transformacijom PT1 člana diferencijalna jednadžba dinamike promjene struje i_q izgleda:

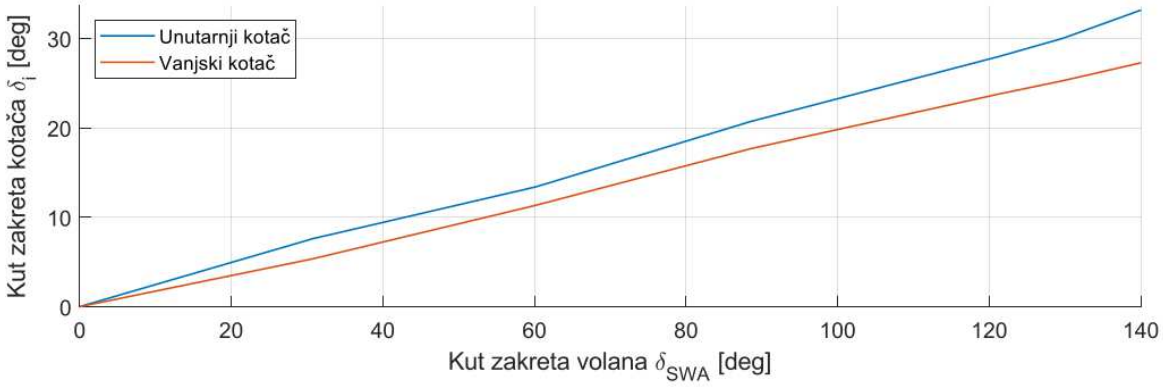
$$\dot{i_q} = -7.9114 i_q + 1.424 u_m. \tag{5.20}$$

U ovoj jednadžbi zanemaren je izgled krivulje momenta pogonskih motora u ovisnosti o brzini rotacije. Motor će u modelu moći razvijati maksimalni moment pri svim vrijednostima brzine rotacije dok u stvarnosti moment nelinearno pada pri velikim okretajima u režimu konstantne snage. Takva je nelinearnost vrlo teška za realizirati u ovome modelu stoga je zanemarena. Simulacija otvorenog kruga sa nasumičnim ulazim signalom prikazana je na slici 5.4.



Slika 5.4. Simulacija otvorenog kruga aktuatora pogona.

Posljednja aktuacija koja ostaje za modelirati je skretanje, tj. kut zakreta kotača. Senzor zakreta volana mjeri kut zakreta volana, a zakret kotača može se dobiti poznavajući geometriju sklopa. Ovisno o kutu volana može se dobiti pojedini kut kotača uz Ackermann geometriju skretanja (slika 5.5.).



Slika 5.5. Ovisnost kuta zakreta kotača o kutu zakreta volana prema Ackermann geometriji [15].

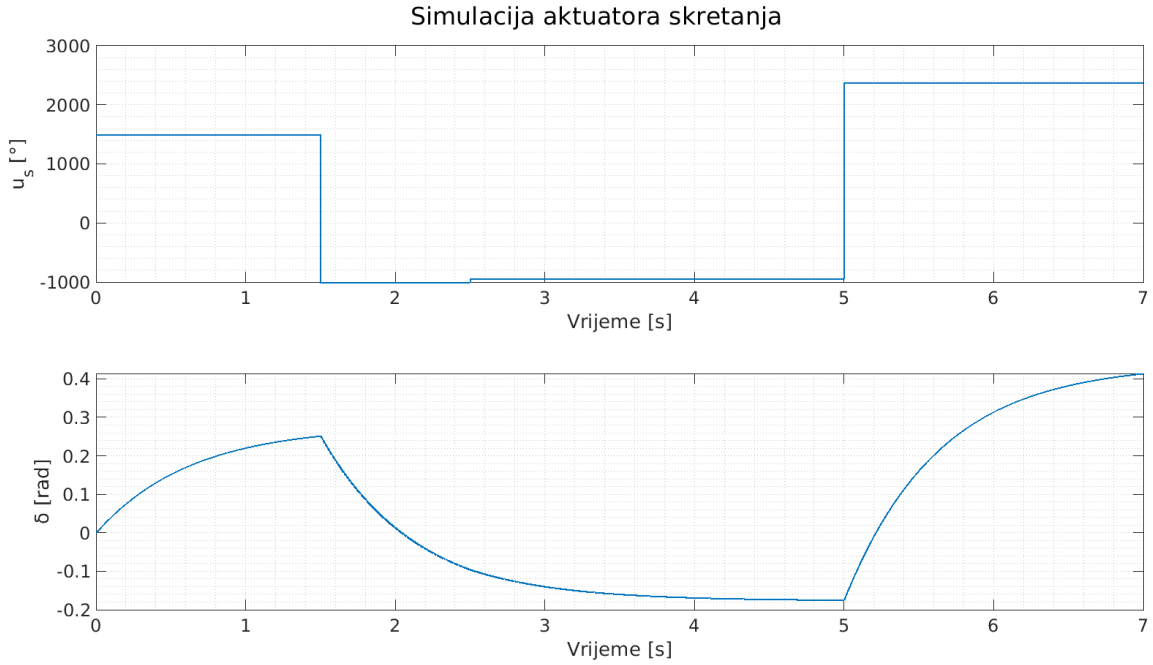
Podaci su dobiveni naprednim tehnikama mjerenja na samome vozilu. Ovisno u kutu volana može se pretpostaviti da su krivulje pravci sa konstantnim nagibom te modelirati kut pojedinog kotača o kutu volana. Problem se javlja što ovisno o kutu skretanja (lijevo ili desno) vanjski kotač može biti desni ili lijevi. Takvi uvjetni izrazi ne mogu se opisati jedinstveno u matematičkom smislu bez uvođenja dodatnih varijabli. Radi toga uzet će se u obzir paralelno skretanje u kojem oba kotača imaju jednak kut pri skretanju. Radi

jednostavnosti zanemareni su i kutevi zakošenja kotača. Također je pretpostavljeno da senzor zakreta kotača mjeri kut kotača umjesto kuta volana. Kut kotača i kut volana povezani su konstantom pri geometriji paralelnog skretanja. Na ovaj način nema potrebe za statičkim izrazima pretvaranja veličina kao pri kočenju i aktuiranju pogonskih motora.

Dinamika uvedena aktuatorom skretanja ponovo će biti defnirana PT1 članom. Aktuator koji se koristi za skretanje isti je kao i aktuator kočenja stoga vremenska konstanta T ostaje jednaka uz pretpostavku da je aktuacija kočenja vremenski jednako duga kao i aktuacija skretanja. Najveći kutevi koje kotač pri skretanju može postići su 0.44 rad, a testiranjima je utvrđeno da aktuator skretanja mora doći na poziciju od 2376° kako bi kotač postigao kut od 0.44 rad. Također je uzeta pretpostavka da se motor rotira tako da pri pozitivnom upravljačkom signalu okreće kotače prema lijevo (pozitivan kut). Stoga konstanta K iznosi $\frac{0.44}{2376}$. Inverznom Laplaceovom transformacijom dobije se diferencijalna jednadžba:

$$\dot{\delta} = -1.5823\delta + 2.9301 \cdot 10^{-4} u_s. \tag{5.21}$$

PT1 modeli za skretanje, simuliran u otvorenom krugu, prikazan je na slici 5.6.



Slika 5.6. Simulacija otvorenog kruga aktuatora skretanja.

Modeliranjem aktuatora, sile u jednadžbama 5.1, 5.2 i 5.3 mogu se dalje raspisati kao:

$$F_{x,rj} = F_m - F_{b,r} - F_{r,r}, \tag{5.22}$$

$$F_{x,fj} = -F_{b,f} - F_{r,f}. \tag{5.23}$$

Pogon djeluje samo na stražnje kotače. Usprkos pojednostavljenju jednakog tlaka u kočnicama, sile na prednjoj i stražnjoj osovini razlikuju se po izvedbi kočnice. Valja naglasiti da su sile kočenja i pogona motora izvedene za pojedini kotač, a sila $F_{r,i}$ je sila otpora kotrljanja definirana za pojedini kotač prednje ili zadnje osovine:

$$F_{r,r} = \frac{1}{2} C_{rr} mg \frac{l_f}{l_f + l_r}, \tag{5.24}$$

$$F_{r,f} = \frac{1}{2} C_{rr} mg \frac{l_r}{l_f + l_r}. \tag{5.25}$$

C_{rr} je konstanta otpora kotrljanja te se definira za svaki kotač. Radi jednostavnosti konstanta je jednaka za sva četiri kotača. Konstante navedene u prijašnjim jednadžbama navedene su u tablici 5.1.

Većina varijabli je pouzdano mjerena, dobivena iz drugih naprednih simulacijskih alata ili uzeta od proizvođača te drugih izvora. Sve konstante koje se odnose na gumu uzete su proizvoljno za potrebe simulacije osim efektivnog polumjera gume, a tok permanentnih magneta je odabran tako da maksimalna struja motora odgovara maksimalnom momentu motora pošto proizvođač nije specifirao vrijednosti.

Nakon potpuno definiranog modela vozila postoji 9 različitih diferencijalnih jednadžbi prvog reda što znači da se u modelu koristi 9 varijabli stanja. Ulaza ima ukupno 3 što odgovara broju aktuacijskih uređaja unutar vozila. Parametar koji nije spomenut je zakrivljenost staze κ. Zakrivljenost staze mijenja se u svakoj točki prijeđenog puta te se ne može matematički modelirati jer staza ne prati pravilan uzorak. Uglavnom je nasumična u disciplinama poput *autocross* i *trackdrive*. U disciplini poput *acceleration* zakrivljenost

Tablica 5.1. Vrijednosti konstanti korištenih u simulacijama.

Simbol	Opis	Vrijednost
Općenito		
m	Masa vozila bez vozača	245 kg
ρ	Gustoća zraka	1.213 kg/m^3
g	Gravitacijska konstanta	9.807 m/s^2
J_{zz}	Moment inercije vozila oko z osi	163.599 kg/m^2
tw_f, tw_r	Širina prednje/stražnje osovine	1.274 m, 1.240 m
l_f, l_r	Udaljenost prednje/stražnje osovine od centra mase	0.842 m, 0.689 m
Guma		
μ_{ij}	Koeficijent trenja između podloge i gume	0.9
B_i, C_i, D_i	Koeficijenti gume iz "magične" formule	10, 1.5, 1
r_{ef}	Efektivni polumjer kotača	0.22 m
C_{rr}	Sila otpora kotrljanja	0.017
Aerodinamika		
A	Prednja površina vozila	1.21 m^2
C_d	koeficijent otpora zraka	1.39
$C_{l,r}, C_{l,f}$	Koeficijent uzgona	1.55, 1.6848
Pogon		
pp	Broj pari polova pogonskog motora	6
λ_m	Tok permanentnih magneta pogonskog motora	0.02 Vs
i	Prijenosni omjer sa motora na kotače	8
Kočenje		
μ_{DO}	Koeficijent trenja između diska i obloge	0.5
A_r, A_f	Površine klipova stražnje/prednje čeljusti	490.87 mm^2, 981.75 mm^2
$R_{m,r}, R_{m,f}$	Srednji promjer stražnje/prednje obloge	0.0775 m, 0.08 m

staze je 0, a u disciplini *skidpad* zakrivljenost je konstantna vrijednost koja odgovara polumjeru centralne putanje kroz stazu. Pošto je zakrivljenost staze nepredvidiva, ponaša

se kao smetnja te će u model vozila biti uključena kao parametar smetnje.

Cjelovit pregled varijabli stanja, ulaza te smetnji uz definirana stvarna ograničenja koja će biti korištena u MPC-u prikazan je u tablici 5.1.

Tablica 5.2. Varijable stanja, ulazi te smetnje u modelu vozila uz ograničenja.

Simbol	Opis	Ograničenja
Varijable stanja		
s	Prijeđeni put vozila	$[0, \inf]\ m$
n	Udaljenost vozila od referentne putanje	$[-N_r + \frac{tw_f}{2}, N_l - \frac{tw_f}{2}]\ m$
ξ	Kut gledanja vozila naspram referentne putanje	$\left[-\frac{\pi}{3}, \frac{\pi}{3}\right]\ rad$
v	Brzina vozila	$[0, 20]\ m/s$
β	Bočni kut klizanja vozila	$[-0.26, 0.26]\ rad$
w_z	Kutna brzina vozila oko z osi	$\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]\ rad/s$
δ	Kut skretanja prednjih kotača	$[-0.44, 0.44]\ rad$
i_q	Struja pogonskih motora u q osi	$[0, 180]\ A$
p	Tlak u sustavu kočenja	$[0, 33]\ bar$
Ulazi		
u_s	Zahtjev za poziciju aktuatora skretanja	$[-2376, 2376]^\circ$
u_m	Zahtjev za promilima momenta pogonskog motora	$[0, 1000]\,‰$
u_b	Zahtjev za postotak momenta aktuatora za kočenje	$[0, 100]\,\%$
Smetnje		
κ	Zakrivljenost staze po referentnoj putanji	$\left[-\frac{1}{5}, \frac{1}{5}\right]\ m$

Neka ograničenja na ulaze i varijable stanja su potrebna jer su ograničenja mehanička, elektronička ili programska. Kršenje takvih ograničenja uzrokovalo bi mehaničke kvarove, okidanje sigurnosnih sustava ili instanciranje greške pri vožnji vozila. Ograničenje za udaljenost vozila od referentne putanje mijenja se dinamično ovisno o putanji. Ako optimalna putanja prolazi blizu čunjeva sa lijeve strane putanje, vozilo ne smije ići više u lijevo jer će srušiti čunjeve dok će na desnoj strani biti više prostora. U ograničenje

se mora uzeti i polovica udaljenosti prednje osovine jer širina staze ne uzima u obzir širinu vozila. Optimizator trajektorije računa definirane širine staza kroz putanju, no radi jednostavnosti simulacija dinamička ograničenja na udaljenost od referentne putanje n nisu uzeta u obzir te su zamijenjena statičkim ograničenjima. Ograničenja za ξ, β i w_z uzete su prema maksimalnim vrijednostima koje je optimizator trajektorije izračunao. Na natjecanjima, u disciplinama *trackdrive* i *autocross* vozila obično nikada ne prelaze brzine od 20 m/s jer se tako visoke brzine smatraju granicama upravljivosti. Prema pravilniku vanjski zavoj smije biti minimalno 9 metara polumjera. Uz minimalnu širinu staze od 3 metra te činjenicu da optimalna putanja može odabrati još oštriji ulaz blizu unutarnjeg zavoja, za minimalni polumjer uzeto je 5 metara koji se preslikava na maksimalnu zakrivljenost zavoja od $\frac{1}{5}$ m. Neke veličine ne moraju imati ograničenja, no zbog boljeg definiranja problematike pri izradi MPC-a poželjno je imati što više ograničenja koja nisu redundantna.

5.2. Linearizacija modela

Neke od metoda linearizacije korištene za adaptivni ili LTV MPC su:

- LPV sustav - predefinirane linearizirane matrice u proizvoljnim radnim točkama koje se učitavaju ovisno o blizini radne točke.
- Analitička linearizacija - dobije se generalni linearizirani izraz za model. Linearizirani model analitičkim postupkom definiran je za svaku moguću radnu točku.

Adaptivni MPC lineariziran LPV sustavom razrađen je u [6]. LPV sustav ne mora imati matematički model, a matrice se estimiraju u okolini radne točke. Model može biti bilo kakav uključujući uvjetne izraze. Analitički lineariziran model uključuje matematički zapis modela. Prednost analitičkog modela je ta što se dobije generalni izraz za linearizirani model u kojeg se trebaju uvrstiti radne točke. LPV model izvodi se jako brzo jer su vrijednosti matrica predefinirane. Nedostatak je što definiranje većeg broja radnih točaka zahtijeva više memorije što čini algoritam nepogodnim za sustave koji imaju velik raspon radnih točaka. Analitička linearizacija za razliku ima samo jedan izraz u kojeg se uvrštavaju radne točke i iznimno je brza pri izvođenju jer koristi samo uvrštavanje vrijednosti. Analitička linearizacija je najtočnija, no u smislu MPC algoritama ima nedostatak

jer su česti kompliciranі procesi koji nemaju matematički model ili su im matematički modeli nepouzdani. Za model vozila definiran je matematički model u prethodom poglavlju stoga će u nastavku model biti analitički lineariziran.

Jednadžbe opisane u prethodnom poglavlju su skup nelinearnih diferencijalnih jednadžbi prvog reda. Da bi zapisali sustav u linearnom vremenski nepromjenjivom kontinuiranom obliku $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} + \mathbf{Ed}$ potrebno je linearizirati sustav. Vektor $\mathbf{d}$ predstavlja smetnju u sustavu. Nelinearni sustav opisan je jednadžbom $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{d})$ gdje je f nelinearna funkcija koja se sastoji od varijabli stanja i ulaza u model kao u prethodnom poglavlju. Matrica $\mathbf{x}$ mora biti zapisana u vektor stupcu:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}. \tag{5.26}$$

Varijable stanja i ulazi indeksirani su jednakim redom kao što su navedeni u tablici 5.1. Za računanje matrica linearizirаnih oblika $\mathbf{A}_l$, $\mathbf{B}_l$ i $\mathbf{E}_l$ svaka derivacija jednadžbe stanja mora biti zapisana kao funkcija svih ostalih varijabli stanja i ulaza (5.27). Diferencijalna jednadžba mora biti prvog reda, ako nije može se razložiti na više diferencijalnih jednadžbi stanja prvog reda uvođenjem nove varijable stanja. Jednadžbe u prethodnom poglavlju zadovoljavaju ovaj oblik stoga nije potrebno modificirati jednadžbe.

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \dots, x_n,\ u_1, u_2, \dots, u_n,\ d_1, d_2, \dots, d_n,), \\ \dot{x}_2 &= f_2(x_1, x_2, \dots, x_n,\ u_1, u_2, \dots, u_n,\ d_1, d_2, \dots, d_n,), \\ &\vdots \\ \dot{x}_n &= f_n(x_1, x_2, \dots, x_n,\ u_1, u_2, \dots, u_n,\ d_1, d_2, \dots, d_n,). \end{aligned} \tag{5.27}$$

Za dobivanje linearizirаnih oblika matrica koristi se jakobijan matrica:

$$\mathbf{A}_l = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{d})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_r} = \left[\begin{array}{cccc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{array}\right]_{\mathbf{x}=\mathbf{x}_r}. \tag{5.28}$$

Jakobijan predstavlja parcijalnu derivaciju funkcija po svim varijablama stanja. Dobivena matrica defnirana je za sva stanja nelinearnog sustava, a da se dobije trenutna linearna matrica $\mathbf{A}_l$ treba se u jakobijan uvrstiti trenutna radna točka. Izrazi analogno vrijede za matrice $\mathbf{B}_l$ i $\mathbf{E}_l$:

$$\mathbf{B}_l = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{d})}{\partial \mathbf{u}}\right|_{\mathbf{u}=\mathbf{u}_r} = \left[\begin{array}{cccc} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \cdots & \frac{\partial f_n}{\partial u_m} \end{array}\right]_{\mathbf{u}=\mathbf{u}_r}, \tag{5.29}$$

$$\mathbf{E}_l = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u},\mathbf{d})}{\partial \mathbf{d}}\right|_{\mathbf{d}=\mathbf{d}_r} = \left[\begin{array}{cccc} \frac{\partial f_1}{\partial d_1} & \frac{\partial f_1}{\partial d_2} & \cdots & \frac{\partial f_1}{\partial d_p} \\ \frac{\partial f_2}{\partial d_1} & \frac{\partial f_2}{\partial d_2} & \cdots & \frac{\partial f_2}{\partial d_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial d_1} & \frac{\partial f_n}{\partial d_2} & \cdots & \frac{\partial f_n}{\partial d_p} \end{array}\right]_{\mathbf{d}=\mathbf{d}_r}. \tag{5.30}$$

Indeks r u izrazima $\mathbf{u}_r$, $\mathbf{x}_r$ i $\mathbf{d}_r$ predstavlja trenutnu radnu točku varijabli stanja, ulaza i smetnji. Sustav opisan u prethodnom poglavlju sastoji se od:

- 9 varijabli stanja ($n = 9$),
- 3 ulaza ($m = 3$),

- jedne smetnje ($d = 1$),

što defnira matrice:

- $A \in \mathbb{R}^{9\times 9}$,
- $B \in \mathbb{R}^{9\times 3}$,
- $E \in \mathbb{R}^{9\times 1}$.

Nakon računanja jakobijana, matrica $\mathbf{A}_l$ za definirani model vozila poprima oblik:

$$\mathbf{A}_l = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} & a_{15} & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{23} & a_{24} & a_{25} & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & a_{35} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} & a_{49} \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} & a_{59} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} & 0 & a_{69} \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.58 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -7.91 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.58 \end{bmatrix}. \tag{5.31}$$

Može se vidjeti da je su neke vrijednosti u modelu linearne i ne mijenjaju se kroz vrijeme, dok vrijednosti a_{ij} sadrže parcijalne derivacije i mijenjaju se u svakoj radnoj točki. Znanje o konačnom izgledu jakobijana matrica uvelike pomaže pri sastavljanju problema za LTV MPC. Jednadžbe modela su vrlo velike jer se za pojedinu varijablu stanja sumiraju vrijednosti koje djeluju na sva četiri kotača. Na primjer, najduži izraz a_{64}, koji predstavlja parcijalnu derivaciju ovisnosti kutne brzine vozila o brzini vozila ima oko 3100 znakova. To znači da je jako puno proračuna na kraju sadržano samo u jedan skalar matrice $\mathbf{A}_l$. Radi toga, nužno je kroz cijelu implementaciju koristiti *double*

preciznosti varijabli tako da se ne gubi na preciznosti izračuna. Matrica $\mathbf{B}_l$ je konstantna zbog linearnog modeliranja ulaza aktuatora te se ne mijenja kroz vrijeme:

$$\mathbf{B}_l = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 2.93 \times 10^{-4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.42 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.52 \end{bmatrix}^T, \tag{5.32}$$

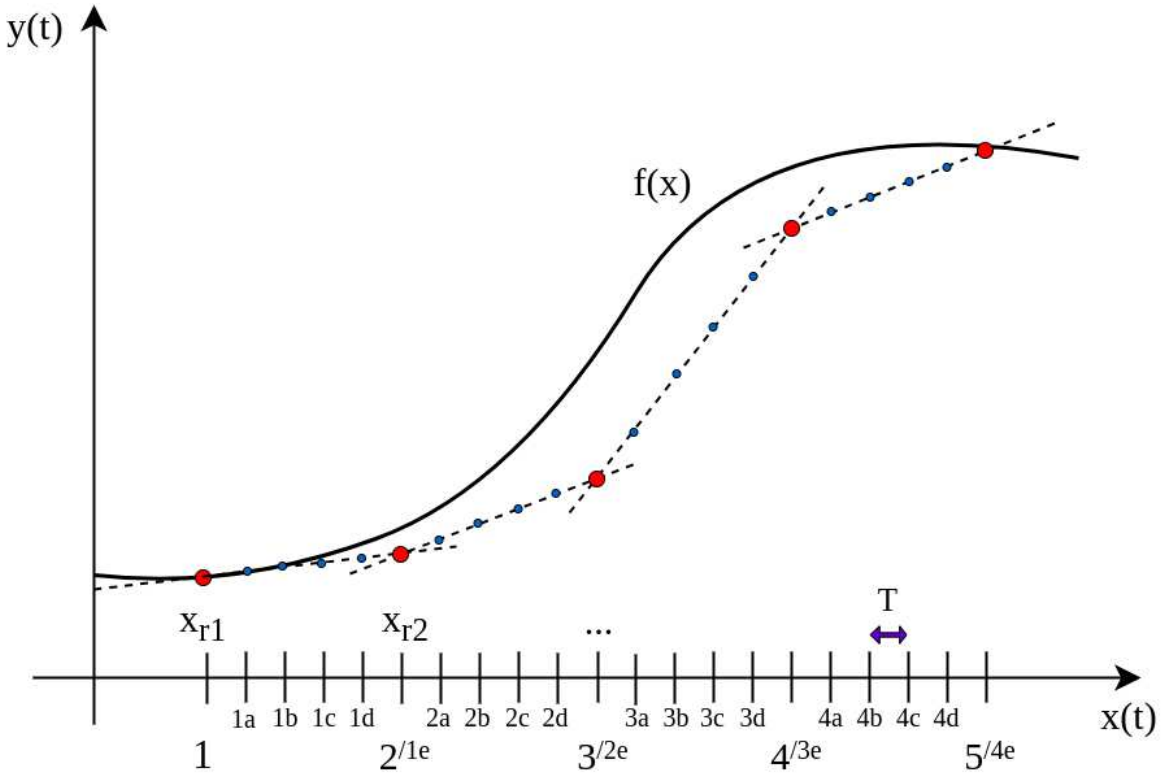
a matrica $\mathbf{E}$ ima oblik:

$$\mathbf{E}_l = \begin{bmatrix} e_1 & 0 & e_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T. \tag{5.33}$$

Brojevi su zaokruženi na dvije decimale radi jednostavnijeg prikaza, no u simulaciji su precizni do 15 decimala. Drugi dio sustava linearnih jednadžbi $\mathbf{y} = \mathbf{Cx} + \mathbf{Du}$ nije uzet u obzir jer su varijable stanja ujedno i izlazi iz modela.

Linearizirani model vozila korišten u adaptivnim regulatorima poput adaptivnog MPC-a pri svakom pozivanju regulatora koristi novu lineariziranu točku. No unutar predikcije koristi isti linearizirani model. Primjer u jednoj dimenziji prikazan je slikom 5.7.

Na slici je proizvoljna jednodimenzionalna nelinearna funkcija $f(x)$. Crvenim točkama označeni su počeci predikcije u kojem se događa linearizacija modela u trenutnoj radnoj točki $f(x_r)$. U jednoj dimenziji jakobijan predstavlja derivaciju funkcije te su iscrtanim linijama označene tangente na krivulju po kojima se linearizirani model kreće. Slika je vrlo pojednostavljena i pretpostavka je da se sustav kreće jednoliko u pozitivnome smjeru. U primjeru se radi predikcija od 5 koraka nakon svake linearizacije. Plave točke označuju korake predikcije, a crvene početak predikcije. Može se vidjeti da linearizirani graf ne prati najbolje nelinearnu funkciju. Točnost naspram nelinearne funkcije ovisi o vremenu diskretizacije te koracima predikcije. Što je vrijeme diskretizacije niže raditi će se manji koraci po tangenti krivulje te će radne točke biti gušće raspoređene po krivulji. Naravno, mana smanjivanja vremena diskretizacije pri jednakom predikcijskom horizontu je kraće vrijeme predikcije. Zbog ovoga načina rada, adaptivni MPC nije ide-



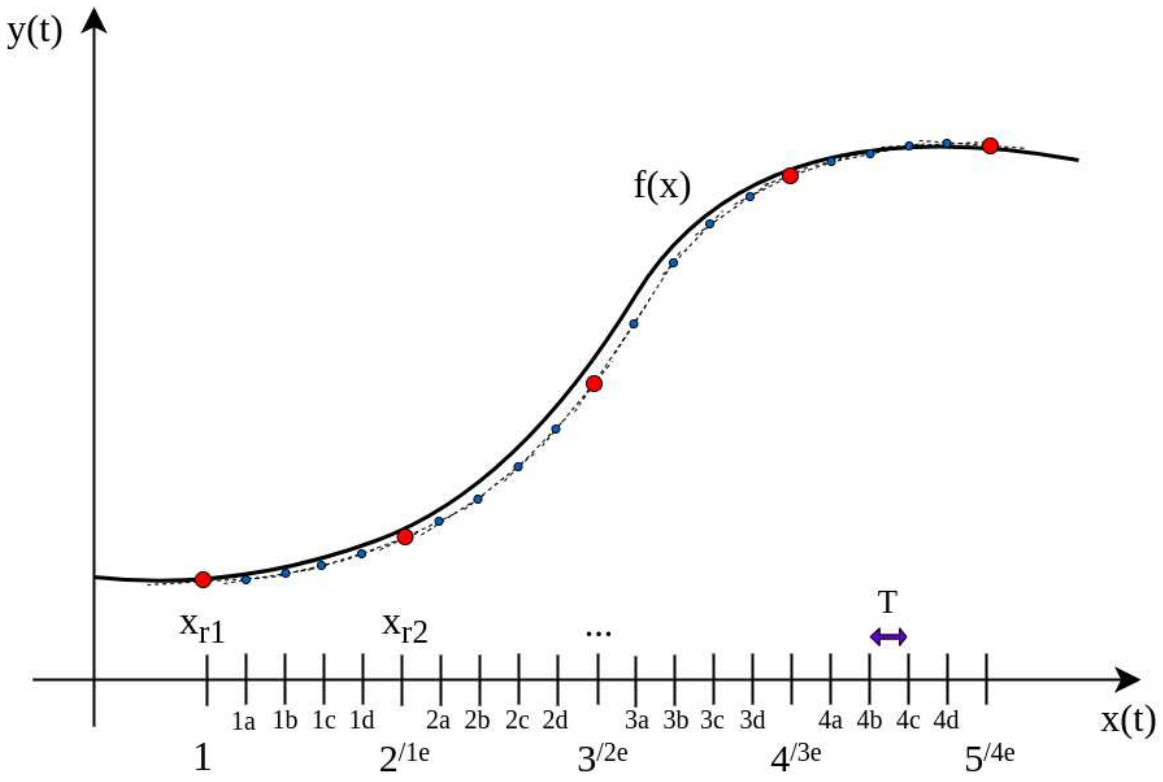
Slika 5.7. Primjer rada adaptivnog MPC-a.

alan za vrlo nelinearne funkcije kojima se radna točka ili derivacija radne točke vrlo brzo mijenja. Linearizirani sustav oko radne točke u sustavu stanja zapisan je jednadžbom:

$$\dot{\mathbf{x}} \approx \mathbf{A}_l(\mathbf{x} - \mathbf{x}_r) + \mathbf{B}_l(\mathbf{u} - \mathbf{u}_r) + \mathbf{E}_l(\mathbf{d} - \mathbf{d}_r). \tag{5.34}$$

Za LTV MPC vrijedi da se matrica $\mathbf{A}_l$ mijenja ne samo pri pozivu MPC-a nego kroz svaki korak predikcije. Primjer kako bi LTV predikcija funkcionirala prikazano je na slici 5.8.

Može se vidjeti da je aproksimacija puno točnija. Nedostatak je što se u koraku predikcije MPC-a moraju unaprijed zadati matrice $\mathbf{A}_l$ za svaki korak predikcije koje je nemoguće točno znati jer se buduća optimalna stanja tek trebaju izračunati.



Slika 5.8. Primjer rada LTV MPC-a.

5.3. Diskretizacija modela

Matrice dobivene linearizacijom izračunate su u kontinuiranoj domeni. Za potrebe MPC-a potrebno ih je diskretizirati. Neke jednostavne metode korištene pri diskretizaciji linearnih sustava su:

- Unaprijedna Eulerova diskretizacija,
- Tustin diskretizacija,
- Egzaktna diskretizacija,
- Runge-Kutta diskretizacija drugog ili četvrtog reda.

Unaprijedna Eulerova diskretizacija je najjednostavnija i intuitivna. Postoji i unazadna Eulerova diskretizacija koja je za razliku od unaprijedne zapisana u implicitnom obliku te zahtijeva rješavanje sustava jednadžbi za rješenje. Kompliciranija je za izvršiti, ali daje stabilnije rješenje. Unaprijedna Eulerova diskretizacija zapisana je u eksplicitnom obliku i može se dobiti aproksimacijom jednadžbe prostora stanja $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{E}\mathbf{d}(t)$. Derivacija se aproksimira vremenom diskretizacije T: $t_{k+1} = t_k + T$.

Derivacija tada postaje:

$$\dot{\mathbf{x}}(t_k) \approx \frac{\mathbf{x}_{k+1} - \mathbf{x}_\mathbf{k}}{T}. \tag{5.35}$$

Uvrštavanjem u jednadžbu prostora stanja te preslagivanjem jednadžbe i pomicanjem jednog diskretnog koraka dobije se konačan izraz u diskretnoj domeni:

$$\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{T} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{E}\mathbf{d}_k, \tag{5.36}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + T \cdot (\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{E}\mathbf{d}_k), \tag{5.37}$$

$$\mathbf{x}_k = \mathbf{x}_{k-1} + T \cdot (\mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{E}\mathbf{d}_{k-1}). \tag{5.38}$$

Unazadna Eulerova metoda koristi unazadnu aproksimaciju derivacije:

$$\dot{\mathbf{x}}(t_k) \approx \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{T}. \tag{5.39}$$

Uvrštavanjem u jednadžbu varijabli stanja i preslagivanjem dobije se:

$$\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{T} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{E}\mathbf{d}_k, \tag{5.40}$$

$$(\mathbf{I} - T\mathbf{A})\mathbf{x}_k = \mathbf{x}_{k-1} + T\mathbf{B}\mathbf{u}_k + T\mathbf{E}\mathbf{d}_k, \tag{5.41}$$

što predstavlja jednadžbu u implicitnom obliku koju treba riješiti da se dobije $\mathbf{x}_k$. Dijeljenjem izrazom $(\mathbf{I} - T\mathbf{A})$ s lijeva dobije se konačni izraz diskretnog oblika unazadne Eulerove diskretizacije:

$$\mathbf{x}_k = \underbrace{(\mathbf{I} - T\mathbf{A})^{-1}}_{\mathbf{A}_d}\mathbf{x}_{k-1} + \underbrace{T(\mathbf{I} - T\mathbf{A})^{-1}\mathbf{B}}_{\mathbf{B}_d}\mathbf{u}_k + \underbrace{T(\mathbf{I} - T\mathbf{A})^{-1}\mathbf{E}}_{\mathbf{E}_d}\mathbf{d}_k. \tag{5.42}$$

Tustin diskretizacija ili bilinearna diskretizacija je točnija od Eulerovih diskretizacija. Euler diskretizacije su aproksimacije Taylorovog niza prvog reda, dok je bilinearna transformacija Padé aproksimacija prvog reda. Tustin diskretizacija u prostoru stanja može se dobiti tako da se jednadžba prebaci u Laplace domenu $s\mathbf{X}(s) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s) + \mathbf{E}\mathbf{D}(s)$ te se Laplaceova varijabla s zamijeni aproksimacijom: $s \approx \frac{2}{T} \cdot \frac{1-z^{-1}}{1+z^{-1}}$. Ovo će ponovo biti zapis u implicitnom obliku kojeg treba riješiti.

Egzaktna diskretizacija diskretizira sustav točno bez ikakve greške. Ne koristi aproksimacije te izrazi za diskretne matrice imaju oblik:

$$\begin{aligned} \mathbf{A}_d &= e^{\mathbf{A}T}, \\ \mathbf{B}_d &= \left(\int_0^T e^{\mathbf{A}\tau} \, d\tau \right) \mathbf{B}, \\ \mathbf{E}_d &= \left(\int_0^T e^{\mathbf{A}\tau} \, d\tau \right) \mathbf{E}. \end{aligned} \tag{5.43}$$

Nedostatak je što za rješavanje izraza treba koristiti daljnje aproksimacije reda, inverz matrica ili druge tehnike za dobivanje rješenja. Runge-Kutta metoda je vrlo precizna metoda za rješavanje diferencijalnih jednadžbi koja se široko koristi. Zapisana je u eksplicitnom obliku te predstavlja Taylor aproksimaciju drugog ili četvrtog reda ovisno o željenoj preciznosti. Fokus će biti na aproksimaciji četvrtog reda jer je vrlo točna a jednostavna. Ideja je da se izračunaju četiri nagiba:

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{E}\mathbf{d}_{k-1}, \\ \mathbf{k}_2 &= \mathbf{A}\left(\mathbf{x}_{k-1} + \frac{\mathbf{k}_1}{2}\right) + \mathbf{B}\mathbf{u}\left(t_{k-1} + \frac{T}{2}\right) + \mathbf{E}\mathbf{d}\left(t_{k-1} + \frac{T}{2}\right), \\ \mathbf{k}_3 &= \mathbf{A}\left(\mathbf{x}_{k-1} + \frac{\mathbf{k}_2}{2}\right) + \mathbf{B}\mathbf{u}\left(t_{k-1} + \frac{T}{2}\right) + \mathbf{E}\mathbf{d}\left(t_{k-1} + \frac{T}{2}\right), \\ \mathbf{k}_4 &= \mathbf{A}\left(\mathbf{x}_{k-1} + \mathbf{k}_3\right) + \mathbf{B}\mathbf{u}(t_{k-1} + T) + \mathbf{E}\mathbf{d}(t_{k-1} + T). \end{aligned} \tag{5.44}$$

Nakon računanja nagiba, konačno rješenje je otežana suma nagiba:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \frac{1}{6}T(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \tag{5.45}$$

Za slučaj MPC-a ne znaju se vrijednosti ulaza na pola diskretizacijskog koraka, a vrijednost smetnje ima jednaku vrijednost na pola diskretizacijskog koraka kao na početku i kraju. Zbog toga se za računanje svih nagiba koristi izraz $\mathbf{u}_{k-1}$ i $\mathbf{d}_{k-1}$.

Računanje diskretizacija zadanih u implicitnom obliku generalno sadrži računanje inverza matrice. Dobivena matrica modela vozila je dimenzija 9×9 te bi pri implicitnim metodama zahtijevalo inverz matrice pri svakom pozivu MPC-a. Adaptivni MPC koristi diskretizirani model oko trenutne radne točke za predikciju pa jedan inverz matrice za jedno pozivanje MPC-a nije problem. No kod LTV MPC-a, diskretizirana matrica $\mathbf{A}_d$ mijenja se u svakom koraku predikcijskog horizonta. Ako bi predikcijski horizont bio velik (50) tada bi se pri svakom pozivu MPC-a inverz matrice morao izračunati 50 puta. Takvi računi mogu značajno usporiti izvršavanje van same optimizacije MPC-a. Eksplicitni oblici ne koriste zahtjevna računanja. Preferirane metode u ovome radu su stoga unaprijedna Eulerova metoda te Runge-Kutta metoda četvrtog reda.

Jednadžba 5.34 djeluje oko radne točke i ravnotežno stanje joj je upravo vrijednost radne točke. Za potrebe boljeg praćenja nelinearnosti cilj je ne samo da se matrica $\mathbf{A}_l$ mijenja svaki korak nego i radna točka $\mathbf{x}_r$. Simulacijama se pokazalo da je predikcija puno bolja ako se za ažuriranje varijabli stanja ne koristi samo trenutno stanje nego i prošlo. Uzmimo za primjer unaprijednu Eulerovu diskretizaciju te trenutke $1a$, $1b$ te $1c$ sa slike 5.8. Da bi se dobilo stanje $\mathbf{x}_{1c}$ treba se znati stanje $\mathbf{x}_{1b}$ prema Eulerovoj jednadžbi 5.38:

$$\text{I.} \quad \mathbf{x}_{1c} = \mathbf{x}_{1b} + T \cdot (\mathbf{A}_{l,b}\mathbf{x}_{1b} + \mathbf{B}_{l,b}\mathbf{u}_{1b} + \mathbf{E}_{l,b}\mathbf{d}_{1b}). \tag{5.46}$$

Stanje $\mathbf{x}_{1b}$ može se dobiti na jednak način:

$$\text{II.} \quad \mathbf{x}_{1b} = \mathbf{x}_{1a} + T \cdot (\mathbf{A}_{l,a}\mathbf{x}_{1a} + \mathbf{B}_{l,a}\mathbf{u}_{1a} + \mathbf{E}_{l,a}\mathbf{d}_{1a}). \tag{5.47}$$

Uz pretpostavku da su parovi matrica $\mathbf{A}_{l,a}$ i $\mathbf{A}_{l,b}$, $\mathbf{B}_{l,a}$ i $\mathbf{B}_{l,b}$ te $\mathbf{E}_{l,a}$ i $\mathbf{E}_{l,b}$ vrlo slični u susjednim koracima, u daljnjem izvodu neće se razlikovati. Uzeti će se matrica linearizirana u trenutnom koraku, tj. $\mathbf{A}_{l,b}$, $\mathbf{B}_{l,b}$ i $\mathbf{E}_{l,b}$. Oduzimanjem jednadžbe II. od I. te sređi-

vanjem dobije se jednadžba predikcije unaprijednog Eulera koja u obzir uzima i prošlo stanje:

$$\mathbf{x}_{1c} = 2 \cdot \mathbf{x}_{1b} + T \cdot \left(\mathbf{A}_{l,b} \left(\mathbf{x}_{1b} - \mathbf{x}_{1a} \right) + \mathbf{B}_{l,b} \left(\mathbf{u}_{1b} - \mathbf{u}_{1a} \right) + \mathbf{E}_{l,b} \left(\mathbf{d}_{1b} - \mathbf{d}_{1a} \right) \right) - \mathbf{x}_{1a}. \quad (5.48)$$

Ili u općenitom zapisu:

$$\mathbf{x}_{k+1} = 2 \cdot \mathbf{x}_{k} + T \cdot \left(\mathbf{A}_{l,k} \left(\mathbf{x}_{k} - \mathbf{x}_{k-1} \right) + \mathbf{B}_{l,k} \left(\mathbf{u}_{k} - \mathbf{u}_{k-1} \right) + \mathbf{E}_{l,k} \left(\mathbf{d}_{k} - \mathbf{d}_{k-1} \right) \right) - \mathbf{x}_{k-1}. \quad (5.49)$$

Jednakim postupkom može se dobiti jednadžba za Runge-Kutta diskretizaciju četvrtog reda koja uzima u obzir trenutno i prošlo stanje:

$$\begin{aligned} \mathbf{x}_{k+1} = 2 \cdot \mathbf{x}_{k} + T \frac{1}{6} \cdot \Big(& \left(\mathbf{k}_{1,k} - \mathbf{k}_{1,k-1} \right) + 2 \cdot \left(\mathbf{k}_{2,k} - \mathbf{k}_{2,k-1} \right) + 2 \cdot \left(\mathbf{k}_{3,k} - \mathbf{k}_{3,k-1} \right) \\ & + \left(\mathbf{k}_{4,k} - \mathbf{k}_{4,k-1} \right) \Big) - \mathbf{x}_{k-1}. \end{aligned} \quad (5.50)$$

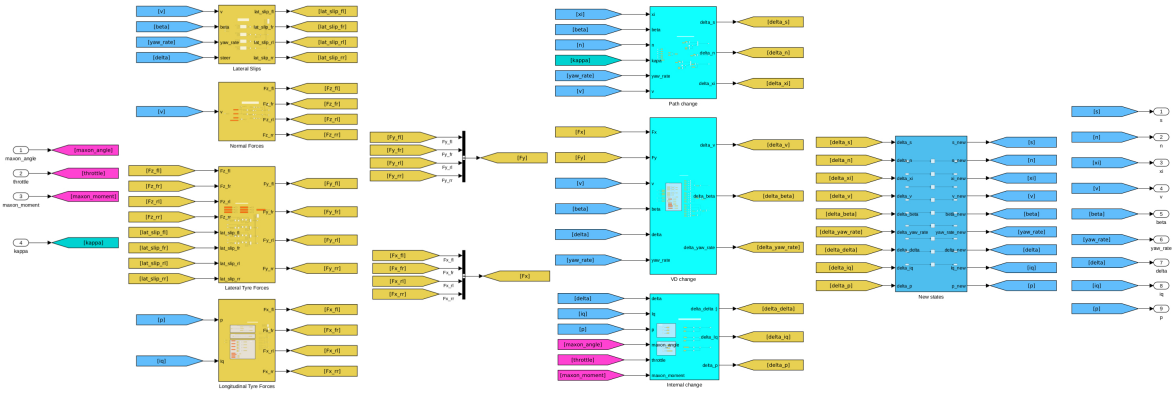
Generalna Runge-Kutta metoda četvrtog reda za rješavanje nelinearnih sustava diferencijalnih jednadžbi za nagibe koristi izraze:

$$\begin{aligned} \mathbf{k}_1 &= f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k), \\ \mathbf{k}_2 &= f\left(\mathbf{x}_k + \frac{T}{2} \mathbf{k}_1, \mathbf{u}_k, \mathbf{d}_k \right), \\ \mathbf{k}_3 &= f\left(\mathbf{x}_k + \frac{T}{2} \mathbf{k}_2, \mathbf{u}_k, \mathbf{d}_k \right), \\ \mathbf{k}_4 &= f\left(\mathbf{x}_k + T \mathbf{k}_3, \mathbf{u}_k, \mathbf{d}_k \right). \end{aligned} \quad (5.51)$$

5.4. Testiranje modela

Potpuni nelinearni model vozila opisan u sekciji 5.1. ostvaren je blokovskom algebrom u *Simulink* okruženju. U ovome poglavlju analizirati će model vozila i usporediti sa li-

neariziranim modelom da se ustvrde karakteristike korištenih metoda diskretizacije.



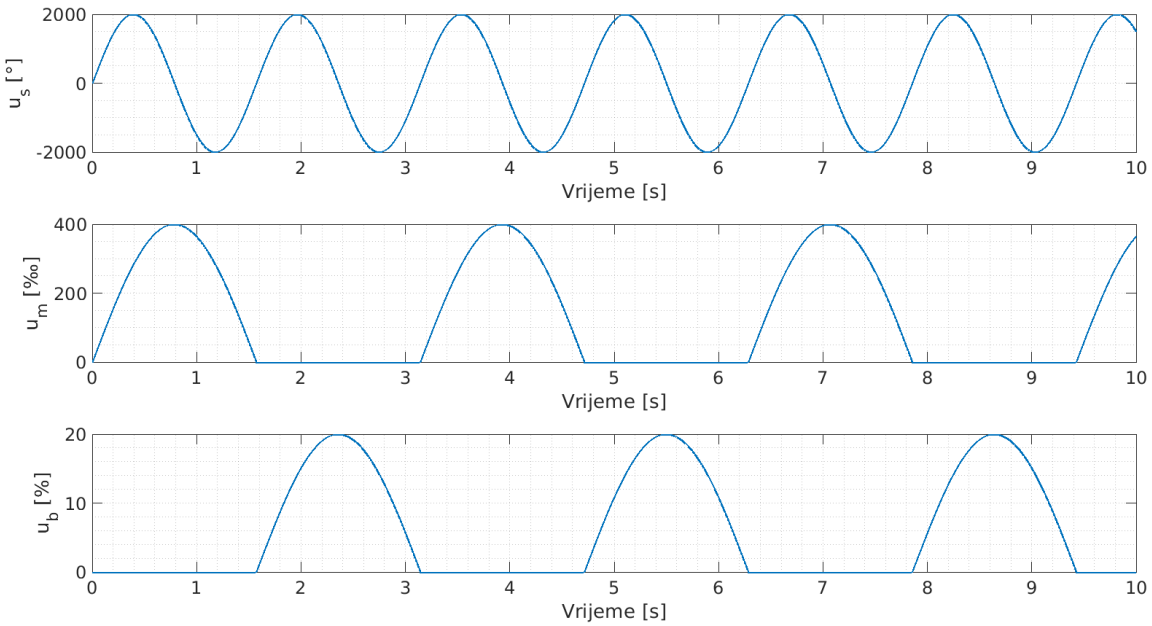
Slika 5.9. Model vozila ostvaren u *Simulink* okruženju.

Manevar koji će se često koristiti za validiranje modela i algoritama jest periodično ubrzavanje i usporavanje uz istovremeno periodično skretanje (slika 5.10.). Za takav manevar koristiti će se izraz *mješoviti manevar* u nastavku rada. Na ovaj način pobuđene su sve varijable stanja i ulazi. Također pri takvom testu cilj je da nelinearne funkcije što više dođu do izražaja. Trigonometrijske funkcije mogu se aproksimirati linearno pri jako malim kutevima, stoga će u testu svi kutevi dosezati visoke vrijednosti. Na isti način kvadratni članovi imaju jaki nelinearni utjecaj na velikim vrijednostima, stoga će se kvadratne varijable poput brzine vozila držati na visokim vrijednostima. Model ima 9 varijabli stanja pa radi jednostavnosti prikaza na grafovima neće biti prikazane sve varijable stanja. Kroz simulacije utvrđeno je da su vrlo nelinearne varijable koje je teško pratiti prijeđeni put (s), udaljenost od referentne putanje (n) te brzina vozila (v), stoga će u nastavku uglavnom biti uspoređivane te vrijednosti.

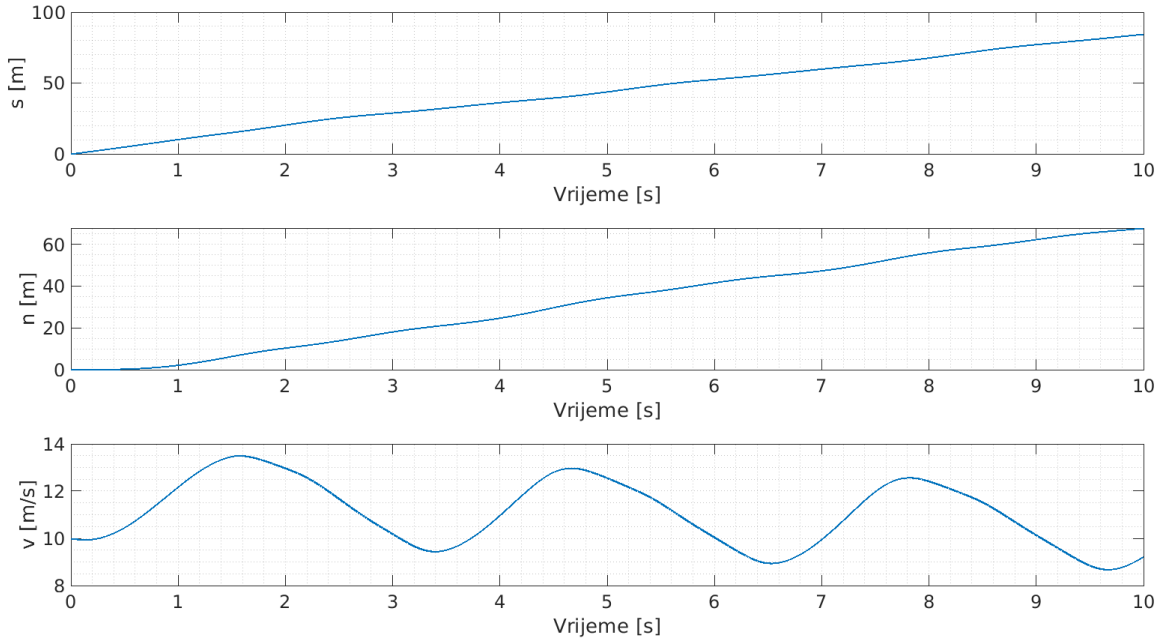
Model je pobuđen sinusnim signalima, te je za kočenje i zadavanje momenta pogonskom motoru ulaz ograničen samo na pozitivne vrijednosti (slika 5.10.).

Vozilo je simulirano sa početnom brzinom od 10 m/s na ravnoj stazi (zakrivljenost je 0) te velikim kutem ξ uz vrijeme diskretizacije 1 ms. Odabrane varijable stanja vidljive su na slici 5.11.

Vozilo se sve više odmiče od referentne putanje zbog početnog kuta ξ. Zbog prilično velike brzine te velikih i brzih skretanja bočni kut klizanja kotača doseže vrijednosti i do 10°. Za testiranje lineariziranih modela ovi odzivi predstavljati će referentne vrijednosti. Napravljena je simulacija u kojoj se procjenjuju Eulerova unaprijedna diskretizacija te



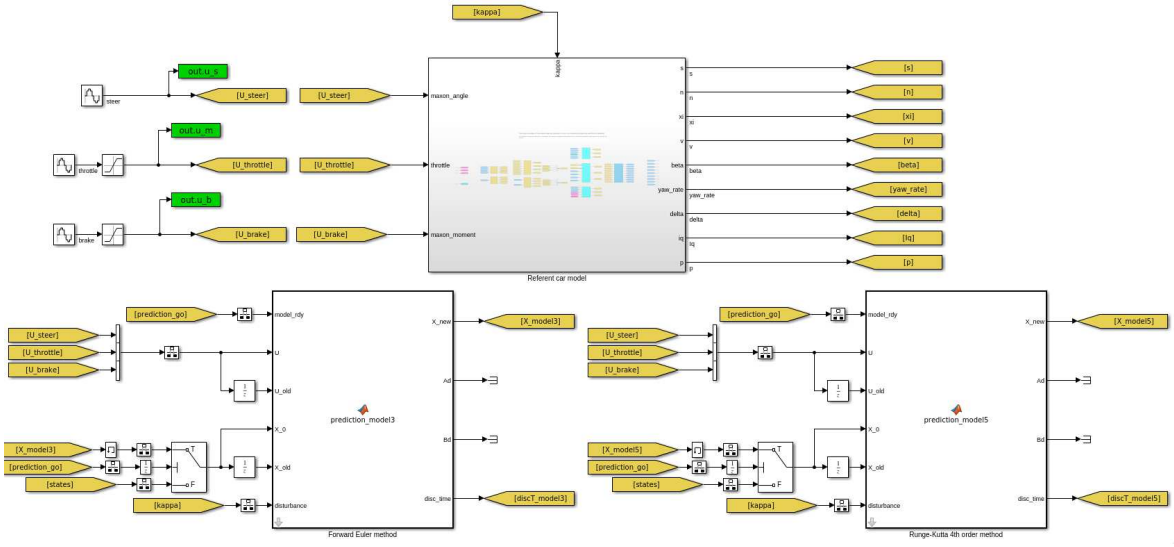
Slika 5.10. Ulazi kojima je pobuđen nelinearni model vozila.



Slika 5.11. Stanja nelinearnog modela vozila pri *mješovitom manevru*.

Runge-Kutta metoda ostvarene jednadžbama 5.49 i 5.50 Pošto razvijen dinamički model vozila ograničuje MPC tako da se koristi samo pri većim brzinama, predikcija počinje u stanju u kojem se proces trenutno nalazi. Radi toga, pri testiranju modeli će početi predikciju u nasumično odabranom vremenu simulacije. Model se testira u otvorenome krugu. U trenutku početka predikcije uzima se trenutna i prošla vrijednost varijabli iz referentnog modela te se dalje predikcija odvija nezavisno o referentnom modelu. Vari-

jable stanja korištene za definiranje daljnjih radnih točaka matrica koriste se iz modela koji se testira, a ne referentnog modela. Na taj način testira se predikcija u otvorenom krugu.



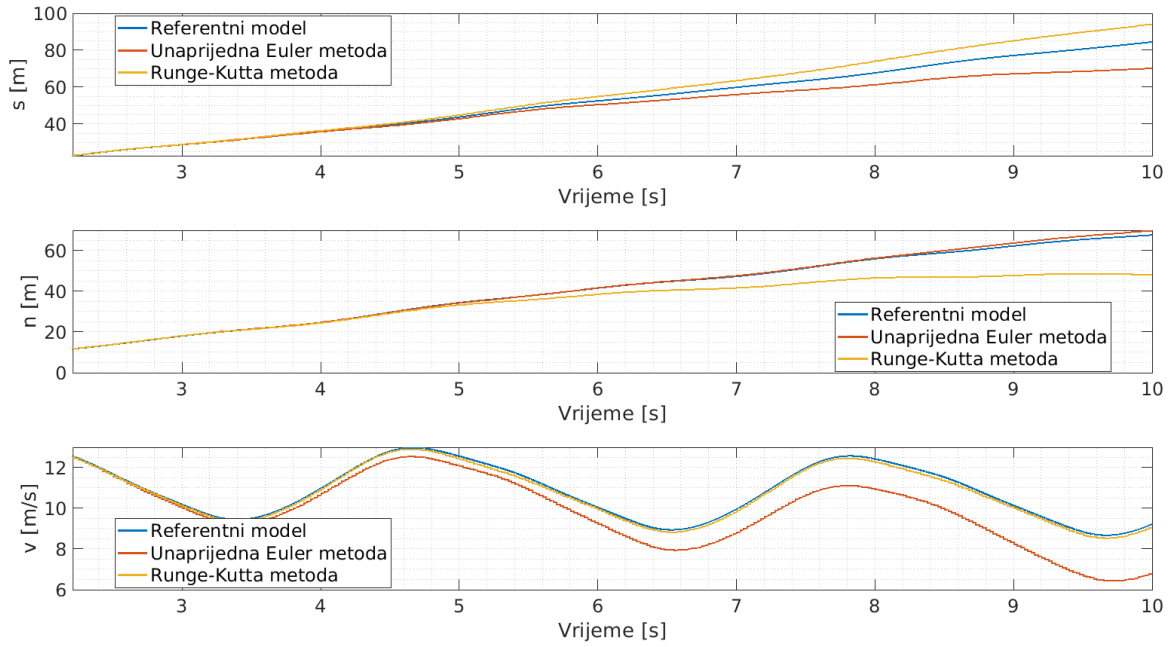
Slika 5.12. *Simulink* model za testiranje metode linearizacije i diskretizacije.

Referentni model ima vrijeme diskretizacije 1 *ms*, a za testne modele može se definirati vrijeme diskretizacije modela. Testni modeli ostvareni su uz pomoć *Matlab* funkcija te su maskirani s ciljem prosljeđivanja varijabli iz radnog prostora. Rezultati simulacije sa vremenom diskretizacije testnih modela od 25 *ms* te sa početkom predikcije u 2.2*s* prikazani su na slici 5.13.

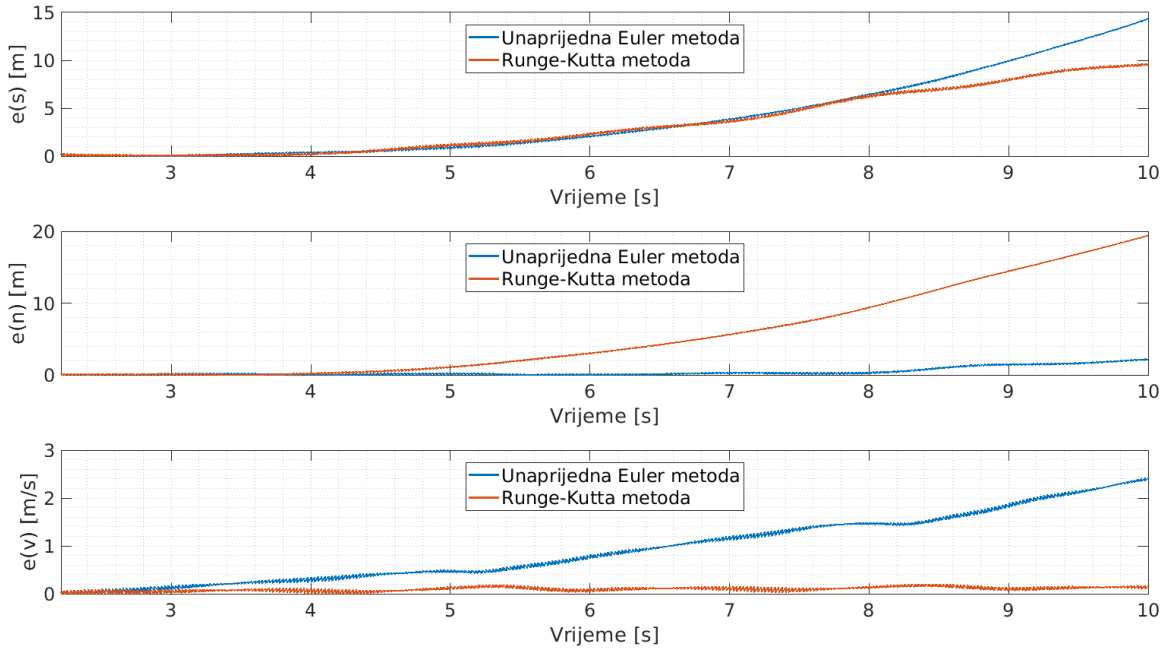
Može se vidjeti da modeli predviđaju prilično dobro. Runge-Kutta model predviđa bolje brzinu dok unaprijedna Euler metoda bolje predviđa udaljenost od referentne putanje. S obzirom na pretpostavku uvedenu na ulaze Runge-Kutta metode te sličnost susjednih radnih točaka ove dvije metode se ne razlikuju primjetno, što se može vidjeti na slici 5.14. na kojoj su prikazane greške.

Pošto se unaprijedna Eulerova metoda u simulacijma ne razlikuje previše od Runge-Kutta metode, a napisana je u eksplicitnom obliku, biti će odabrana predikcijska metoda za modelsko prediktivno upravljanje. Povećavanjem vremena diskretizacije opis modela postaje neprecizniji. Rezultati simulacije sa različitim vremenima diskretizacije vidljivi su na slici (slika 5.15.).

Simulacije potvrđuju da povećavanjem vremena diskretizacije greške predviđanje pos-

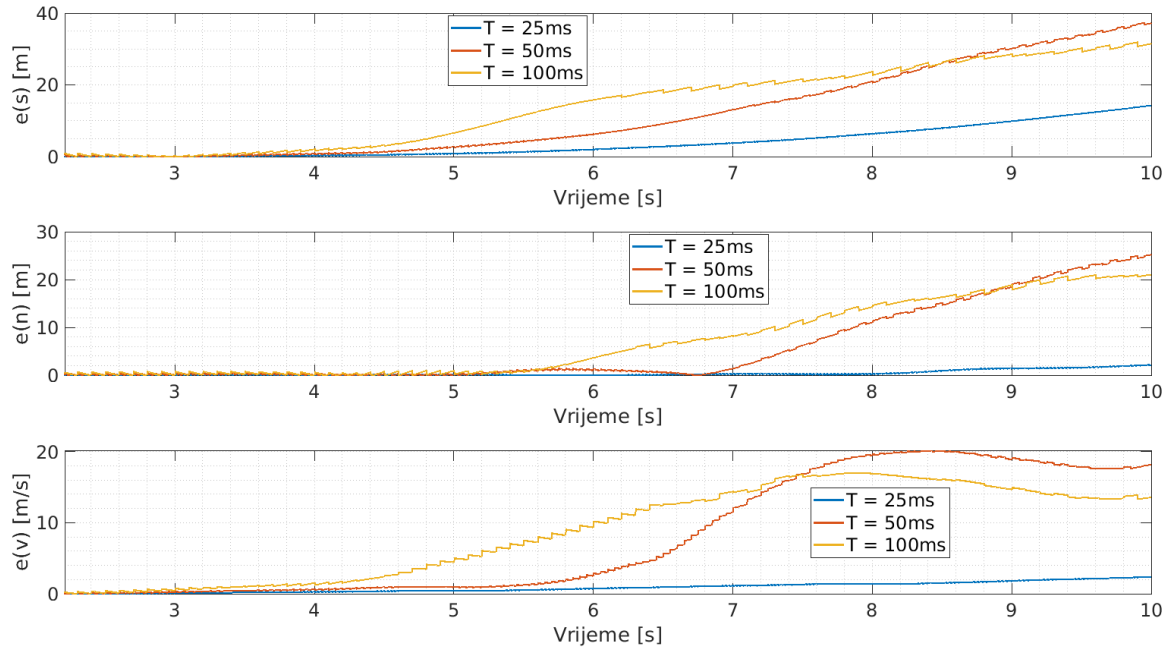


Slika 5.13. Rezultati predikcije diskretnih modela.



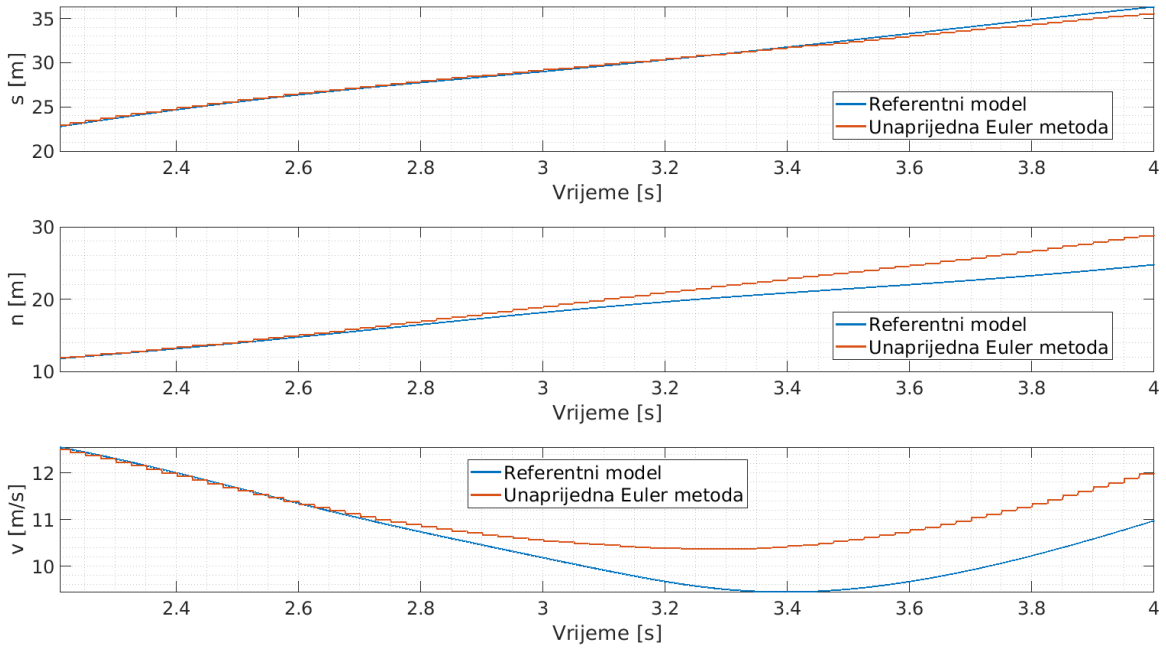
Slika 5.14. Greške predikcijskih modela.

taju sve veće. Modeli su simulirani u idealnom slučaju bez ikakvog šuma. Ovim testom mogu se vidjeti ograničenja samog algoritma. Prema prikazanim grafovima, predikcija vjerno prati model do dvije sekunde pri niskim vremenima diskretizacije. Stoga nema smisla uzimati vrijeme predikcije veće od dvije sekunde za ovakav predikcijski model sa sličnim vremenima diskretizacije.



Slika 5.15. Greške različitih vremena diskretizacije unaprijedne Eulerove metode.

Prijašnje predikcije rezultat su LTV predikcije u kojoj se matrice sukcesivno linearizi-raju u svakom diskretizacijskom koraku. Na idućoj slici prikazana je predikcija u kojoj je model lineariziran samo u trenutku početka predikcije, tj. u ovom primjeru u vremenu nakon 2.2 s simulacije. Matrica A_l konstantna je i ne mijenja se do kraja predikcije.

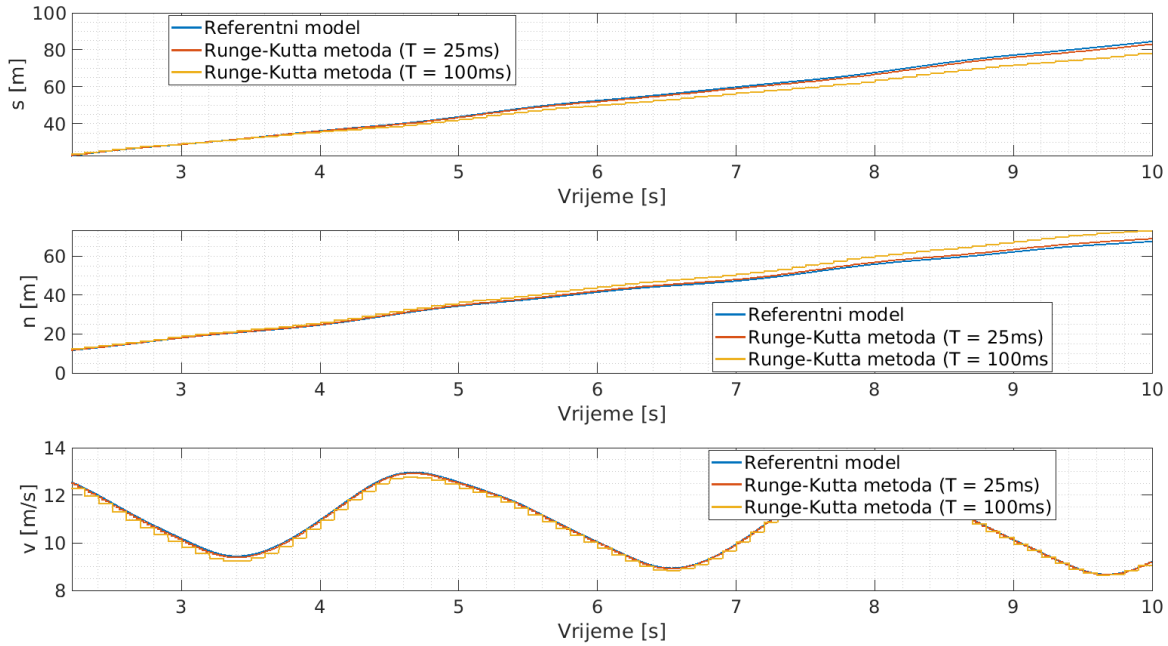


Slika 5.16. Rezultati predikcije jednom linearizirano modela uz $T = 25\ ms$.

Na slici 5.16. prikazano je vrijeme simulacije do 4 s jer greška vrlo brzo raste te u de-setoj sekundi simulacije greška udaljenosti od referentne putanje je čak veća od 100 m.

Može se vidjeti da model vjerno predviđa do 0.75 *s* pri vremenu diskretizacije od 25 *ms* stoga za adaptivni MPC nema smisla predviđati više od 0.75 *s* za slična vremena diskretizacije.

Korištenjem Runge-Kutta metode izravno nad nelinearnim jednadžbama je puno točnije nego korištenje u lineariziranom prikazu prostora stanja. Simulacija predikcije Runge-Kutta metodom opisanom jednadžbama 5.51 pri vremenu diskretizacije 25 *ms* prikazano je na slici 5.17.

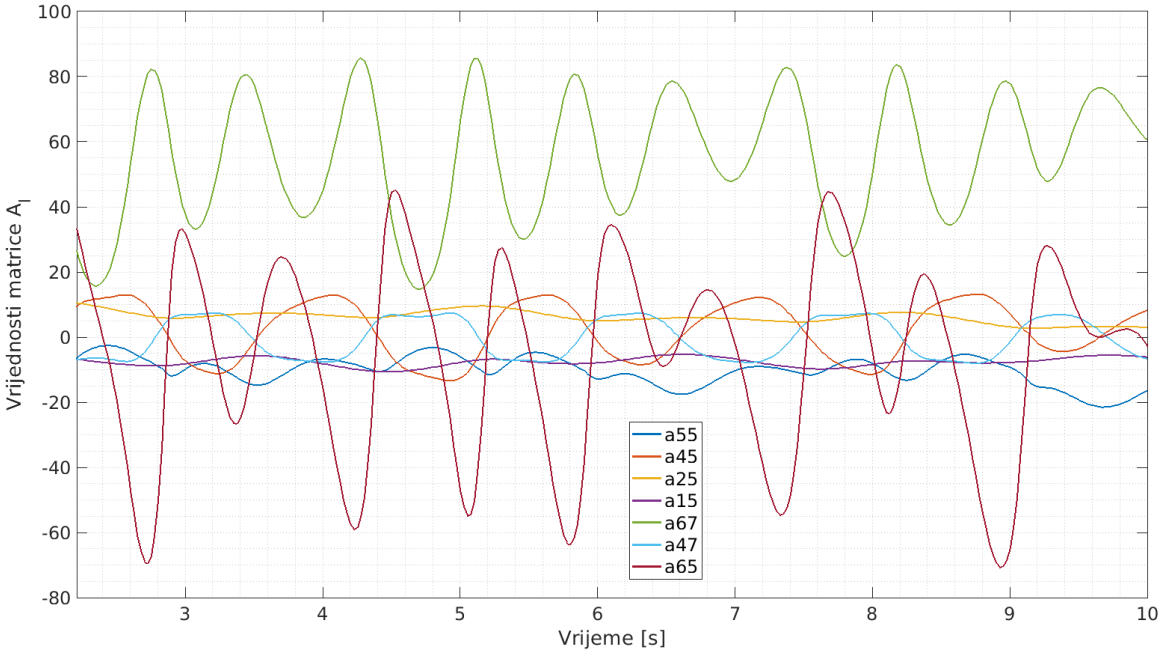


Slika 5.17. Rezultati predikcije Runge-Kutta metodom uz različita vremena diskretizacije.

Predikcijski model vrlo je točan čak i kako vrijeme odmiče. Vrijeme diskretizacije od 25 *ms* prilično je nisko vrijeme pri kojem je predikcija vrlo točna. Za vrijeme diskretizacije od 100 *ms* predikcija je također vrlo točna, ali s vremenom brže divergira. S obzirom da je iskoristivo vrijeme predikcije dvije sekunde, unutar dvije sekunde ova metoda može se smatrati savršeno točnom. Pošto se koriste direktno nelinearne jednadžbe, ovaj način korištenja Runge-Kutta metode ne može se koristiti pri sastavljanju problema za MPC optimizaciju. No zbog iznimno velike točnosti i velike brzine izvođenja može se koristiti za pouzdano dobivanje matrica $\mathbf{A}_l$ i $\mathbf{E}_l$ koje služe LTV MPC-u.

Za konačnu simulaciju modela biti će prikazane same karakteristike matrice linearizacije. Pri *mješovitom manevru* radne točke varijabli stanja se brzo mijenjaju kroz vrijeme što je vidljivo na slici 5.11. Neke od vrijednosti matrice $\mathbf{A}_l$ pri simulaciji prikazane su na

slici 5.18.



Slika 5.18. Odabrane vrijednosti matrice A_l pri simulaciji.

Može se vidjeti da iz trenutka u trenutak linerizirana matrica iznimno brzo te značajno mijenja vrijednosti. Matrica na koju se graf odnosi je linearizirana matrica modela vozila 5.31 Na slici je odabrano nekoliko vrijednosti za koje je primijećena najveća promjena. Najviše se mijenjaju ovisnost kutne brzine o bočnom kutu klizanja vozila te ovisnost kutne brzine o kutu zakreta prednjih kotača. Iznimno brze te značajne promjene vrijednosti matrice ukazuju da je problem upravljanja vozilom uz prošireni dinamički model vozila vrlo zahtijevan za adaptvni MPC.

6. Implementacija i simulacija adaptivnog MPC-a

Daljnje simulacije su rađene u okruženju *Matlab 2021a* iz razloga što je ECU *Simulink* model VulpesD vozila razvijen u istoj inačici. Korišten je operacijski sustav *Ubuntu 20.04.6 LTS* radi potencijalnog budućeg spajanja simulacija sa ostatkom autonomnih algoritama.

Matlab paketi korišteni za potrebe simulacija su:

- *Simulink*,
- *Symbolic Math Toolbox*,
- *Vehicle Dynamics Blockset*,
- *Control System Toolbox*,
- *Model Predictive Control Toolbox*,
- *Robust Control Toolbox*,
- *Simulink Control Design*,
- *Optimization Toolbox*,
- *Signal Processing Toolbox*,
- *DSP Sytem Toolbox*,
- *Statistics and Machine Learning Toolbox*,
- *Multiple XY Graph*.

Potrebno je instalirati i YALMIP sučelje [8]. Uz navedene pakete, zasebno su instalirani i razni kvadratni optimizatori radi usporedbe performansi u potpoglavlju 7.2.5. Procesor na kojem su izvršavane simulacije je *Intel® Core™ i5-8265U CPU @ 1.60GHz* × *8*

6.1. Implementacija

Modelsko prediktivno upravljanje implementirano je u *Simulink* model na način u kojem bi bilo korišteno u vozilu. Oko logike koja predstavlja MPC izgrađeno je simulacijsko sučelje za potrebe testiranja i validacije.

Svi parametri za glavni ECU model definiraju se u radnom prostoru *Matlaba* te su logično podijeljeni prema funkcionalnosti. Takvi parametri se učitavaju u model prije generiranja C koda. Na isti način, svi parametri za simulaciju MPC-a definirani su u *LoadMPCParams.m* skripti. Cijela MPC logika ostvarena je u jednom bloku koji predstavlja *Matlab* skriptu. Blok je maskiran, što omogućuje da se definirane varijable iz radnog prostora proslijede skripti. *Simulink* podržava samo osnovne tipove objekata stoga unutar skripte sve varijable moraju biti tih tipova. Ne smije se koristiti simbolička logika za dobivanje rješenja ili neki drugi napredni tipovi objekata koji se često koriste u *Matlabu*.

Među YALMIP dokumentacijom nalazi se uputa kako implementirati optimizacijski problem unutar *Simulink* okruženja da se postigne najbrže moguće vrijeme izvođenja [8]. YALMIP koristi vlastiti tip objekta nazvan *sdpvar* kojim se definiraju varijable koje predstavljaju rješenje optimizacije. Definiranjem takvih varijabli, YALMIP ima vrlo intuitivan način za sastavljanje optimizacijskog problema. Definiranje MPC optimizacijakog problema za najbrže vrijeme rješavanja dano je pseudokodom:

Algorithm 1 Pseudokod za najbrže rješavanje optimizacijskog problema u YALMIP-u

Ako je prvo pozivanje MPC-a **onda:**
  Definiraj $\mathbf{x}$, $\mathbf{u}$, $\mathbf{d}$, $\mathbf{A}$, $\mathbf{E}$, $\mathbf{r}$ kao `sdpvar` varijable za optimizaciju
  Definiraj konstante: $\mathbf{B}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{S}$, N, T
  Inicijaliziraj optimizacijski kriterij $J = 0$
  Inicijaliziraj listu ograničenja $lim = \emptyset$
  Definiraj postavke optimizatora
  Za $k = 1$ **do** N **činiti:**
    Proširi lim $\mathbf{x}_{k+1} = 2 \cdot \mathbf{x}_k + T \cdot (\mathbf{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) + \mathbf{B}(\mathbf{u}_k - \mathbf{u}_{k-1}) + \mathbf{E}(\mathbf{d}_k - \mathbf{d}_{k-1})) - \mathbf{x}_{k-1}$
    Proširi lim $(\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}, \mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max}, \mathbf{d}_{\min} \leq \mathbf{d}_k \leq \mathbf{d}_{\max})$
    Definiraj $\mathbf{e}_k = \mathbf{x}_k - \mathbf{r}_k$
    Proširi optimizacijski kriterij J izrazom $(\mathbf{e}_k^T \mathbf{Q} \mathbf{e}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k)$

  Proširi J terminalnim uvjetom $(\mathbf{e}_N^T \mathbf{S} \mathbf{e}_N)$
  Proglasi ulazne varijable $\mathbf{A}$, $\mathbf{E}$, $\mathbf{x}_0$, $\mathbf{r}$, $\mathbf{d}$
  Proglasi izlazne varijable $\mathbf{x}$, $\mathbf{u}$
  Definiraj objekt optimizacijskog problema *Kontroler*
Inače:
  Definiraj trenutne ulazne varijable ($\mathbf{A}$, $\mathbf{E}$, $\mathbf{x}_0$, $\mathbf{r}$, $\mathbf{d}$)
  Vrati optimalna rješenja $\mathbf{x}$, $\mathbf{u}$

U prvom pozivanju MPC-a definiraju se sve konstante, optimizacijske varijable, ulazi te izlazi. Stvaranjem objekta *Kontroler* YALMIP stvara cijeli optimizacijski problem. Taj dio najduže traje te nije pogodan izvršavanju u stvarnom vremenu. Kroz ostatak rada, objekt *Kontroler* sadrži cijeli problem spreman za optimizaciju te je potrebno samo proslijediti ulazne varijable. Varijable koje se mijenjaju pri svakom pozivanju MPC-a ($\mathbf{A}$, $\mathbf{E}$, $\mathbf{x}_0$, $\mathbf{r}$, $\mathbf{d}$) definiraju se kao ulazne varijable koje *Kontroler* očekuje dobiti jer je optimizacijski problem stvoren sa nepoznatim (praznim) vrijednostima tih varijabli.

Cijeli optimizacijski problem definira se unutar jedne petlje. Za svaki diskretni korak predikcijskog horizonta definiraju se ograničenja te optimizacijski kriterij. Jednadžba prostora stanja koja opisuje dinamiku procesa tumači se kao ograničenje u obliku jednakosti. Nakon tako jednostavnog definiranja problema, YALMIP izvršava pretprocesiranja, svu interakciju sa vanjskim optmizatorima te sastavljanje optimizacijskog problema opisanog jednadžbama od jednadžbe 4.11 do jednadžbe 4.16

Jedan vrlo bitan pojam pri sastavljanju kriterijske funkcije je skaliranje. Prvenstveno zbog samog optimizatora jer je vrlo osjetljiv na različite skale signala te daje vrlo loše rezultate ili ne može riješiti problem. Druga stvar je radi intuitivnijeg definiranja matrica

Q i **R**. Kriterijska funkcija je samo jedna te sve varijable što se definiraju u njoj imaju utjecaj na druge varijable. Na primjer, u izrazu $\mathbf{e}_k^T\mathbf{Q}\mathbf{e}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k$ recimo da su greške iznimno male (jednoznamenkasti brojevi), a ulazni signali su vrlo veliki zbog njihovih mjernih jedinica (stotine ili tisuće). Pri jednakim vrijednostima matrica **Q** i **R** optimizator će jako puno kažnjavati visoke ulaze jer će vrijednost funkcije tada biti iznimno velika, dok smanjivanje greške uopće neće doći do izražaja zbog malih brojeva. Rješenje bi tada bilo matricu **Q** inicijalizirati vrlo velikim brojevima tako da i greške praćenja reference dođu do izražaja. No definiranje matrice **Q** u skali tisućica, a matrice **R** kao jednoznamenkastih brojeva nije intuitivno. Na prvi pogled čini se da velika odstupanja sustava od reference bivaju kažnjavana puno više nego velike vrijednosti ulaznih signala, dok su u stvarnosti oba kriterija kažnjavana jednako radi velike skale ulaznih signala.

Skalirati treba upravljačke signale te odstupanja od reference. Upravljački signali se generalno skaliraju tako da budu signali bez mjernih jedinica u rasponu od 0 do 1. Radi toga trebaju se skalirati sa maksimalnim vrijednostima koje mogu postići, tj. gornjim ograničenjima na upravljački signal $\mathbf{u}_{\max}$. Svaki od upravljačkih signala ima svoj raspon te se za skaliranje cijelog vektora mora koristiti Hadamar-Schur produkt (*engl. element-wise multiplication*) sa inverzom vektora maksimalnih vrijednosti upravljačkih signala: $\mathbf{u}_k \cdot \mathbf{u}_{\max}^{-1}$. Drugim riječima treba podijeliti kontrolne signale sa pripadajućim maksimalnim vrijednostima. Da bi izraz $\mathbf{B}\mathbf{u}_k$ ostao jednak nakon takvog množenja potreban je Hadamar-Schur produkt sa vektorom $\mathbf{u}_{\max}$.

$$\underbrace{\mathbf{B} \cdot \mathbf{u}_{\max}}_{\mathbf{B}_s}\left(\mathbf{u}_k \cdot \mathbf{u}_{\max}^{-1}\right). \tag{6.1}$$

Hadamar-Schur produkt posjeduje svojstvo komutativnosti. Matrica **B** i vektor $\mathbf{u}_{\max}$ nisu jednakih dimenzija, stoga je potrebno vektor proširiti nulama na početku. Matrica $\mathbf{B}_s$ tada postaje matrica skaliranih upravljačkih signala te će se u nastavku smatrati da je pri svakoj simulaciji izvršeno skaliranje matrica, a koristiti će se jednak izraz **B** radi jednostavnosti. U izrazima za ograničenja upravljačkih signala ($\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}$), gornje granice su sada 1, a donje 0 ili -1 ovisno o signalu. Optimizator će vratiti vektor optimalnih upravljačkih varijabli u rasponu od -1 do 1 te je potreban Hadamar-Schur produkt sa vektorom $\mathbf{u}_{\max}$ da se vrati skala za daljnje upravljanje.

Jednak postupak vrijedi za skaliranje odstupanja od reference $\mathbf{e}_k$. Maksimalne vrijednosti odstupanja od reference nisu poznate za sve signale, jer su i same granice nekih varijabli stanja defnirane proizvoljno. Vektor koji će skalirati greške ($\mathbf{e}_{\max}$) treba definirati proizvoljno tako da maksimalne vrijednosti odgovaraju karakteristici discipline koju vozilo vozi. Odstupanja od reference treba podijeliti sa maksimalnim vrijednostima, tj. napraviti Hadamar-Schur produkt sa inverzom vektora skaliranja.

$$\mathbf{e}_k \cdot \mathbf{e}_{\max}^{-1}. \tag{6.2}$$

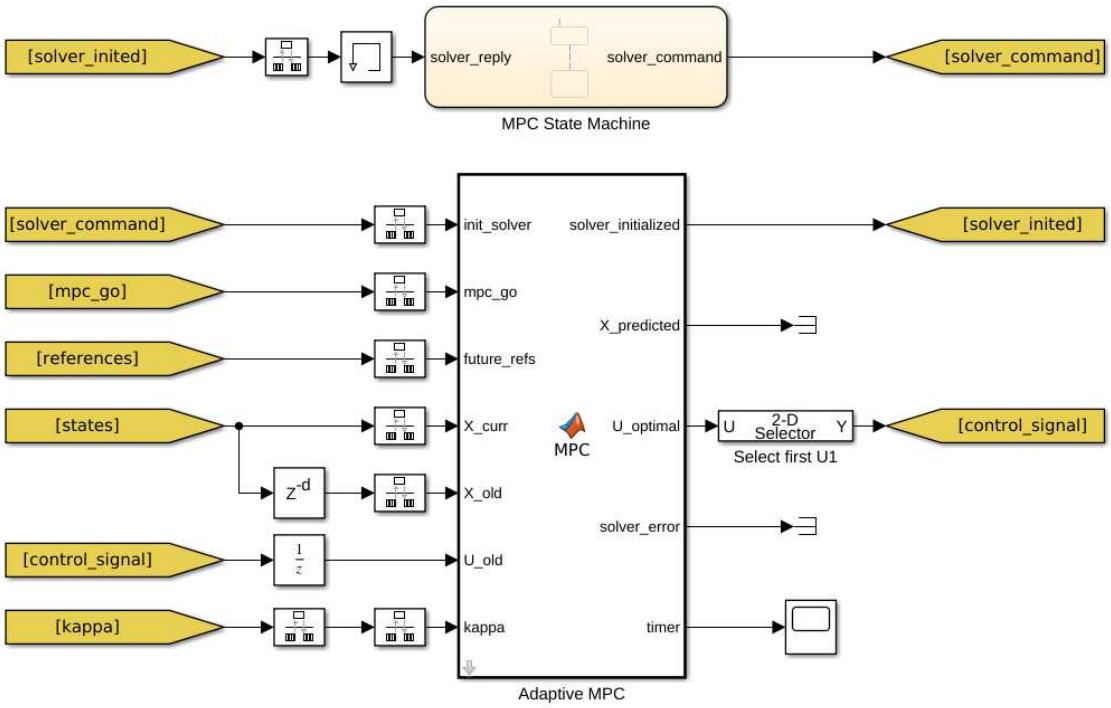
Skaliranje odstupanja od reference nema utjecaja na jednadžbe stanja sustava jer se ne koristi jednadžba $\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{D}\mathbf{u}_k$. Pri pisanju koda, vektor $\mathbf{e}_{\max}^{-1}$ nazvan je $\mathbf{C}_s$ jer se generalno skaliraju izlazi preko matrice $\mathbf{C}$ te se mijenjaju ograničenja na izlaze na jednak način kao i na ulaze. Varijable stanja se ne skaliraju te njihove granice ostaju nepromijenjene.

Prijeđeni put s nema gornju granicu. Umjesto definiranja gornje granice kao beskonačno bolje je ne definirati gornju granicu. Za element vektora skaliranja greške $\mathbf{C}_s$ koji odgovara greški prijeđenog puta, koristit će se trenutno prijeđeni put iz vektora $\mathbf{x}_0$. Na taj način skala greške prijeđenog puta će stalno biti oko 1.

Matrica $\mathbf{B}$ je konstantna te se definira u samom MPC-u. Matrica $\mathbf{A}$ mijenja se pri svakom pozivanju MPC-a stoga se treba uvijek proslijediti *Kontroleru*. Za sastavljanje MPC problema matrica $\mathbf{A}$ definira se kao *spdvar* nepoznata matrica. No međutim definiranjem cijele matrice nepoznatom YALMIP sastavlja problem sa pretpostavljenih 81 nepoznatih vrijednosti matrice. Takav problem je loše definiran i ne daje zadovoljavajuća rješenja. Potrebno je koristiti što više *apriori* informacija za sastavljanje problema kako bi problem bio bolje definiran [8]. Stoga se matrica $\mathbf{A}$ definira kao konstanta u obliku zadanom 5.31 Samo nepoznati elementi matrice kojih ima 28 proglašavaju se nepoznatim te se prosljeđuju *Kontroleru* pri svakom pozivanju MPC-a. Jednako vrijedi i za matricu $\mathbf{E}$ koja ima samo 2 promjenjiva elementa. Pri svakom izvršavanju MPC-a prosljeđuje se također i prošla vrijednost upravljačkog signala kao i prošla vrijednost varijabli stanja zbog jednadžbe kojom je predikcija ostvarena. Prošli upravljački signal također se mora proslijediti u skaliranom obliku.

U slučaju da optimizacijski problem ne može biti riješen *Kontroler* vraća grešku te se prekida cijela simulacija. Za uklanjanje pogrešaka u kodu može se mijenjati vrijednost varijable *verbose*. Varijabla prima vrijednosti od 0 do 2, a veći brojevi pružaju više dijagnostike. Sva dijagnostika usporava izvršavanje optimizacije, stoga je za najbrže izvođenje MPC-a potrebno onesposobiti svu dijagnostiku postavljanjem varijable *verbose* na vrijednost 0.

U *Simulink* blok koji predstavlja MPC implementirano je mjerenje vremena izvršavanja bloka. Na taj način će se uspoređivati vremena izvođenja raznih algoritama za optimizaciju.



Slika 6.1. Blokovski prikaz MPC logike u razvojnom okruženju *Simulink*.

Na slici 6.1. prikazan je izgled kompletne MPC logike. Blok MPC izvršava se frekvencijom od 50 Hz kao što je spomenutu u poglavlju 4.5. Cijela simulacija izvodi se korakom od 1 *ms* stoga su potrebni *Rate Transition* blokovi koji služe kao međuspremnici za signale. Razvijen je i automat stanja koji upravlja radom MPC-a. Svrha automata stanja je da vrši komunikaciju sa ostatkom ECU modela vozila. U ovoj simulaciji daje informacije o tome da je stvoren objekt *Kontroler* te da je MPC spreman za izvođenje u stvarnom vremenu. Također može služiti da prekine izvršavanje MPC-a pri niskim brzinama vozila te da prosljeđuje upravljačke signale iz PID regulatora umjesto MPC-a. Kasnije se istom

logikom može ponovo pokrenuti MPC kada se prijeđe željena brzina vozila. MPC blok vraća cijeli horizont optimalnih varijabli stanja te upravljačkih signala radi dijagnostike. Po načelu pomičnog horizonta bira se prvi upravljački signal, a ta je logika ostvarena *Simulink* blokom.

S obzirom da MPC radi u vremenski diskretnim koracima bitno je da reference i smetnje budu definirane u jednako vremenski udaljenim koracima. Ako je vrijeme diskretizacije modela 25 ms, sve reference moraju biti vremenski udaljene 25 ms. Reference za ovu simulaciju biti će dobivene iz otvorenog kruga modela. Vozilo je pobuđeno proizvoljnim ulazima te su pohranjene varijable stanja kroz simulaciju. Te iste varijable stanja služiti će kao reference MPC-u da proba rekreirati vožnju iz prethodne simulacije.

MPC može pratiti reference svih 9 varijabli stanja. No u stvarnim vožnjama vrlo je teško točno dobiti reference poput struje i_q te tlaka u sustavu kočenja p. Algoritam optimizacije trajektorije koristi jednak model vozila te vraća reference svih varijabli stanja. Ako bi se pratile reference svih varijabli stanja bilo bi teško odrediti koje varijable stanja zahtijevaju veću težinu, tj. bolje praćenje reference. Na primjer, isplati li se više pratiti reference kutne brzine vozila (w_z) ili pozicije zakreta kotača (δ)? Bilo bi teško i ne intuitivno odrediti potrebne težine matrice $\mathbf{Q}$. Zato se u ostatku simulacija neće pratiti sve varijable stanja, već samo one koje su bitne za autonomnu vožnju. To su reference na brzinu vozila (v), udaljenost vozila od referentne putanje (n) te kut gledanja vozila naspram referentne putanje (ξ). MPC će sam određivati optimalne vrijednosti varijabli stanja koje ne prate referencu.

Odabir varijabli koje će pratiti referencu vrši se kroz matricu $\mathbf{Q}$. Matrica $\mathbf{Q}$ je dijagonalna matrica u kojoj svaki element na dijagonali otežava odgovarajuću varijablu stajna u vektoru $\mathbf{e}_k$. U dijagonali matrice $\mathbf{Q}$, svi elementi koji otežavaju varijable nebitne za praćenje reference postavljeni su na vrijednost 0. Na taj način optimizator neće uzimati u obzir greške varijabli koje ne prate referencu jer su nakon množenja jednake nuli, tj. ne dodaju nikakvu težinu kriterijskoj funkciji.

Reference se alociraju prema prijeđenom putu. Na taj način su ovisne o putu s na kojem se vozilo nalazi a ne o vremenu vožnje. Trenuto prijeđeni put s određuje početnu referencu koja se zadaje, a idućih N referenci naspram početne su jednako vremenski

udaljeni prema vremenu diskretizacije T.

Poznato je kako će mjerene smetnje (κ) utjecati na vladanje modela, stoga se i one moraju proslijediti MPC-u kao i buduće reference. Zakrivljenosti putanje u budućim vremenskim trenucima možemo proslijediti samo ako pretpostavimo da će vozilo stvarno proći tim dijelom staze. Drugim riječima pretpostavka je da MPC dobro prati reference, te se jedino u tom trenutku isplati prosljeđivati buduće zakrivljenosti putanje. Na primjer, u slučaju kada vozilo kasni za referentnom brzinom te vozi puno sporije nego što bi trebalo. Referentne zakrivljenosti putanje definirane su za idućih nekoliko desetaka metara putanje jer će se pri velikoj brzini vozila prijeći veći put za isti predikcijski horizont. No vozilo vozi puno sporije te će mu u stvarnosti predikcija vrijediti za nekoliko metara. U tom slučaju zavoj koji bi se pojavio nakon nekoliko desetaka metara, u predikciji će se pojaviti za nekoliko metara. MPC će trenutno poduzeti akcije za nadolazeći *virtualni* zavoj te završiti van staze. To se može riješiti aktivnim prilagođavanjem referenci zakrivljenosti putanje prema trenutnoj brzini.

Još jedan zahtjev na upravljanje je da se ne smije u isto vrijeme kočiti i zadavati moment na pogonske motore. Kao što je opisano u poglavlju 2., vozilo se gasi pri istodobnom kočenju i zadavanju momenta. Prema [3] taj problem može se riješiti dodatnim ograničenjem:

$$\mathbf{u}_m \mathbf{u}_b = 0. \tag{6.3}$$

Ova jednadžba je istinita samo ako je jedna vrijednost jednaka nuli ili obje. Ovo ograničenje je nelinearno jer množi dvije varijable. Neki optimizacijski algoritmi koji rješavaju kvadratne probleme mogu riješiti problem uz malo nelinearnih izraza. Algoritam za rješavanje tada ima znak + na kraju imena da se naglasi korištenje nelinearnih izraza, poput *Gurobi+*. Simulacijama se pokazalo da jedan nelinearni izraz u takvim kvadratnim optimizatorima ne usporava izvršavanje MPC-a naspram slučaja u kojem nema nelinearnih izraza.

6.2. Rezultati

Vrijeme trajanja simulacije je 22 s. Kao što je spomenuto, model se prvo upravlja u otvorenom krugu, a zatim su se snimale varijable stanja za reference. MPC zatim pokušava rekreirati ponašanje otvorenog kruga uz dane reference brzine (v), udaljenosti od referentne putanje (n) te kuta gledanja vozila naspram referentne putanje (ξ). U otvorenome krugu vozilo je simulirano da postigne *mješoviti manevar* s ciljem što boljeg pobuđivanja nelinearnih elemenata.

Ograničenja na varijable stanja postavljena su jednako kao u tablici 5.1. osim vrijednosti stanja (n) za koju se koriste konstantne vrijednosti visokih vrijednosti jer je teško generirati dobru putanju u otvorenom krugu. Otvoreni krug simuliran je ulaznim signalima koji definiraju *mješoviti manevar*. Razlika naspram testiranja modela vozila u odjeljku 5.4. je što su aktuatori kočenja i pogona pobuđeni step signalima s ciljem još kompliciranijeg zadatka za MPC. Vozilo u simulaciji kreće iz nekog početnog stanja, a vrijednosti početnih stanja definirane su tako da varijabla n kroz simulaciju bude relativno blizu nule. Vrijednost zakrivljenosti putanje je nula radi jednostavnosti što znači da se vozilo kreće po ravnoj stazi.

Matrica **Q** na dijagonali jedino ima težine za n, ξ i v dok su svi ostali elementi nule. Zbog toga će se u nastavku definirati samo koeficijenti težine za pojedino stanje: Q_n, Q_ξ i Q_v. Jednako vrijedi i za matricu **R** koja na dijagonalama ima vrijednosti R_s, R_m i R_b. Matrica terminalnih vrijednosti **S** je identična matrici **Q**, ali radi jednostavnosti svi koeficijenti težina imaju jednaku vrijednost.

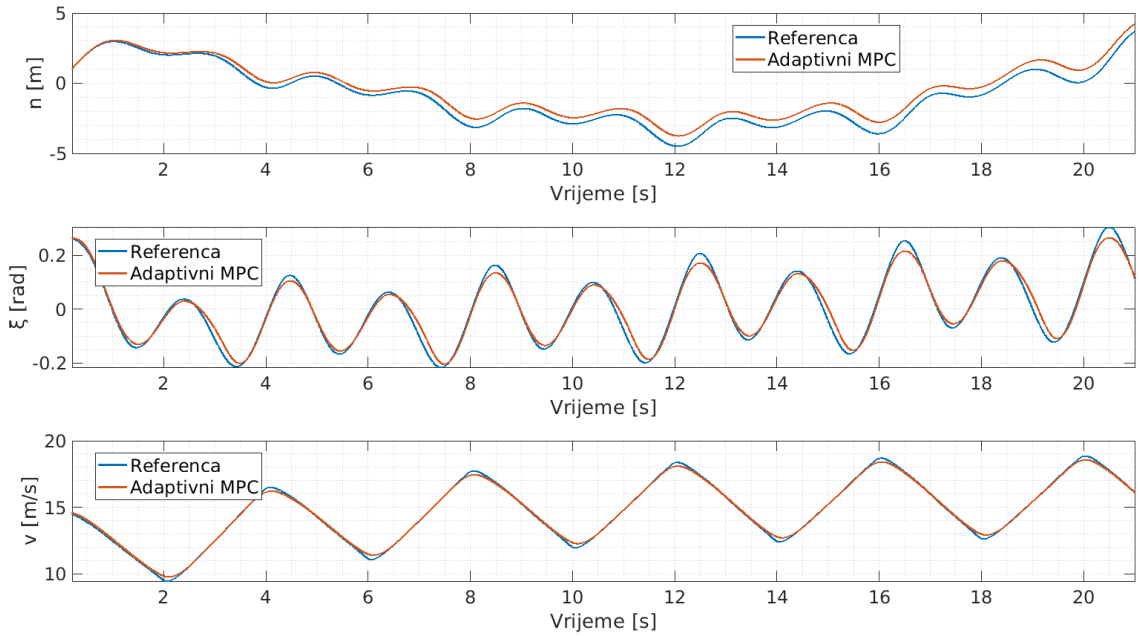
6.2.1. Simulacija 1 - Agresivno upravljanje

Tablica 6.1. Vrijednosti konstanti simulacije 1.

Konstante	Q_i	R_i	S	T	N
Vrijednosti	2, 1, 1	0.01, 0.01, 0.01	1.5	25 ms	30

Vrijednosti konstanti simulacije navedene su u tablici 6.1. Ovakve postavke matrice **R** rezultiraju vrlo agresivnim upravljačkim signalima. No na taj način MPC najbolje prati referencu jer nije ograničen u smislu upravljačkih signala. Rezultati praćenja referenci

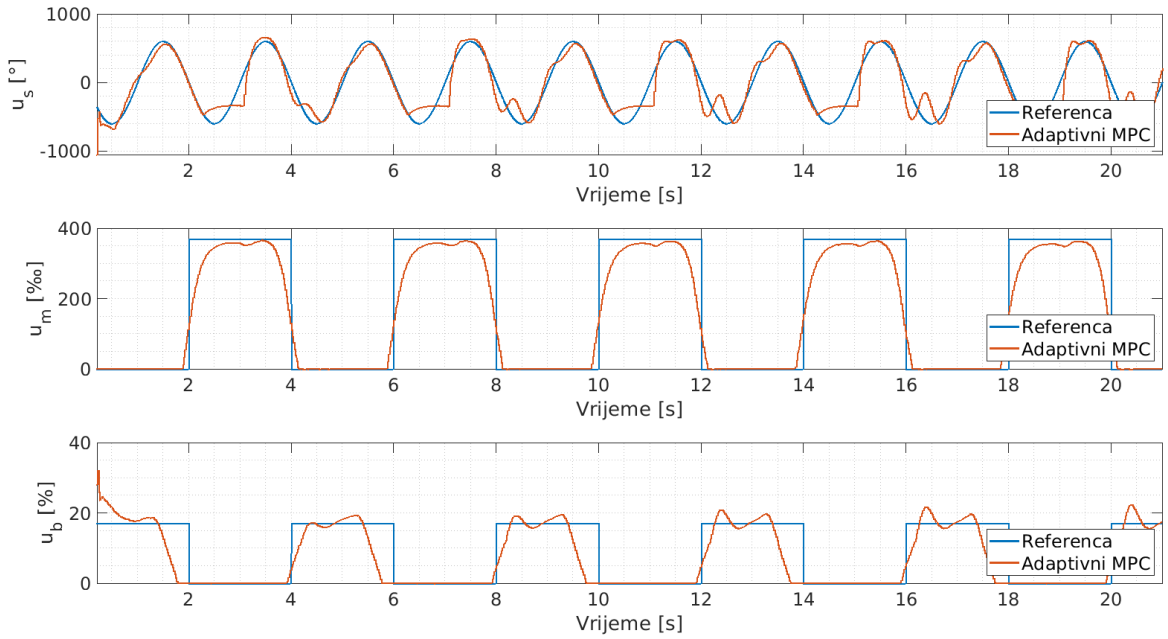
uz ove postavke prikazani su slikom 6.2.



Slika 6.2. Rezultati praćenja referenci adaptivnog MPC-a simulacije 1.

MPC uspijeva vrlo dobro pratiti referencu. Kroz simulaciju se postupno povećava brzina vozila te se može vidjeti da je s vremenom praćenje reference n sve lošije. Najveće odstupanje od referentne udaljenosti iznosi 0.9 m. Na slici 6.3. prikazan je graf koji uspoređuje signal kojim su generirane reference za *mješoviti manevar* te upravljački signal kojim je MPC pokušao pratiti zadane reference.

Vidljivo je da signali izgledaju slično iako MPC prati samo reference n, ξ i v koje su vrlo općenite. Na početku simulacije vozilom se ne upravlja sve do 0.2 s simulacije kada MPC počinje raditi. Zbog toga se na samom početku upravljačkih signala mogu vidjeti ispravljačke naredbe jer u 0.2 s vremena pri velikim brzinama vozilo može značajno promijeniti radnu točku.



Slika 6.3. Usporedba upravljačkog signala adaptivnog MPC-a i referentnog upravljačkog signala simulacije 1.

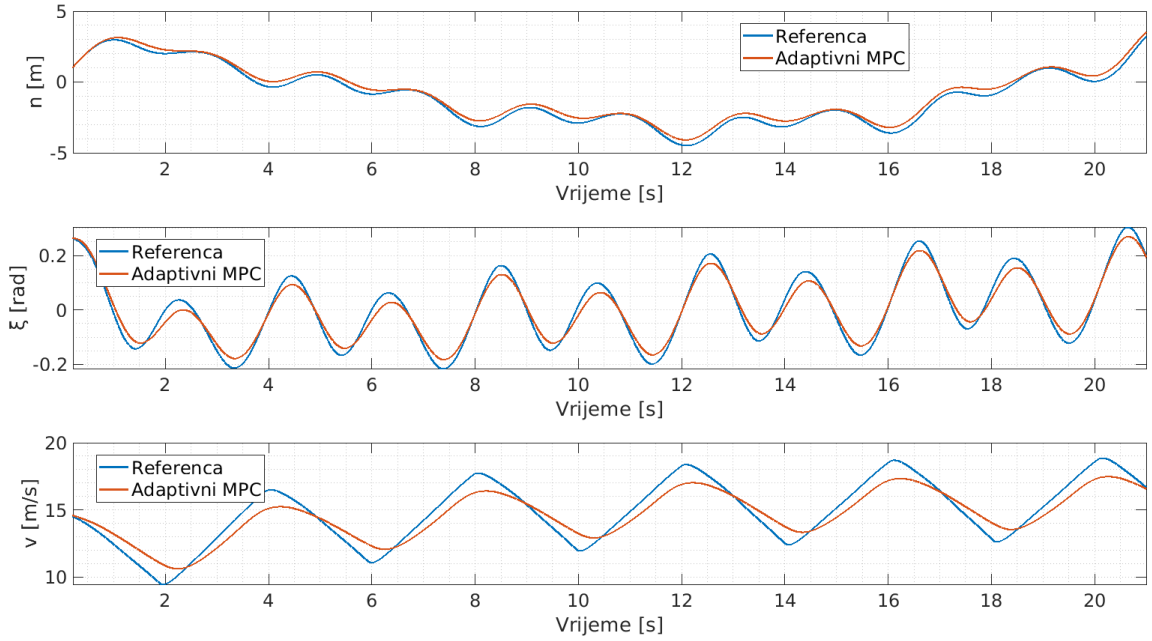
6.2.2. Simulacija 2 - Blago upravljanje

U stvarnosti je cilj provesti vozilo zadanom putanjom uz što blaže upravljačke signale. Na taj način aktuatori se manje zagrijavaju i prediktivna svojstva više dolaze do izražaja. U idućoj simulaciji zadržane su iste postavke osim matrice R koja je promijenjena tako da MPC koristi blaže upravljačke signale.

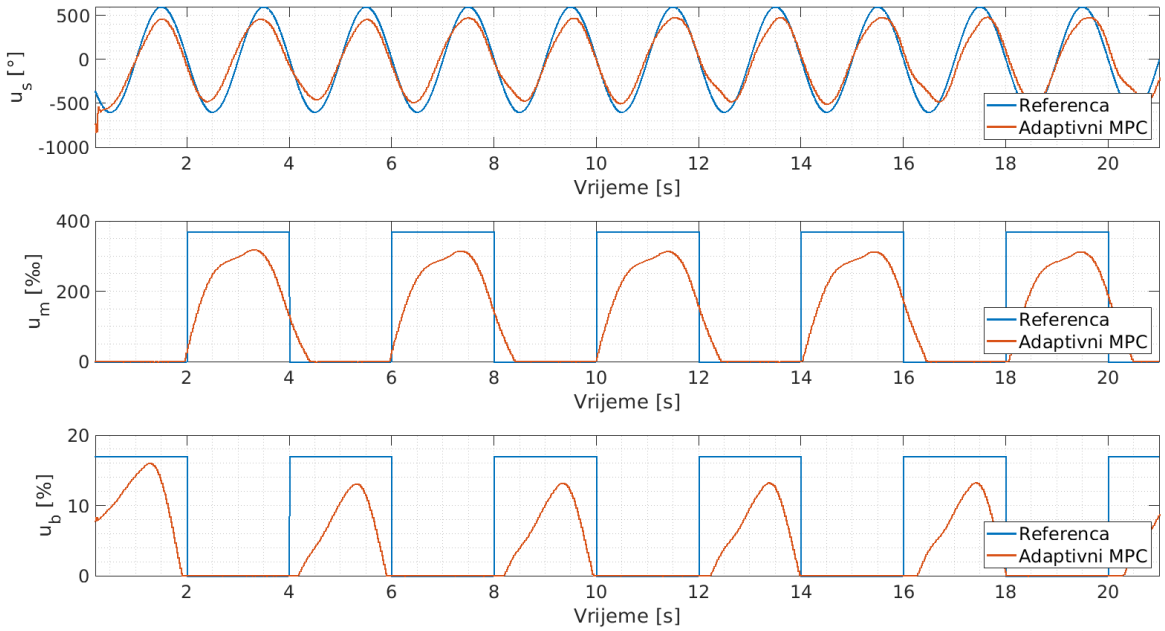
Tablica 6.2. Vrijednosti konstanti simulacije 2.

Konstante	Q_i	R_i	S	T	N
Vrijednosti	2, 1, 1	0.1, 0.1, 0.1	1.5	25 ms	30

MPC sada ne koristi visoke vrijednosti upravljačkih signala. Referenca brzine vozila ne uspijeva se pratiti idealno, no manjak agresivnosti na upravljačkim signalima rezultira boljim praćenjem reference n. Jednako tako MPC, upravljački signal je glađi iako nisu navedena ograničenja ili težine na promjenu ulaznog signala.



Slika 6.4. Rezultati praćenja referenci adaptivnog MPC-a simulacije 2.



Slika 6.5. Usporedba upravljačkog signala adaptivnog MPC-a i referentnog upravljačkog signala simulacije 2.

6.2.3. Simulacija 3 - Rub stabilnosti

U ovoj simulaciji koristiti će se veće vrijeme diskretizacije. Postavke su iste kao u prošloj simulaciji osim vremena diskretizacije i predikcijskog horizonta. Kombinacija vremena diskretizacije i predikcijskog horizonta odabrana je tako da MPC predviđa 0.75 s u budućnost.

Tablica 6.3. Vrijednosti konstanti simulacije 3.

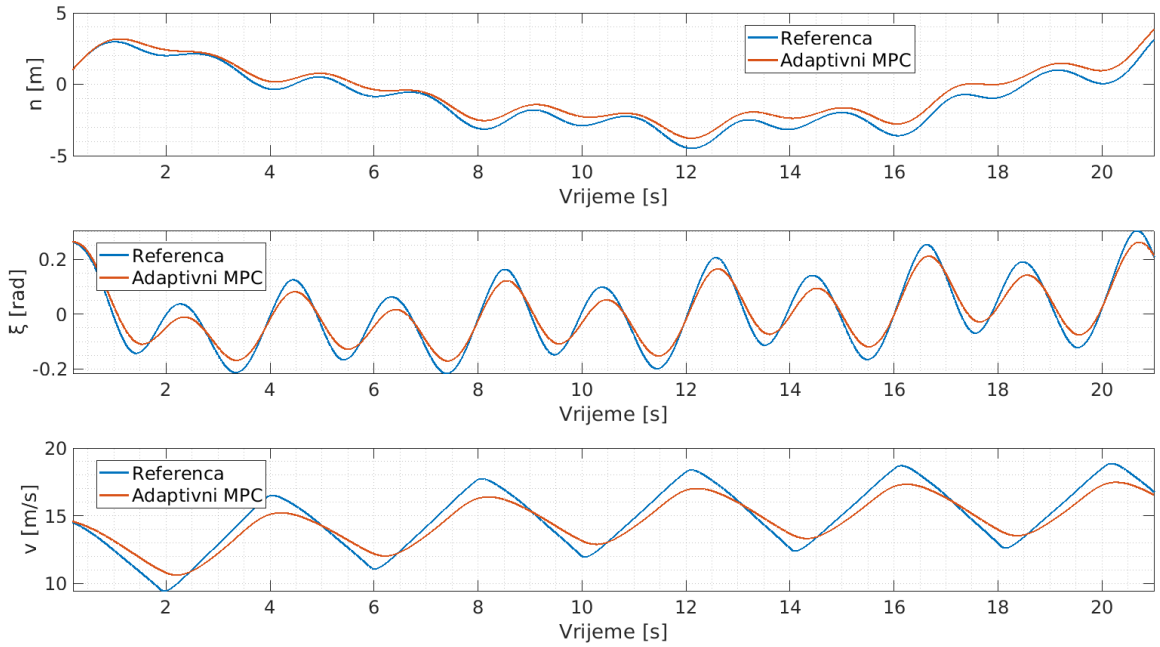
Konstante	Q_i	R_i	S	T	N
Vrijednosti	2, 1, 1	0.1, 0.1, 0.1	1.5	75 ms	10

Pri ovako visokom vremenu diskretizacije sustav je vrlo blizu nestabilnosti. Bez definiranja ograničenja na promjenu ulaznog signala sustav postane nestabilan i optimizator ne može pronaći rješenje. Zbog toga je u ograničenja MPC-a dodan i uvjet na gradijent promjene ulaznog signala:

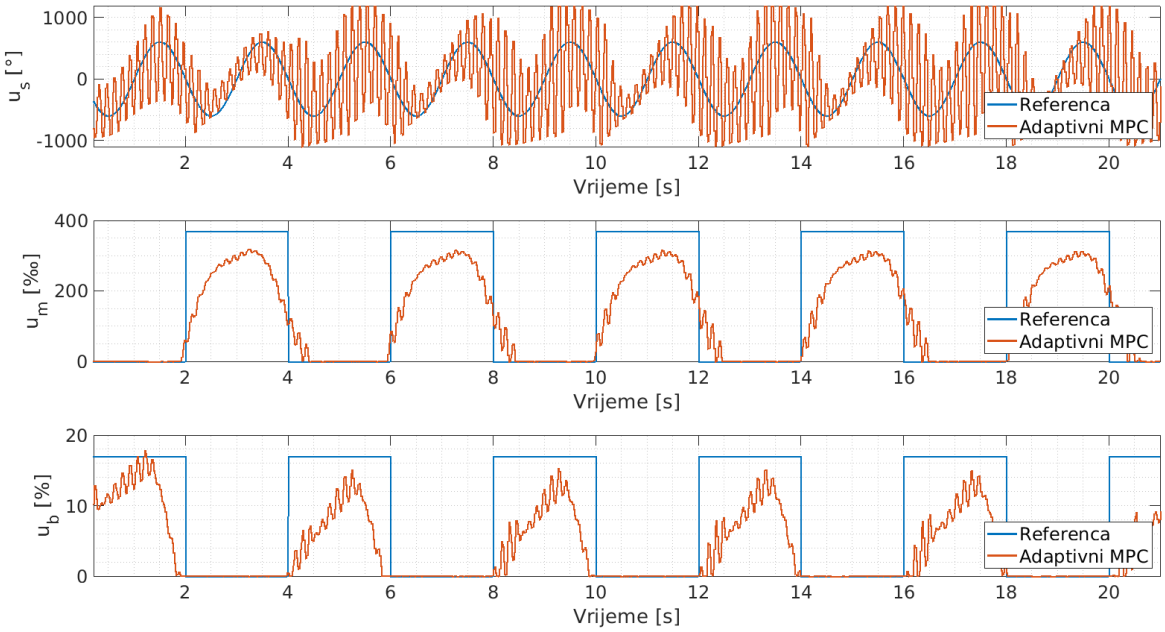
$$-\alpha <= \mathbf{u}_k - \mathbf{u}_{k-1} <= \alpha. \tag{6.4}$$

Konstanta α dobivena je eksperimentalno na način da ograničenje na gradijent ulaznog signala nema utjecaj kada je MPC stabilan, a da sprječava odlazak u potpunu nestabilnost prilikom divergencije signala. Čak i pri niskim vremenima diskretizacije sustav može biti blizu nestabilnosti pri istovremenim naglim promjenama nekih signala. Definirano ograničenje sprječava odlazak u nestabilno područje i smanjuje vrijeme u kojem je MPC blizu nestabilnosti. Radi navedenih korisnih svojstava, ograničenja na gradijent ulaznog signala ostati će implementiran u ostatku simulacija.

U ovoj simulaciji MPC je konstantno blizu nestabilnosti, a ograničenja na gradijent ulaznog signala su aktivirana u svakom koraku. Sustav i dalje dobro prati referentne signale što se može vidjeti na slici 6.6., ali je upravljački signal neiskoristiv.



Slika 6.6. Rezultati praćenja referenci adaptivnog MPC-a simulacije 3.



Slika 6.7. Usporedba upravljačkog signala adaptivnog MPC-a i referentnog upravljačkog signala simulacije 3.

6.2.4. Nedostatak Adaptivnog MPC-a

Problem adaptivnog MPC-a javlja se uključivanjem zakrivljenosti putanje u simulaciju. Pri raznim postavkama ne može se postići praćenje reference jer optimizator ne može riješiti problem niti u jednom trenutku.

Pri linearizaciji matrica **A** i **E**, zakrivljenost staze κ utječe na obje matrice. Zbog toga

što se linearizacija događa samo jednom pri pozivu MPC-a, kroz predikciju se pretpostavlja konstantan zavoj. Ako je pri pozivanju MPC-a trenutni zavoj određene vrijednosti κ, matrice će biti linearizirane oko radne točke toga zavoja. Buduće vrijednosti κ ulaze u predikciju, ali dinamika modela je potpuno kriva jer se promjenjive vrijednosti zavoja računaju sa konstantnim matricama. Što je predikcija duža ta nepravilnost više dolazi do izražaja. Može se smanjiti vrijeme predikcije čime se osigurava da predikcija djeluje blizu radne točke zavoja. No smanjenje vremena predikcije nije poželjno iz dva razloga. Prvi je smanjenje stabilnosti MPC-a, a drugi je loše praćenje reference zbog kratke predikcije.

Još jedan način na koji se može zanemariti ovaj problem je definiranje novog modela koji ne uključuje zakrivljenost staze. Zakrivljenost staze preslikavala bi se na vrijednosti n i ξ pri čemu bi one postizale vrlo visoke vrijednosti ovisno o tome koliko je staza dugačka. No tada bi vrijednost ξ poprimala iznimno velike vrijednosti te bi dolazilo do skokova u fazi koji su veliki problem za analitički model.

Potrebno je kroz samu predikciju mijenjati matrice $\mathbf{A}$ i $\mathbf{E}$ da bi model uspio pratiti promjene zakrivljenosti staze. Stoga je za daljnje simulacije razvijen LTV MPC.

7. Implementacija i simulacija LTV MPC-a

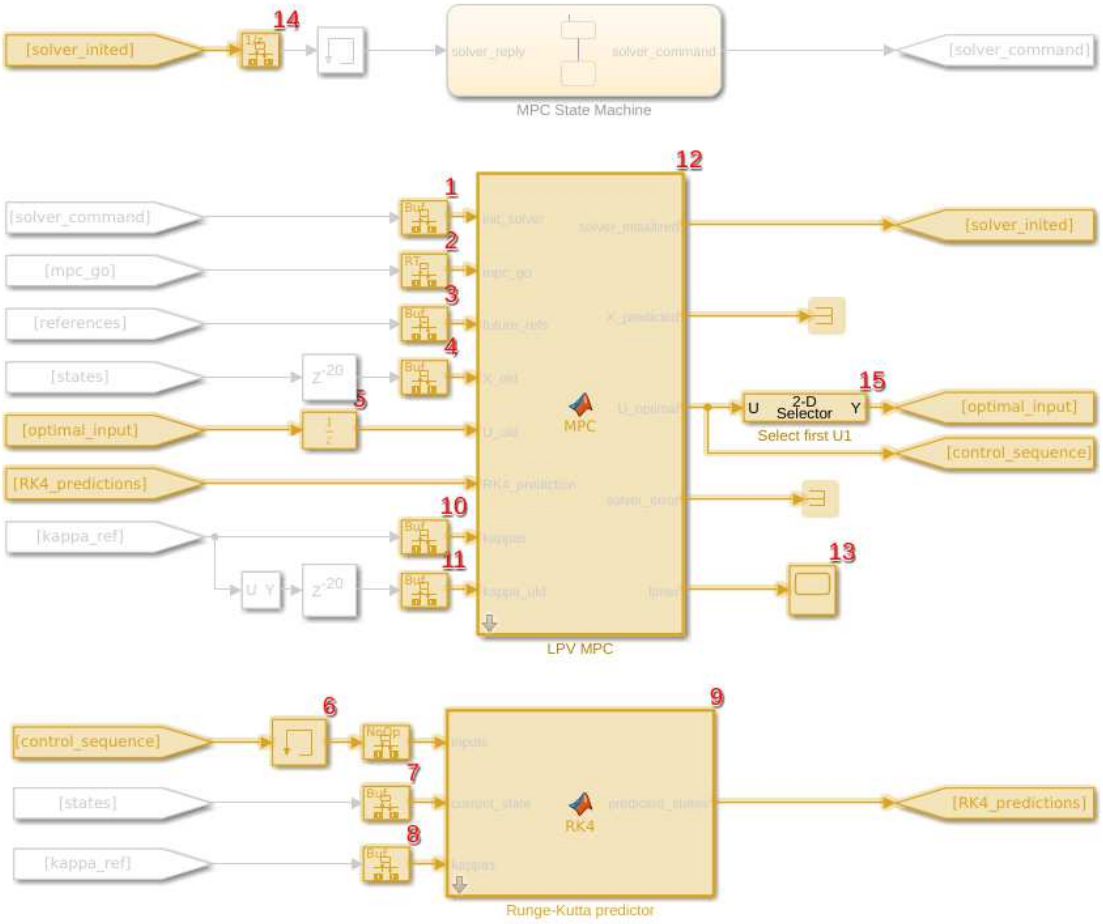
LTV MPC je u suštini jednak adaptivnom MPC-u uz iznimku što zahtijeva već gotove matrice **A** i **E** za predikciju. Problem je što LTV MPC zahtijeva matrice linearizirane oko budućih radnih točaka koje su nepoznate jer ih MPC tek treba izračunati. Pojednostavljenje problema je koristiti optimalne radne točke izračunate u prošlom koraku izvršavanja. No takve radne točke su u ovome slučaju izračunate koristeći unaprijedna Eulerovu metodu koja nije vrlo točna. Stoga će se buduće radne točke računati Runge-Kutta metodom četvrtog reda koristeći upravljačke signale dobivene iz prošloga koraka izvršavanja MPC-a.

7.1. Implementacija

Simulink model LTV MPC-a izgleda isto kao i model adaptivnog MPC-a. Implementacija je prikazana na slici 7.1. te je dodatno označen poredak izvršavanja blokova.

Na slici su označeni blokovi koji se izvršavaju frekvencijom pozivanja MPC-a. Neoznačeni blokovi izvršavaju se prvi jer su vezani za frekvenciju same simulacije. Brojevi u indeksu blokova označavaju kojim redom se izvršavaju blokovi. Jedan korak simulacije počinje računanjem referenci na temelju trenutnog puta (s) te ažuriranjem stanja vozila na temelju upravljačkih signala iz prošlog koraka [1]. Potom se izvršava MPC logika. Može se vidjeti da se prvo uzimaju prošle vrijednosti optimalnog upravljačkog signala za MPC te za predikciju radnih točki Runge-Kutta metodom. Zatim se izvršava Runge-Kutta predikcija koja na temelju prijašnjih ulaza, budućih zakrivljenosti putanje te trenutnog stanja radi predikciju varijabli stanja. Predviđene varijable stanja ulaze u MPC blok gdje se računaju linearizirane matrice **A** i **E** na temelju izraza dobivenog iz modela

[1] Izgled cijele simulacije nije prikazan na slici 7.1.



Slika 7.1. Implementacija LTV MPC-a u razvojnom okruženju *Simulink*.

vozila. Nakon toga vrši se optimizacija te se odabire prvi upravljački signal. Odabirom upravljačkog signala završava jedan korak simulacije.

U kriterijsku funkciju LTV MPC-a, uz već navedene težine, dodan je izraz:

$$(\mathbf{u}_k - \mathbf{u}_{k-1})\mathbf{V}(\mathbf{u}_k - \mathbf{u}_{k-1}). \tag{7.1}$$

Vrijednosti matrice $\mathbf{V}$ definiraju koliko će se kažnjavati derivacije upravljačkog signala. Što je veća vrijednost matrice $\mathbf{V}$, upravljački signal izgledati će glađe. Ovaj izraz je uveden jer je simulacijama utvrđeno da vrlo nagli skokovi upravljačkih signala lako mogu dovesti MPC blizu nestabilnosti. Nagli skokovi ulaznih signala značajno mijenjaju Runge-Kutta predikciju iz koraka u korak stoga je bitno da upravljački signal ne bude vrlo agresivan.

LTV MPC treba početi sa radom kada je vozilo već u nekoj radnoj točki. Runge-Kutta predikcija tada ne može uzeti vrijednosti prošlih ulaza jer ih nema. U tom slučaju za vrijednosti prošlih ulaza uzimaju se nule. Prvih par izvođenja LTV MPC-a radne točke za matrice **A** i **E** neće biti točne, no to ne smeta radu LTV MPC-a. Već nakon par izvršavanja Runge-Kutta predikcija i predikcija LTV MPC-a se usklade.

7.1.1. Optimalna trajektorija

U simulaciji potpoglavlja 7.2.4., za reference se neće koristiti *mješoviti manevar* kao do sada nego brzine i putanje dobivene optimiranjem trajektorije [6][3]. Nakon rješavanja MLTP problema dobivene su optimalne varijable stanja kroz trajektoriju zadanu rubovima staze. Optimalne vrijednosti diskretizirane su po putu s na način da su definirane svaka 3 m. Ovako udaljene reference mogu se također prosljeđivati MPC-u na način da je kroz cijelu predikciju referenca konstantna te se možda promjeni nekoliko puta ovisno o vremenu predikcije. MPC idealno zahtijeva da mu se reference mijenjaju u svakom koraku predikcije za što bolje upravljanje vozilom jer se i u stvarnosti reference mijenjaju kontinuirano. Zbog toga se dobivene reference iz MLTP optimizacije prije izvršavanja interpoliraju.

Odabrana metoda interpolacije parametara s, n, ξ, v i κ je kubični splajn. Razlog odabira kubičnog splajna je zato što osigurava neprekinutost druge derivacije (C^2). Neprekinutost druge derivacije dolazi do izražaja u izrazima za prelazak iz koordinatnog sustava u krivolinijski te obrnuto. Vrijeme je interpolirano linearnom interpolacijom. Reference za LTV MPC u simulaciji optimalne trajektorije računaju se na način da se trenutno prijeđen put s upari sa jednakim putem iz optimalne putanje s_r. Nakon toga se idućih N referenci odabire u jednakim vremenskim razmacima T.

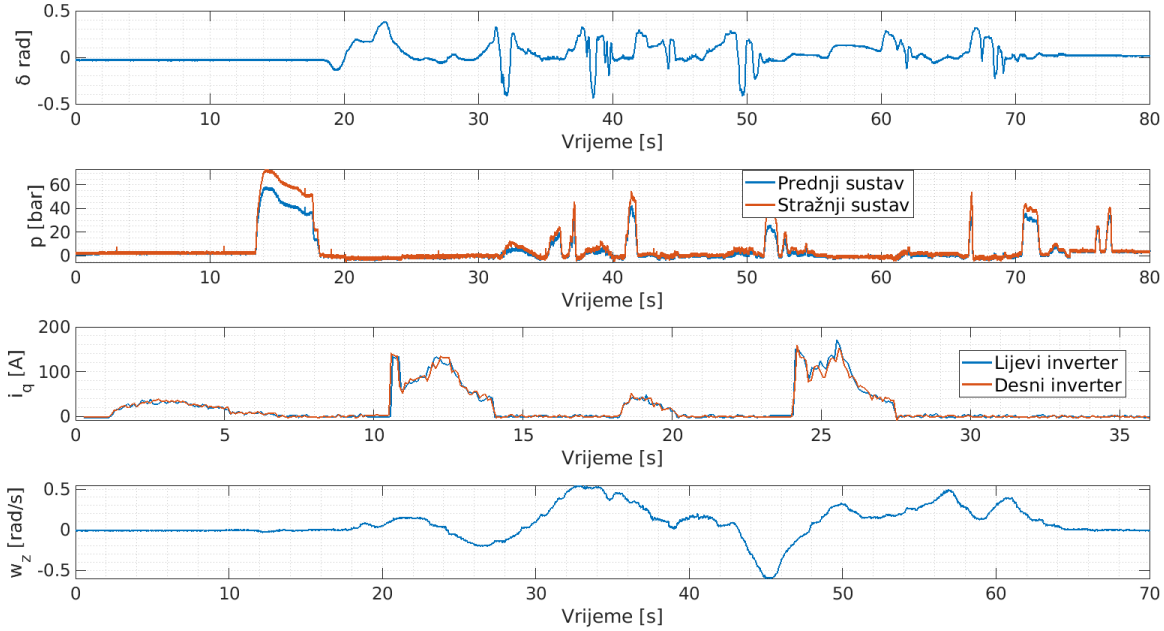
Točke putanje definirane u koordinatnom sustavu mogu se prebaciti u krivolinijski sustav za potrebe simulacije MPC-a. Zakrivljenost putanje κ može se izračunati koristeći izraz [16]:

$$\kappa(s) = \frac{\frac{d^2y(s)}{ds^2}\frac{dx(s)}{ds} - \frac{d^2x(s)}{ds^2}\frac{dy(s)}{ds}}{\left(\left(\frac{dx(s)}{ds}\right)^2 + \left(\frac{dy(s)}{ds}\right)^2\right)^{3/2}}. \tag{7.2}$$

7.1.2. Analiza šuma

U simulaciji potpoglavlja 7.2.3. testiraju se performanse LTV MPC-a s obzirom na šum i kašnjenja. Kašnjenja su definirana u poglavlju 4.5., a šum je dobiven iz stvarnih podataka vožnje. Varijable stanja s, n, ξ, v i β dobivaju se direktno iz SLAM-a ili nezavisnih estimacija brzine vozila i kuta β unutar ECU modela. Na njih je dodan proizvoljan mali šum pošto sve varijable prolaze kroz određene filtre estimacije. Varijable w_z, i_q, δ i p očitavaju se sa senzora stoga postoje podaci iz prijašnjih vožnji vozila VulpesD nad kojima će šum biti analiziran i korišten u simulacijama.

Iz različitih vožnji uzete su snimke koje imaju puno vremena vozila u mirovanju. Signali koji su analizirani prikazani su na slici 7.2.



Slika 7.2. Signali nad kojima je izvršena analiza šuma.

Na početku ili kraju signala vozilo je u mirovanju te na tim mjestima djeluje samo bijeli šum. Na mjestima u kojima se pojavljuje samo bijeli šum izvršeno je računanje standardne devijacije signala. Na taj način su dobivene standardne devijacije šuma koje su upotrebljavane pri simulaciji šuma. Dobivene standardne devijacije prikazane su u tablici 7.1.

Najbolji filtar za uklanjanje bijelog šuma je pomični prozor (eng. *Moving Average*). Stoga su u simulaciji svi signali filtrirani pomičnim prozorom s ciljem da se što više smanje smetnje bijelog šuma na djelovanje MPC-a. Pomični prozor ima jedan parametar, a

Tablica 7.1. Vrijednosti standardne devijacije bijelog šuma signala.

Varijable	δ	p	i_q	w_z
σ	0.0033	0.4637	1.6925	0.0023

to je širina prozora. Što je širina prozora veća, veći dio signala se usrednjava te je signal bolje filtriran. No odabirom šireg prozora veći dio korisnog signala se usrednjava te se time stvara kašnjenje signala. Kašnjenje signala isto utječe na rad MPC-a. Stoga je bitno pronaći ravnotežu između dobrog filtriranja bijelog šuma uz što manje zakašnjenje signala.

7.2. Rezultati

U ovome odjeljku analizirane su performanse LTV MPC-a pri raznim referencama, aktiviranjima ograničenja te šumovima i kašnjenjima. LTV MPC ima vrlo dobar odziv na promjenjive signale zakrivljenosti staze. U narednim simulacijama uzima se u obzir zakrivljenost puta. zakrivljenost κ generirana je sa sinusoidnim signalom u simulacijama *mješovitog manevra*. Takva staza predstavlja oblik zmijice koja je blaža jer joj se polumjeri postepeno mijenjaju. Minimalni radijus u simulacijama *mješovitog manevra* je 20 m. MPC vrlo dobro radi i sa vrlo niskim radijusima od 5 m što je donja granica.

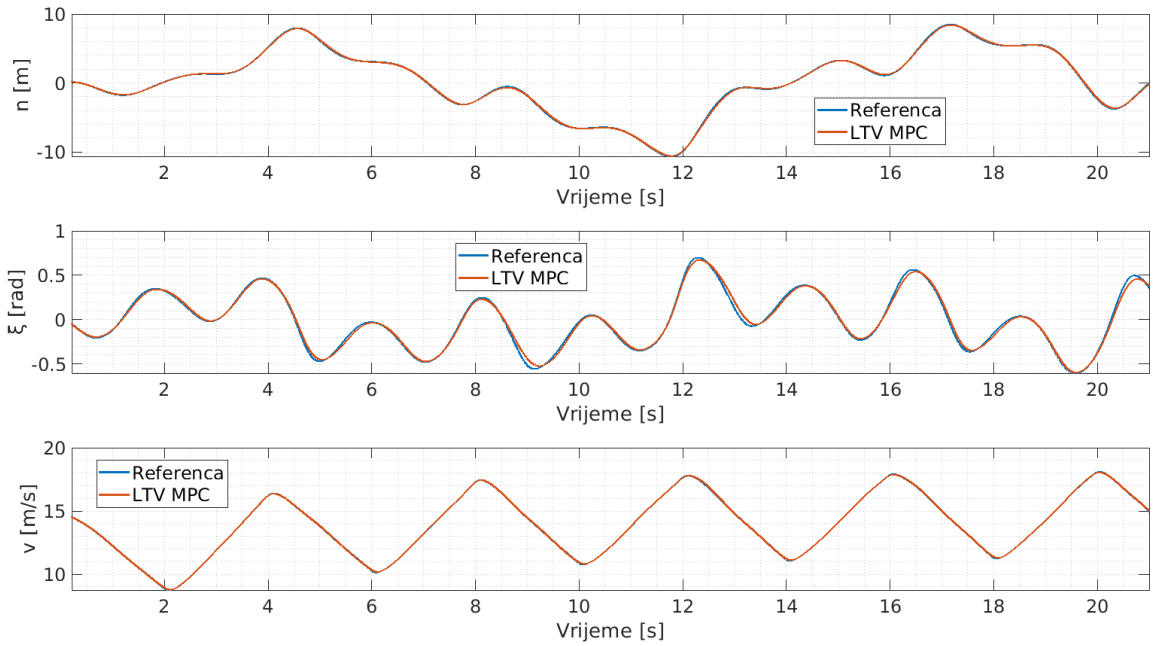
7.2.1. Simulacija 4 - Idealno praćenje reference

U ovoj simulaciji prikazan je rad LTV MPC-a bez dosezanja ograničenja i bez šuma te kašnjenja. Cilj ove simulacije je pokazati točnost samog algoritma. Parametri korišteni pri simulaciji prikazani su u tablici 7.2.

Tablica 7.2. Vrijednosti konstanti simulacije 4.

Konstante	Q_i	R_i	V	S	T	N
Vrijednosti	1, 1, 1	0.1, 0.1, 0.1	0.1	2	20 ms	60

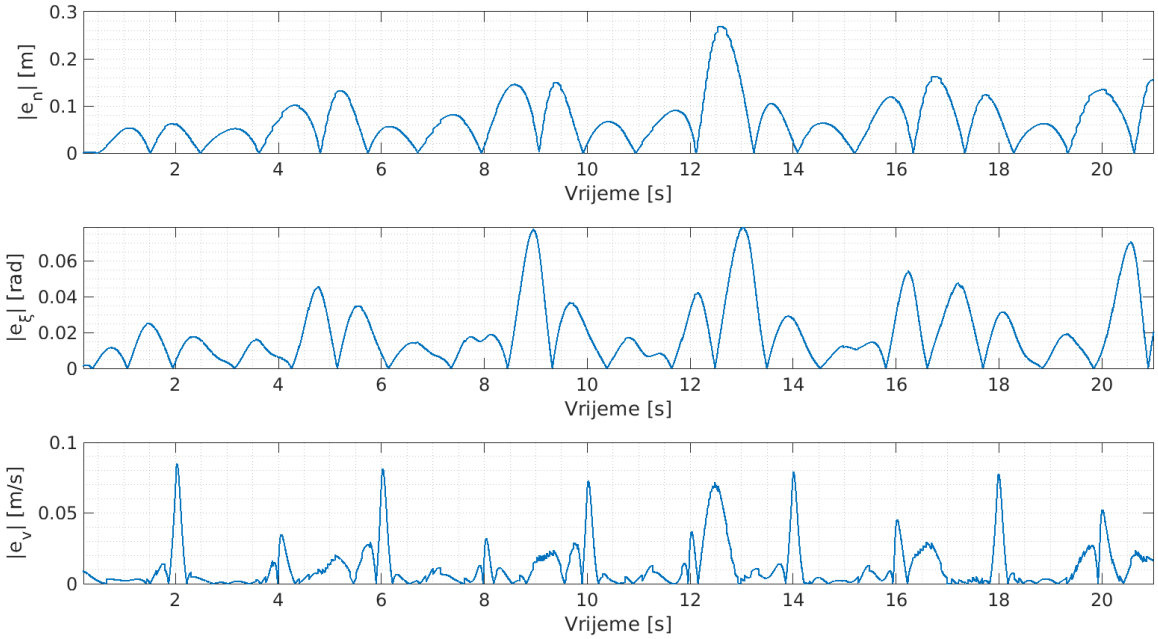
Korišteno je nisko vrijeme diskretizacije od 20 ms. Zbog toga potrebno je imati veliki predikcijski horizont kako bi se dobilo značajno vrijeme predikcije. Rezultati varijabli stanja koji prate reference prikazani su na slici 7.3.



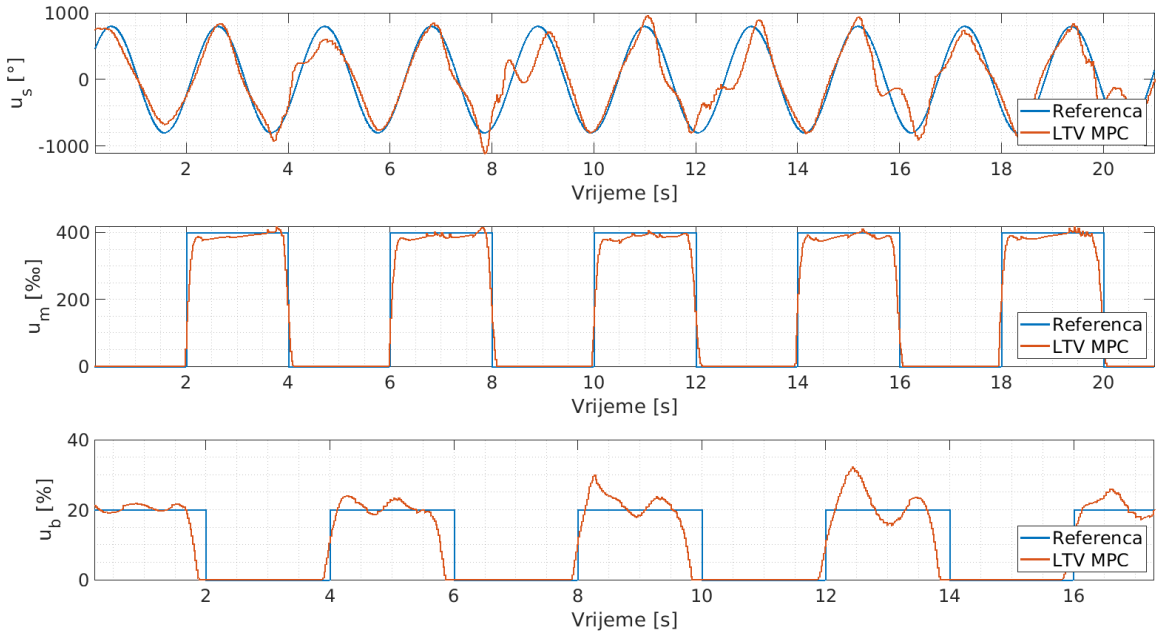
Slika 7.3. Praćenje reference varijabli stanja LTV MPC-a simulacije 4.

Za razliku od adaptivnog MPC-a, LTV MPC ne samo da vrlo dobro radi sa signalima smetnje već gotovo savršeno prati referencu. Apsolutne greške reference i praćenih varijabli stanja prikazani su na slici 7.4. Vozilo odstupa od zadane putanje za najviše 0.3 m i to pri stalnom skretanju i brzini od skoro 20 m/s, što je iznimno točno. MPC čak uspijeva i gotovo savršeno pratiti brzinu vozila pri čemu je najviše odstupanje od 0.1 m/s. Iako je MPC-u zadano da prati generiranu putanju, upravljački signal je gotovo isti signalu s kojim je generiran *mješoviti manevar*. Usporedba upravljačkih signala može se vidjeti na slici 7.5.

Za LTV MPC, simulacijama je dobiveno da se odlazak prema nestabilnom području javlja puno brže nego kod adaptivnog MPC-a. Za vrijeme diskretizacije veće od 30 ms upravljački signali počinju divergirati, a za vrijeme veće od 40 ms problem je gotovo neriješiv. Zbog toga vremena diskretizacije moraju biti vrlo niska, što rezultira prilično velikim predikcijskim horizontima za postizanje velikog vremena predikcije.



Slika 7.4. Apsolutna greška varijabli stanja LTV MPC-a simulacije 4.



Slika 7.5. Usporedba upravljačkog signala LTV MPC-a i referentnog upravljačkog signala simulacije 4.

7.2.2. Simulacija 5 - Aktivacija ograničenja

Cilj ove simulacije je testirati vladanje MPC-a na početno odstupanje od reference te aktiviranje granica. U početku simulacije vozilo ima brzinu od 8 m/s, što je daleko od željene brzine od 15 m/s. Da bi situacija bila još teža, vozilo je u početku udaljeno 1 m od referentne putanje te se kreće u suprotnom smjeru od nadolazećeg zavoja za 0.3 rad

ili približno 20°. Također je postavljeno ograničenje da vozilo ne smije prijeći apsolutni iznos kutne brzine (w_z) za više od 0.8 rad/s. Uz ograničenje varijable stanja koje ne prati referencu, stavljena su ograničenja na brzinu vozila te na udaljenost vozila od referentne putanje. Kroz cijelu simulaciju vozilo ne smije preći brzinu od 16 m/s te se ne smije udaljiti više od 9 m naspram referentne putanje prema desno.

Parametri sa kojima je izvršena simulacija prikazani su u tablici 7.3. Koristi se iznimno nisko vrijeme diskretizacije kako bi model bio što točniji. Pri tako niskom vremenu diskretizacije, garantirano je da upravljački signali neće biti nestabilni niti u kojem trenutku.

Tablica 7.3. Vrijednosti konstanti simulacije 5

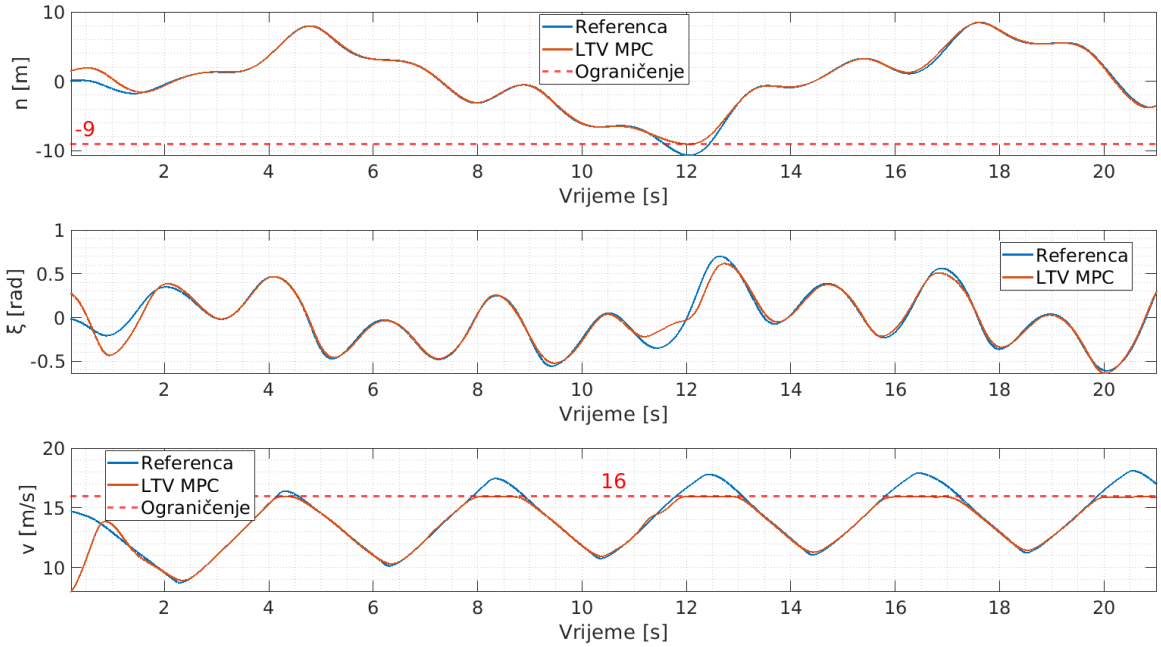
Konstante	Q_i	R_i	V	S	T	N
Vrijednosti	1, 1, 0.1	0.1, 0.1, 0.1	0.5	2	10 ms	150

Da bi se vozilo vratilo na referentnu putanju potrebno je veliko vrijeme predviđanja. Za male devijacije od referenci vozilo se može u kraćem vremenu vratiti na referencu. U ovoj simulaciji vozilo se nalazi u vrlo specifičnoj situaciji u kojoj se može naći nakon izlazak sa staze ili rotiranja vozila proklizavanjem. Za takve situacije potrebno je duže vrijeme da se vozilo vrati na stazu. Radi toga koristi se predikcija od 1.5 s. Za vrlo niske vrijednosti vremena predikcije, algoritam optimizacije ne može u kratkom roku naći optimalno rješenje za povratak na stazu što rezultira greškom u optimizaciji. Rezultati praćenja referenci prikazani su na slici 7.6.

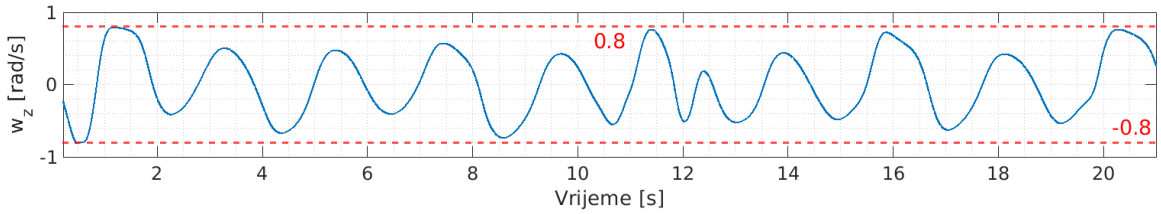
U vremenu kraćem od 2 s vozilo se vratilo na putanju te željenu brzinu. Pri vraćanju na putanju te kroz ostatak simulacije nije se prešlo ograničenje na kutnu brzinu vozila (slika 7.7.). Bez ograničenja na kutnu brzinu, vozilo pri vraćanju na putanju obično dosegne vrijednosti od 1 rad/s u oba smjera.

Vrijednost matrice **Q** koja predstavlja težinu za brzinu vozila stavljena je na nisku vrijednost. Na taj način MPC će prioritizirati vraćanje na putanju umjesto praćenja brzine. Ako se vozilo krene udaljavati od putanje MPC će čak usporiti vozilo da se vrati na putanju. To ne bi bilo moguće kada bi se greška na brzinu vozila jednako otežavala.

Granice na varijable stanja ostvarene su kao meka ograničenja (jednadžba 4.1.). Va-



Slika 7.6. Praćenje reference varijabli stanja LTV MPC-a simulacije 5.

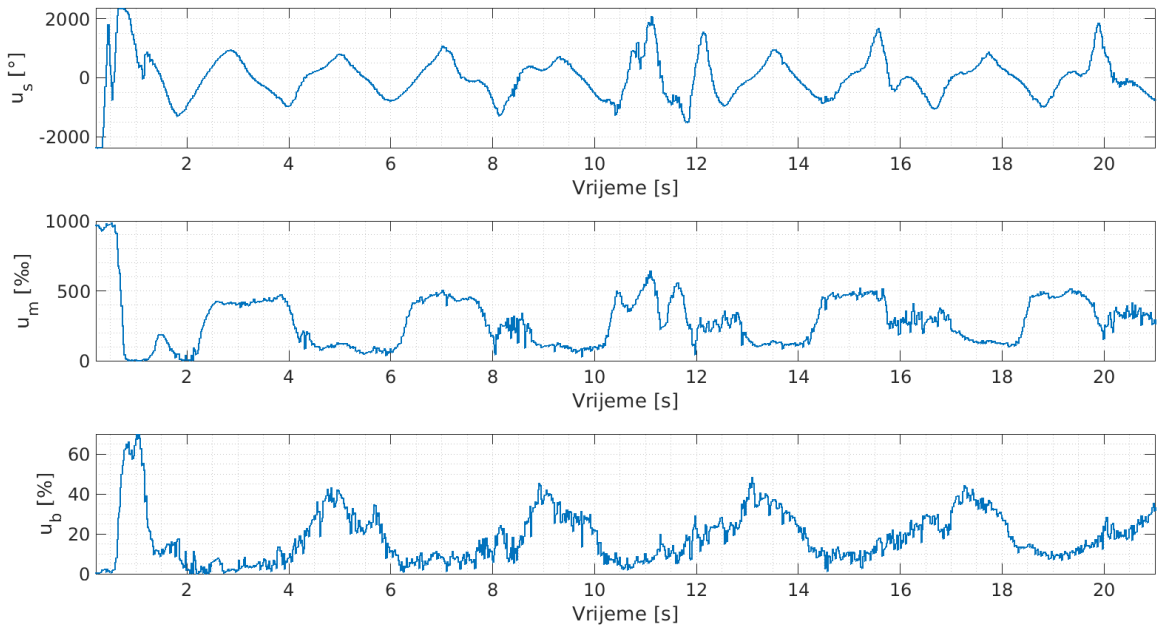


Slika 7.7. Praćenje ograničenja varijable stanja LTV MPC-a simulacije 5.

rijabla ϵ ostvarena je u linearnom obliku, jer kvadratni oblik pogoršava vladanje MPC-a u simulaciji. Za sve varijable stanja bolje je proglasiti meka ograničenja. Pri dostizanju tvrdih ograničenja, MPC je sposoban držati varijablu stanja čak i na treću decimalu od ograničenja kroz neko vrijeme. Čim se granica prijeđe, makar na trećoj decimali, događa se greška u optimizaciji. Ako se koeficijent λ postavi na vrlo visoku vrijednost, meka ograničenja će se ponašati kao tvrda, ali neće se dogoditi greška pri optimizaciji. Primijećeno je da MPC krene ranije reagirati na meka ograničenja nego na tvrda.

Na ulazne signale je bolje staviti tvrda ograničenja, jer aktuacija može imati velike posljedice pri prelasku ograničenja. Tvrda ograničenja upravljačkih signala ne utječu na rad MPC-a i neće se prijeći jer su ograničenja na ulaze uvijek poštivana u optimizaciji. Ulazni signali također prate ograničenja i prikazani su na slici 7.8.

U trenucima ograničenja na brzinu gdje se treba držati konstantna brzina nije intuitivno koliko treba kočiti a koliko ubrzavati. MPC u tim trenucima u isto vrijeme koči i



Slika 7.8. Upravljački signali LTV MPC-a simulacije 5.

ubrzava. Do toga dolazi radi načina definiranja ograničenja na ulaze koja su razrađena u poglavlju 7.2.6.

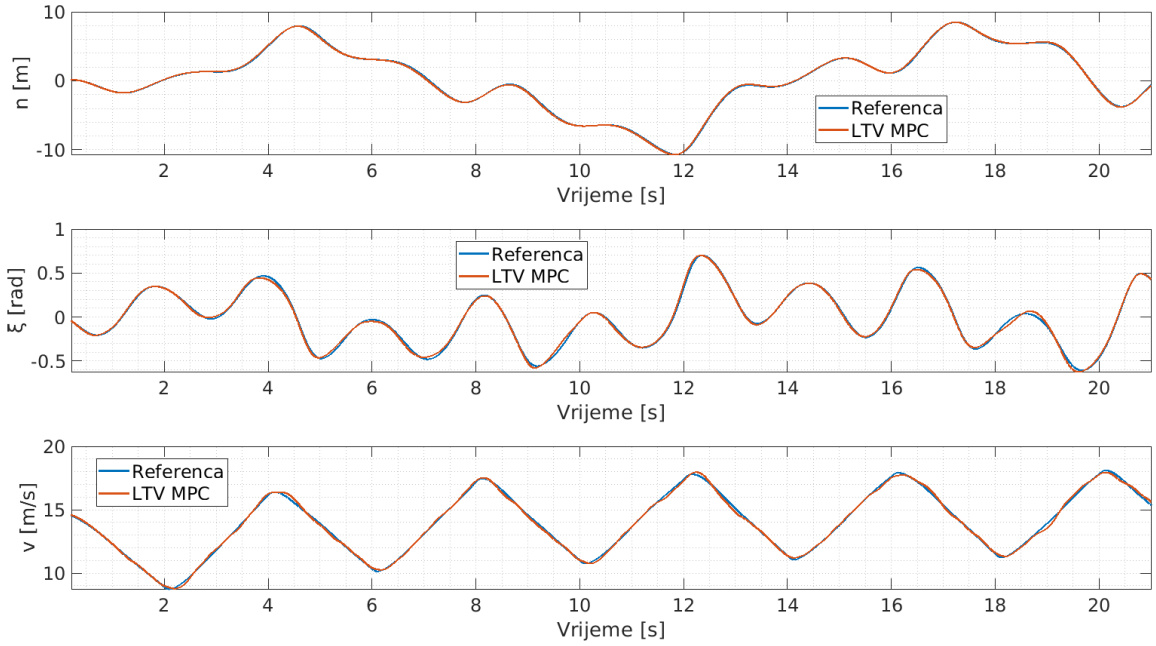
7.2.3. Simulacija 6 - Analiza šuma i kašnjenja

Simulacija *mješovitog manevra* analizirana je uz prisustvo šuma te kašnjenja. U isto vrijeme djeluju sva definirana kašnjenja te šumovi obrađeni u poglavlju 7.1.2. Ova simulacija u velikoj mjeri odgovara stvarnom korištenju LTV MPC-a unutar vozila VulpesD. Vrijeme predikcije je vrlo kratko i iznosi 0.6 s. Težine su gotovo nepromijenjene u odnosu na simulaciju 7.2.1. što je prikazano u tablici 7.2.

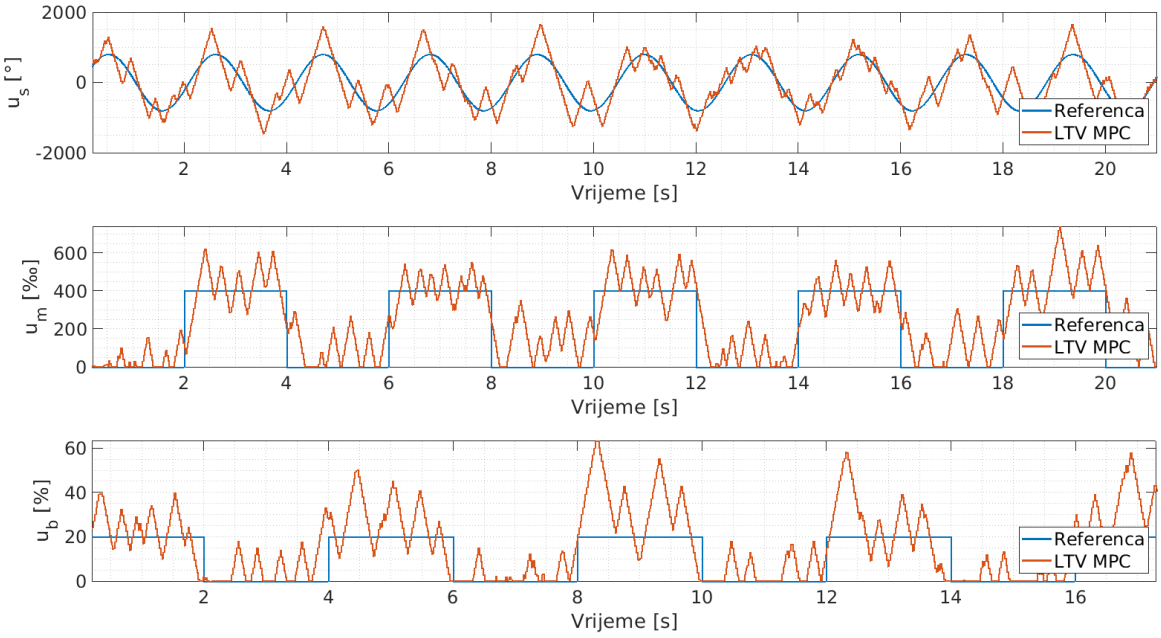
Tablica 7.4. Vrijednosti konstanti simulacije 6.

Konstante	Q_i	R_i	V	S	T	N
Vrijednosti	1, 1, 0.1	0.01, 0.01, 0.01	0.5	2	30 ms	20

Rezultati su prikazani na slici 7.9. Referenca se i dalje iznimno dobro prati usprkos šumu i kašnjenju. Mogu se vidjeti povremena odstupanja od reference brzine iz razloga što je šum struje i_q i tlaka p prilično velik. Najveći iznosi šuma iznosi čak 1 % maksimalne vrijednosti signala. Šum i kašnjenja ne dolaze značajno do izražaja pri praćenju referenci, ali značajno mijenjaju upravljačke signale što je vidljivo na slici 7.10.



Slika 7.9. Praćenje reference varijabli stanja LTV MPC-a simulacije 6.



Slika 7.10. Usporedba upravljačkog signala LTV MPC-a i referentnog upravljačkog signala simulacije 6.

Upravljački signali poprimaju pilasti oblik. U stvarnoj vožnji vozilo bi pratilo referencu ali bi se jako čudno ponašalo interpretirajući ove ulaze. Ovakvi upravljački signali pri velikim brzinama mogu čak izbaciti vozilo sa staze. Pilasti oblici javljaju se samo pri djelovanju šuma, a djelovanje kašnjenja samo pojačava amplitudu.

Kašnjenja u očitavanju trenutnih varijabli stanja i nemaju toliki utjecaj kao što ima

kašnjenje upravljačkog signala. To je uvelike utjecano modeliranjem aktuatora PT1 članom. PT1 član ima vrlo nagli odziv na step promjenu referenci. Boljim modeliranjem aktuatora može se čak uključiti i kašnjenje pri aktuaciji te se može potpuno neutralizirati utjecaj kašnjenja upravljačkog signala koji ima vrlo velik utjecaj u ovoj simulaciji.

Signali varijabli stanja vozila filtrirani su pomičnim prozorom. Struja i tlak kočenja filtrirani su većim prozorom koji usrednjava signale dobivene do prije 40 ms. A ostali signali koji nemaju značajan šum imaju prozor od 10 ms. Vremenska širina prozora uzrokuje kašnjenje signala otprilike za istu vrijednost. Zbog toga je signal tlaka značajno zakašnjen. No za slučaj razvijenog MPC-a, bolje je uzrokovati veće kašnjenje varijabli stanja nego dozvoliti velik šum. Vrlo mala širina prozora uzrokuje da su susjedna očitanja varijabli stanja vrlo različita zbog šuma, čak do te mjere da MPC može zaključiti da vozilo skreće prema desno, dok u stvarnosti skreće prema lijevo. Velika širina prozora uzrokuje veliko kašnjenje i deformaciju signala sa kojom MPC ne može raditi.

Vrijeme diskretizacije modela isto uvelike utječe na vladanje MPC-a. Odabiranjem većeg vremena diskretizacije, susjedni uzorci varijabli stanja biti će dovoljno vremenski razmaknuti da filtar stigne filtrirati šum. Na vremenima diskretizacije ispod 20 ms MPC nije mogao nalaziti rješenja optimizacije. Zbog šuma se također smanjuje i iskoristivo vrijeme predikcije, jer lošim očitavanjem trenutnih stanja predikcija postaje lošija kroz vrijeme. Najveće vrijeme predikcije pri kojem je MPC mogao raditi bez grešaka u optimizaciji je 1 s.

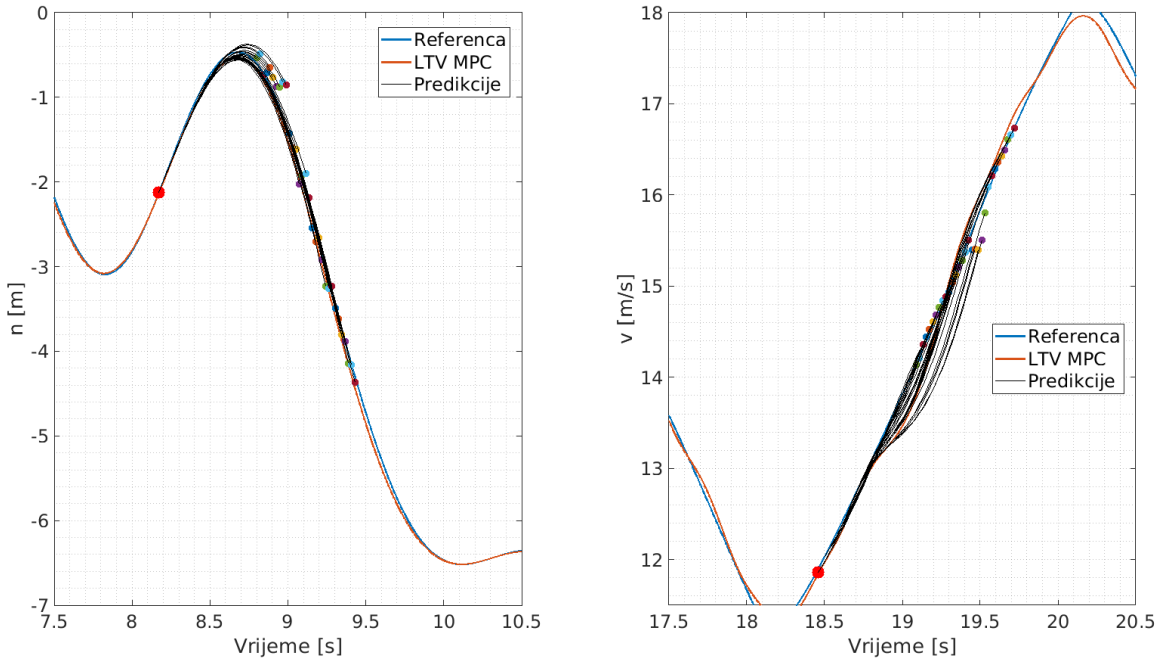
MPC je testiran i uz uvođenje faktora koji smanjuje vrijednost daljih predikcija. Parametar η implementiran je na način da sa svakim korakom predikcije smanjuje vrijednosti koeficijenata matrice $\mathbf{Q}$:

$$J = \sum_{k=0}^{N-1} \left(\mathbf{e}_k^\top \mathbf{Q} \eta^k \mathbf{e}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \right) + \mathbf{e}_N^\top \mathbf{S} \mathbf{e}_N + \epsilon^\top \lambda \epsilon^\top. \tag{7.3}$$

Zbog toga, predikcije koje su dalje u budućnosti uzete su manje u obzir jer su netočne zbog šuma. Parametar η mora imati vrijednosti između 0 i 1. Vrijednost 1 ne smanjuje vrijednosti predikcija, a što je vrijednost niža to se više smanjuju težine predikcija. Rezultati dobiveni ovim načinom nemaju velik značaj na rad MPC-a te su u nekim slučajevima

rezultati i lošiji.

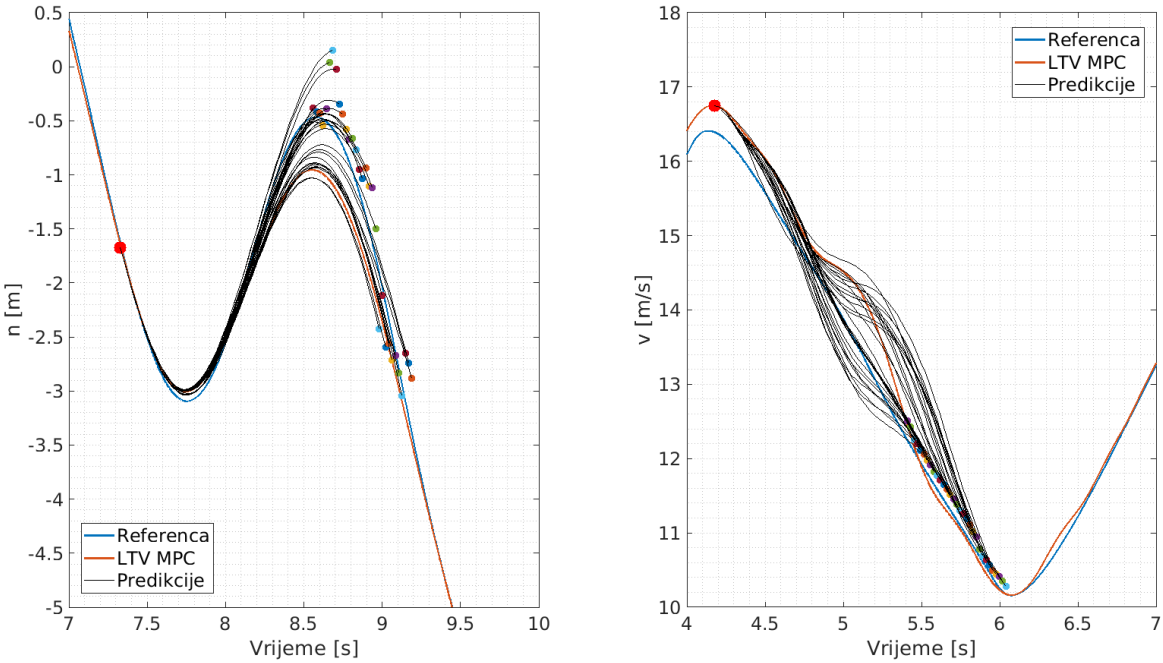
Predikcije MPC-a vizualizirane su na slici 7.11. Isječak na slici pokazuje 30 uzastopnih predikcija MPC-a. Vrijeme predikcije je 0.6 s, a početak prve predikcije označen je velikom crvenom točkom. Predikcije su označene tankim crnim linijama, a završeci predikcija točkama u različitim bojama.



Slika 7.11. Prikaz predikcija LTV MPC-a uz vrijeme predikcije 0.6 s.

Za predikcije varijabli n i nema toliko grešaka jer signali koji utječu na nju nemaju velik šum. Veliki šum signala i_q i p značajno utječu na predikciju brzine vozila. Predikcije su nakon određenog trenutka krenule u krivom smjeru jer su trenutno i prethodno stanje varijabli struje i tlaka krivo očitani zbog šuma.

Za još veće vrijeme predikcije od 1.2 s rezultati predikcije prikazani su na slici 7.12. Na većem predikcijskom horizontu šum više dolazi do izražaja. Završeci predikcija varijabli n i v značajno se razlikuju iz koraka u korak. Za varijablu v može se vidjeti da je u određenom trenutku MPC krenuo sa netočnim predikcijama, te je u konačnici upravljao vozilo na način da postigne brzinu koja je netočna.



Slika 7.12. Prikaz predikcija LTV MPC-a uz vrijeme predikcije 1.2 *s*.

7.2.4. Simulacija 7 - Optimalna trajektorija

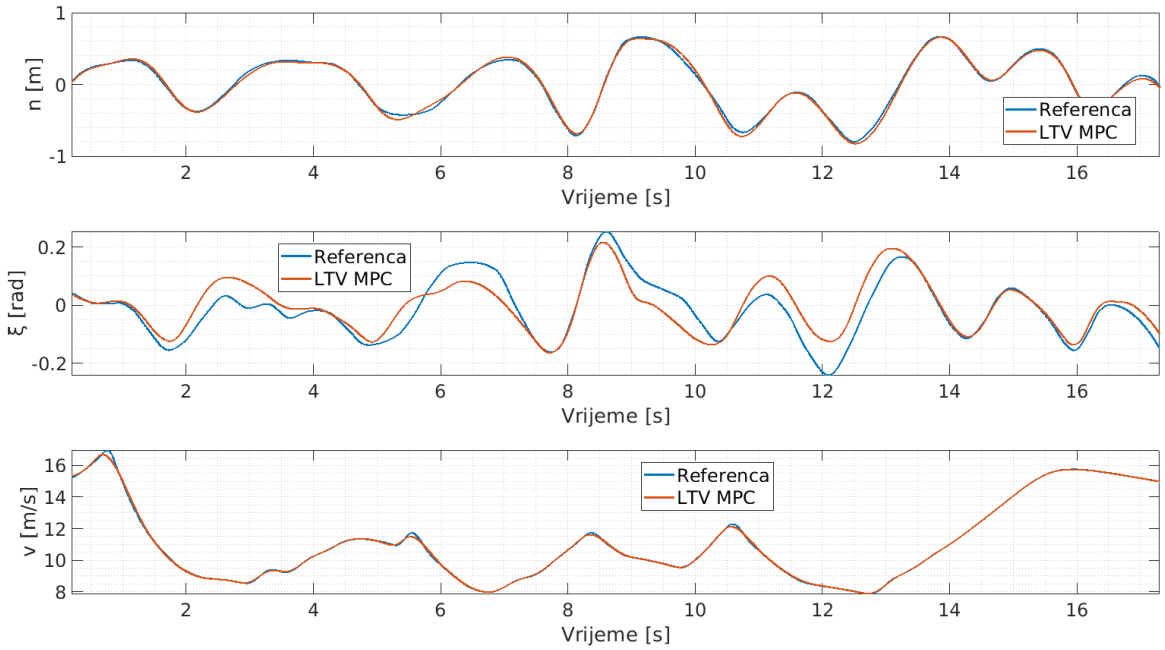
Dosadašnje simulacije su sve bile simulirane *mješovitim manevrom*. U ovoj simulaciji reference su dobivene iz optimalne trajektorije. Staza nad kojom je rađena optimizacija predstavlja pravu stazu na kojoj se vozilo 2018. godine na natjecanju u Njemačkoj [10]. U ovoj simulaciji nisu korišteni zašumljeni signali niti kašnjenja. Korišteni parametri MPC-a prikazani su u tablici 7.5.

Tablica 7.5. Vrijednosti konstanti simulacije 7.

Konstante	Q_i	R_i	V	S	T	N
Vrijednosti	1, 3, 0.1	0.1, 0.1, 0.1	0.5	2	20 *ms*	70

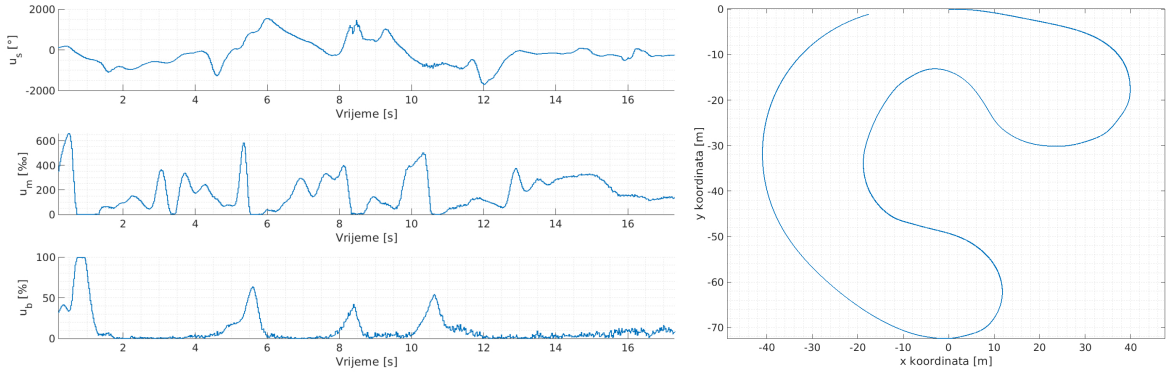
Rezultati simulacije prikazani su slikom 7.13. MPC i dalje uspijeva jako dobro pratiti reference. Parametar Q_ξ stavljen je na veću vrijednost tako da se pokuša bolje pratiti referenca varijable ξ. Model vozila kojim su generirane optimalne reference ima modeliranu puno bržu aktuaciju skretanja. Zbog toga je trenutnom modelu vozila teže pratiti referencu ξ. MPC sa trenutnim modelom ne može raditi tako brza skretanja, ali i dalje vrlo dobro prati varijablu n koja je najbitnija.

Upravljački signali, kao i rekonstruirani izgled staze prikazani su na slici 7.14. Staza



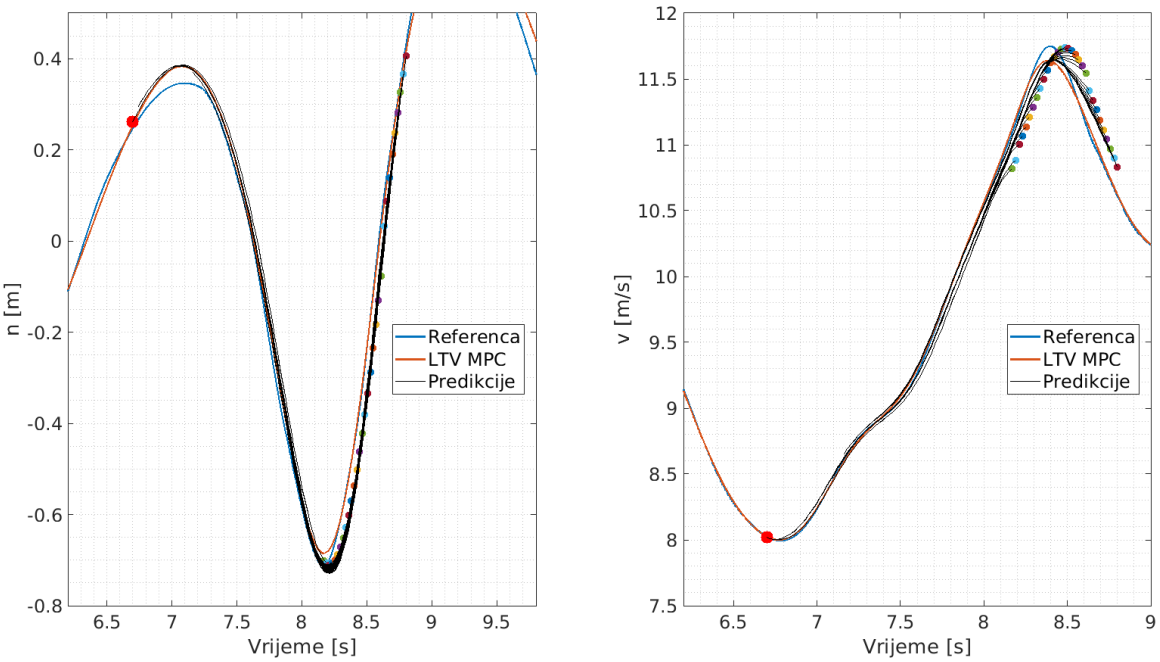
Slika 7.13. Praćenje reference varijabli stanja LTV MPC-a simulacije 7.

započinje i završava u ishodištu koordinatnog sustava.



Slika 7.14. Upravljački signali LTV MPC-a i izgled staze simulacije 7.

Vrijeme predikcije iznosi 1.4 *s* što je prilično daleka predikcija. Na slici 7.15. prikazano je 30 uzastopnih predikcija koje počinju u trenutku označenom crvenom točkom. Predikcije su vrlo točne i koncentrirane, usprkos prilično velikom vremenu predikcije. To znači da je sam algoritam iznimno točan i ima vrlo malo grešaka, a sve greške u predikcijama dolaze iz kašnjenja signala, zašumljenih podataka ili netočnog modela vozila.



Slika 7.15. Prikaz predikcija LTV MPC-a uz vrijeme predikcije 1.4 *s*.

7.2.5. Usporedba algoritama optimizacije

Ovo poglavlje analizira karakteristike optimizacijskih algoritama navedenih u poglavlju 4.4. Navedeni rezultati izvršeni su nad simulacijom optimalne trajektorije bez šuma i kašnjenja. Rezultat optimizacije ovisi o sastavljenom problemu, a pošto je problem kvadratnog oblika ima samo jedan optimum kojeg bi svi optimizatiori trebali pronaći. rezultat optimizacije je gotovo uvijek jednak neovisno o algoritmu, stoga se na naglasak stavlja vrijeme izvođenja optimizacije. To je iznimno bitan faktor jer uz vrlo niska vremena optimizacije može se koristiti dostupnije sklopovlje za izvođenje algoritma te veća vremena predikcije uz manje vrijeme diskretizacije.

Svi algoritmi kvadratne optimizacije nisu dostupni jer YALMIP ima specifične vanjske optimizatore koji se mogu koristiti [8]. Stoga je testirano što više algoritama koji imaju različit način rješavanja problema. Nad potpuno definiranim optimizacijskim problemom (osim korištenja mekih ograničenja) testirana su vremena izvođenja algoritama uz predikcijski horizont od 30 koraka.

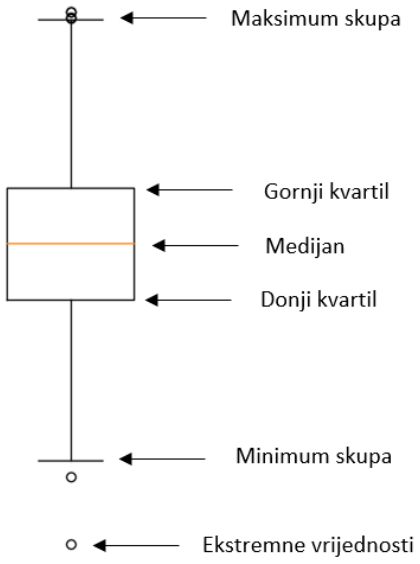
Algoritam koji je dostupan, a predstavlja Douglas-Rachford metodu je OSQP. Za rješavanje problema adaptivnog MPC-a OSQP algoritam je bio najbrži te je zadržavao jednaku brzinu čak i sa vrlo velikim predikcijskim horizontima. No problem LTV MPC-a je

vrlo kompliciran te ga OSQP ne može riješiti.

Za rješavanje problema koji koriste aktivni skup, dostupan algoritam je qpOASES. Problem definiran LTV MPC-om uspijeva riješiti, ali mu za to treba oko 0.8 *s* što je prilično sporo za definirani predikcijski horizont.

Algoritmi temeljeni na metodi proširenog Lagranžijana nisu dostupni unutar YALMIP sučelja. ProxQP algoritam, koji ima najbolje performanse prema slici 4.5., razvijen je tek nedavno te nije dostupan unutar YALMIP sučelja.

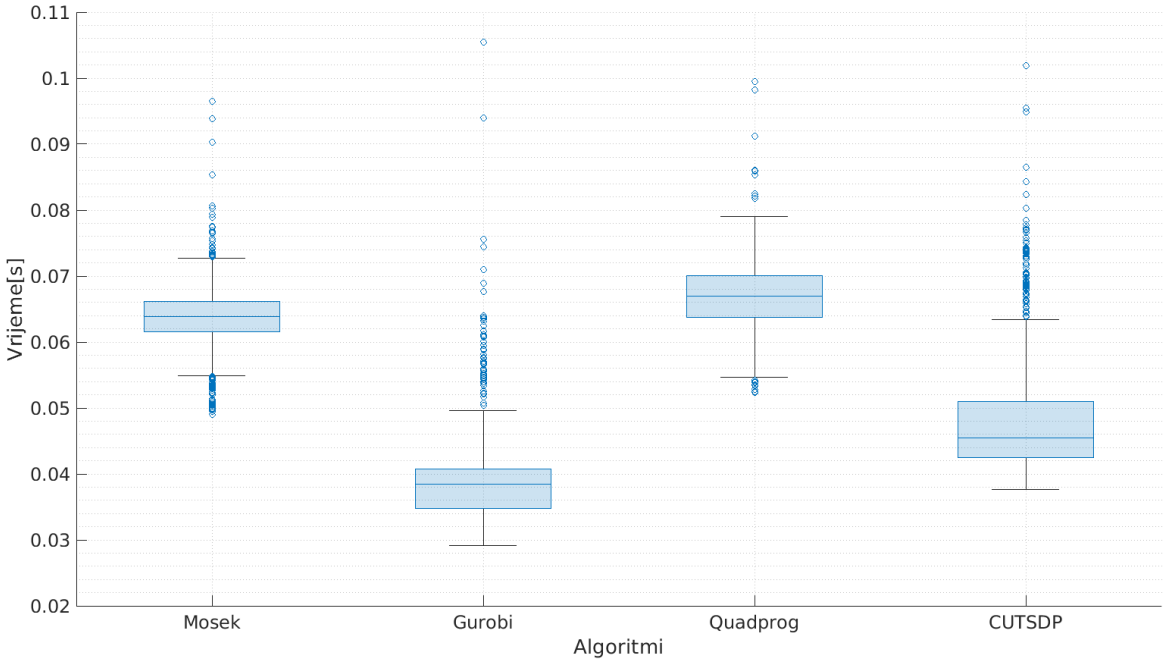
Jedni algoritmi koji uspijevaju riješiti LTV MPC problem u dovoljno brzom vremenu temeljeni su na metodi traženja unutarnje točke te Goldfard-Idnani metodi. Vremena izvođenja algoritama prikazana su na slici 7.17. Graf predstavlja *boxplot* prikaz podataka koji je pogodan za statističku analizu.



Slika 7.16. Prikaz *boxplot*-a.

Medijan predstavlja srednju vrijednost skupa podataka. Donji kvartil dijeli prvu četvrtinu niza podataka od druge tri četvrtine. Gornji kvartil isto tako dijeli prve tri četvrtine skupa podataka od zadnje četvrtine. Takvom definicijom prostor između donjeg i gornjeg kvartila čini 50 % skupa. Ekstremne vrijednosti su vrijednosti koje se ističu i nalaze se daleko od skupa podataka.

Na grafu su podaci jedne simulacije. MPC se u simulaciji pozove oko 1000 puta stoga je u svakom stupcu statistički prikaz tih podataka. U obzir nisu uzeta vremena sastavlja-

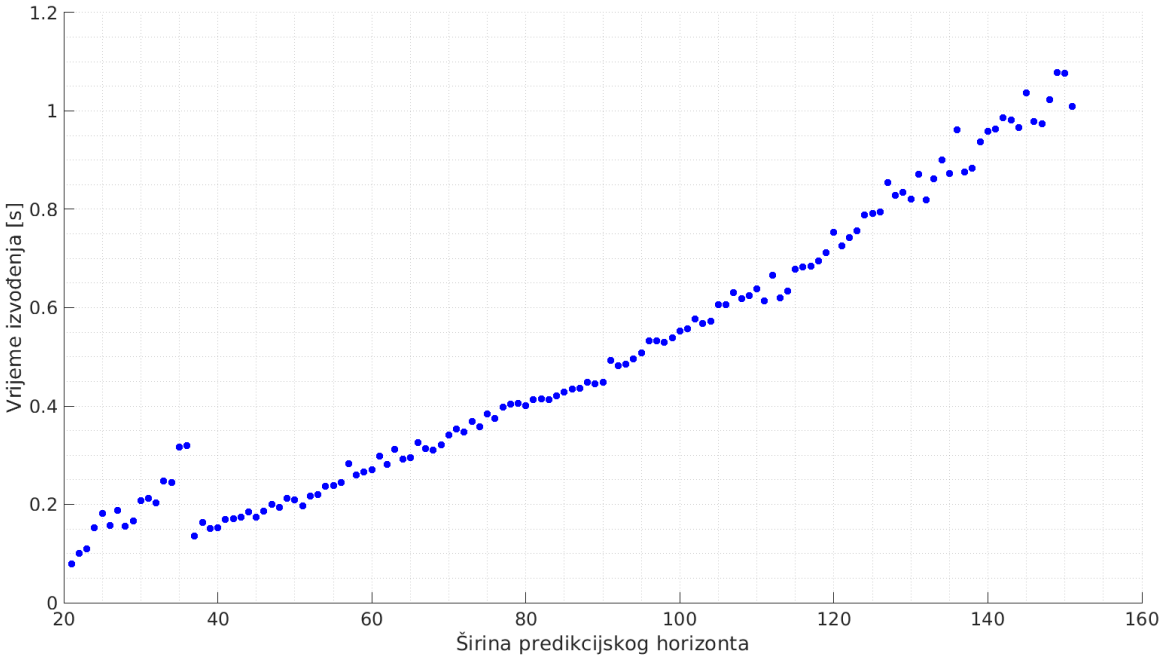


Slika 7.17. Usporedba vremena izvođenja optimizacijskih algoritama.

nja *Kontroler* objekta na početku simulacije. Takva vremena mogu poprimati vrijednosti od 1 *s* pa sve do 8 *s*, ovisno o veličini optimizacijskog problema. CUTSDP je interni YALMIP algoritam koji koristi vanjske optimizatore. Najbrži algoritam u simulaciji je Gurobi, iako su ostali algoritmi vremenski slični. Osim prosječnih vrijednosti izvršavanja algoritama. jako bitni podatci su ekstremne vrijednosti skupa. Iako Gurobi ima najbolje prosječno vrijeme izvođenja, najsporiji je po pitanju maksimuma skupa. A frekvencija izvršavanja MPC-a određuje se prema najsporijem uočenom vremenu izvršavanja optimizacije. Stoga je vrlo bitan podatak koncentriranost vremena izvođenja. Mosek i Quadprog su vrlo konzistentni sa vremenom izvršavanja.

Među algoritmima u grafu prevladavaju algoritmi temeljeni na metodi traženja unutarnje točke što je i preferirana metoda za MPC probleme. Gurobi ne može riješiti optimizacijski problem nakon dodavanja mekih ograničenja i vrlo je nekonzistentan na većim predikcijskim horizontima, a CUTSDP koristi Mosek u pozadini. Stoga su jedine opcije za rješavanje LTV MPC-a Mosek i Quadprog. Ta dva algoritma su također i najbolje implentirana u YALMIP-u te su preporučena za korištenje unutar okruženja. Quadprog je algoritam implementiran u *Matlabu* stoga ne zahtijeva vanjske licence poput Moseka. Radi toga je dugoročno gledajući Quadprog više dostupan *FSB Racing teamu* te je stoga odabran kao algoritam za rješavanje MPC problema.

Sa većim predikcijskim horizontom, količina MPC problema se povećava jer se problem mora riješiti za svaki korak predikcije. Stoga vrijeme izvršavanja algoritma raste sa povećanjem predikcijskog horizonta. U istoj simulaciji, korištenjem Quadprog algoritma mjerena su vremena izvršavanja pri različitim predikcijskim horizontima. Rezultati su prikazani na slici 7.18. Test je izvršen za svaki predikcijski horizont od 20 do 150 koraka te su vremena izvođenja usrednjena. Prema grafu se može zaključiti da vrijeme izvršavanja blago eksponencijalno raste sa širinom predikcijskog horizonta.



Slika 7.18. Ovisnost vremena izvođenja o predikcijskom horizontu.

Većina vremena izvođenja nisu u granici 20 ms koja predstavlja frekvenciju izvršavanja MPC-a od 50 Hz. Vrijeme je mjereno za cijeli LTV MPC ciklus, od dobivanja predikcija Runge-Kutta metodom do odabiranja prvog optimalnog ulaza. Pri svakom izvođenju MPC-a u *Simulinku* poziva se *Matlab* te i to ulazi u mjerenje. Generiranjem C koda te izvršavanjem na ECU, vremena izvršavanja bi trebala biti mnogo niža nego u simulacijama.

Algoritmima za rješavanje MPC problema mogu se inicijalizirati početna rješenja kao što je već spomenuto. Jedini algoritmi koji to mogu raditi kroz YALMIP sučelje su Quadprog i Gurobi. U simulacijama je testirano i inicijaliziranje početnih rješenja onima iz prošlog koraka izvršavanja MPC-a kao što je inače preksa. No u ovome slučaju nisu primijećena značajna ubrzanja rješavanja problema stoga se ta postavka ne koristi.

7.2.6. Ograničenja na ulaze

Dodatno ograničenje koje mora biti implementirano je nemogućnost kočenja i ubrzavanja istovremeno. Inače se implementira nelinearnom jednakosti kao što je već spomenuto.

Iako su gotovo svi kvadratni optimizatori u simulaciji mogli rješavati problem sa jednom nelinearnom jednakosti, problem nije bio dobro riješen. Na mnogim mjestima su vrijednosti aktuacije kočnice i pogonskog motora bili na maksimalnim vrijednostima istovremeno. Zbog toga, izraz koji je umjesto nelinearnog izraza bio korišten pri svim navedenim simulacijama glasi:

$$\mathbf{u}_m + \mathbf{u}_b \leq 1. \tag{7.4}$$

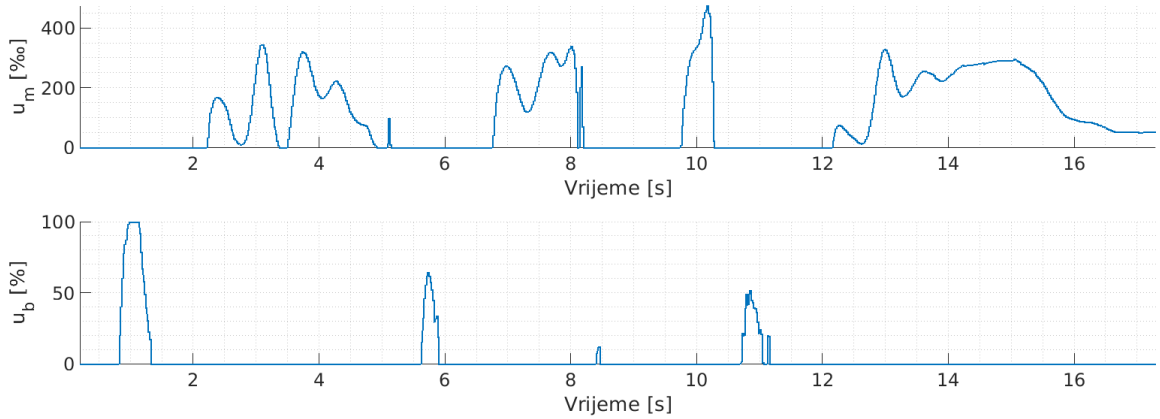
Pri takvoj formulaciji problema, u najgorem slučaju MPC može na pola maksimalne vrijednost aktuirati moment na motoru te na pola maksimalne vrijednosti aktuirati kočnicu. No čak i u tom slučaju vozilo VulpesD će se gotovo sigurno ugasiti.

Linearnim ograničenjima nemoguće je ostvariti zadani uvjet, bez da oba signala mogu koristiti maksimalni potencijal. Jedino rješenje je koristiti uvjetni izraz napisan u matematičkom obliku. No takav zapis nije pogodan za algoritme kvadratnog programiranja. Tako definirani problem mogu riješiti algoritmi mješovitog cjelobrojnog kvadratnog programiranja. Mješoviti cjelobrojni problemi za rješenje mogu koristiti skup realnih brojeva, kao što se inače koristi te skup cijelih brojeva. Na taj način ograničenje se može redefinirati kao:

$$\begin{aligned} \mathbf{u}_m &\leq (1 - b), \\ \mathbf{u}_b &\leq b. \end{aligned} \tag{7.5}$$

U ovom slučaju b je cjelobrojna binarna varijabla koja može poprimati vrijednost 1 ili 0. Ovisno o vrijednosti binarne varijable, samo jedan ulaz može davati vrijednosti. Uz ovakvo definiranje ograničenja, MPC može upravljati vozilom u simulacijama. *Mosek* podržava mješovito cjelobrojno kvadratno programiranje. Na slici 7.19. prikazani su

rezultati simulacije nad optimiziranom trajektorijom. U usporedbi sa slikom 7.14. koja ima linearno ograničenje na aktuatore, mješovitim cjelobrojnim programiranjem ograničenja se prate bez greške.



Slika 7.19. Upravljački signali dobiveni mješovitim cjelobrojnim kvadratnim programiranjem.

No korištenje ovakvog pristupa ima mane. Na trenutke optimizacijski problem zahtijeva i do 10 puta više vremena za rješavanje nego inače. Osim toga, performanse MPC-a su malo lošije te ne prati reference dobro kao inače. Također ne koristi puni potencijal vozila, što se može vidjeti iz upravljačkih signala. Između kočenja i ubrzavanja postoje trenutci praznog hoda koji traju čak i jednu sekundu. U tom razdoblju MPC uopće ne aktuira vozilo u longitudinalnom smjeru.

8. Rasprava

Razvijeni MPC nema puno varijabli za parametriranje, a svi parametri su intuitivni. Predikcijski horizont sa vremenom diskretizacije definira vrijeme predikcije MPC-a. Vrijeme predikcije ne smije biti jako kratko jer MPC ne može konvergirati. Na jednak način ne smije biti vrlo dugačko jer predikcija postaje lošija s vremenom, bilo radi nepreciznog modela, lošeg očitanja trenutnog stanja ili šuma. Povećanjem predikcijskog horizonta problem će se duže rješavati čime gubi izvedbu u stvarnom vremenu. Smanjenjem predikcijskog horizonta uvelike se ubrzava rad MPC-a, ali radi zadržavanja istog vremena predikcije treba se povećati vrijeme diskretizacije. Vrlo nisko vrijeme diskretizacije bolje opisuje model i dinamičke jednadžbe, ali jako izražava bijeli šum prisutan u mjerenjima. Vrlo visoko vrijeme diskretizacije je otpornije na bijeli šum, ali sve lošije opisuje model i dovodi MPC u nestabilno područje.

Skaliranje signala pogreške i težine matrice **Q** usko su povezani. Vrijednosti skaliranja i težine na jednak način otežavaju varijable optimizacije. Varijable skaliranja se jednom definiraju i uvijek su konstantne, a vrijednosti matrice trebaju se parametrirati. Vrijednosti skaliranja trebaju biti maksimalne očekivane vrijednosti greške pojedine varijable. U protivnom, ako su varijable u optimizacijskoj funkciji različite za nekoliko redova veličina problem će biti loše definiran, a time i loše riješen.

Bijeli šum značajno utječe na izgled upravljačkog signala MPC-a. Zbog toga ne treba zanemariti raspored elektroničkih komponenti i filtriranja. Velik šum može biti utjecan i odabirom senzora koji ima preveliko radno područje. Na primjer, senzor tlaka u sustavu kočenja vozila VulpesD dizajniran je za mjerenje puno većih tlakova nego što se mogu ostvariti u vozilu. Zbog toga senzor iskorištava samo dio radnog područja, što za posljedicu ima manju preciznost očitanja te veći utjecaj bijelog šuma.

Nedostatak upravljanja vozilom rješavanjem kvadratnog problema su linearne konstante. Većina ograničenja koja postoje za vozila su defninirana na kružnici ili elipsi što je nelinearno. Takva ograničenja ne mogu se definirati za kvadratnu optimizaciju. Korištenjem Eulerove diskretizacije i modela gume definiranog "magičnom" formulom, model zahtijeva vrlo nisko vrijeme diskretizacije za vjerno definiranje modela. Korištenjem drugih metoda diskretizacije i modela gume, iskoristivo vrijeme diskretizacije bi moglo biti veće, što pospješuje odziv modela na bijeli šum te omogućava upotrebe vrlo malo predikcijskih koraka, što u konačnici ubrzava vrijeme izvođenja MPC-a. Upotreba dinamičkog modela vozila kao posljedicu ima korištenje alternativnog načina kontrole pri niskim brzinama. Konvencionalne tehnike upravljanja poput PID regulatora mogu vrlo dobro upravljati vozilom pri niskim brzinama, što opravdava korištenje MPC-a samo pri srednje velikim te velikim brzinama.

Način zadavanja referenci isto je vrlo bitan. Svi analizirani rezultati usklađeni su s obzirom na prijeđeni put. Zadavanje referenci prema prijeđenom putu skuplja grešku kroz vrijeme. Ako vozilo pri lošem upravljanju uzme puno veći zavoj od referentnog prijeći će veći put. Ako se reference i dalje zadaju po putu, vozilo će primati reference za idealno prijeđeni put, a ne trenutno prijeđeni put. Radi toga, treba se također aktivno računati ortogonalna projekcija prijeđenog puta na referentni put. Na taj način, vozilo će uvijek dobivati ispravne reference s obzirom na put na kojem se nalazi. Jednak problem se javlja i pri zadavanju budućih vrijednosti κ. Pri svim simulacijama korištene su statičke reference, dok se u stvarnosti reference trebaju dinamički prilagođavati prema trenutnom stanju vozila.

Greškom u optimizaciji MPC ne može pronaći optimalna rješenja te su posljedično upravljački signali tada nedefinirani. Greška se može dogoditi samo radi trenutnog utjecaja šuma na mjerenja, a u idućem koraku izvršavanja bi se moglo naći rješenje. Gašenje vozila i ne završavanje discipline na natjecanju radi jedne greške u optimizaciji nikako nije poželjno. Preporuka je da se vozilo ugasi samo ako se greške pojavljuju konstantno u određenom vremenskom razdoblju. Vozilo je samo po sebi prilično sigurno pri autonomnoj vožnji te nema smisla ugasiti vozilo radi oporavljive greške.

Razvijeni LTV MPC je jedan od najboljih linearnih načina kontrole za nelinearni sustav. Idući korak bio bi razviti nelinearni MPC. Za predikciju može se koristiti Runge-

Kutta metoda koja je prilično točna i za velike diskretizacijske korake, ako se koristi na nelinearnom modelu. Nelinearni MPC bi na taj način mogao imati vrlo malo predikcijskih koraka te zbog toga biti vrlo brz. Nedostatak toga bi bio što rješenje koje bi nelinearni MPC vratio nije nužno globalno optimalno. Aktuatori vozila modelirani su PT1 članom. Definiranje aktuatora PT2 članom omogućava modeliranje kašnjenja upravljačkog signala. No povećanjem člana za sva tri aktuatora dodaju se tri nove varijable stanja. Model bi imao matricu veličine 12 umjesto 9. Povećanjem dimenzija matrica varijabli stanja, optimizacijski problem postaje veći i time se usporava vrijeme izvođenja MPC-a.

Vrijednosti konstanti PT1 člana aktuatora modelirani su *apriori* znanjem. Bolji i preferirani način definiranja konstanti bi bio identifikacijom modela. U svakoj VulpesD vožnji snimaju se svi podaci. Identifikacijom modela iz snimljenih podataka mogu se dobiti puno točnije aproksimacije PT1 člana. Čak i mijenjanje svojstava vozila ovisno o vanjskim parametima može se uzeti u obzir. Ako se odziv aktuatora značajno mijenja ovisno o temperaturi, model aktuatora može se identificirati ARMAX modelom koji bi uključio i smetnju poput temperature. Temperature bi se tada mogle prosljeđivati MPC-u koji bi aktivno prilagođavao predikciju aktuatora ovisno o temperaturi aktuatora. Jednako vrijedi i za moment aktuatora skretanja. Moment koji aktuator za skretanje mora savladati ovisi o brzini kojom se vozilo kreće. Pri manjim brzinama vozilo će sporije skretati jer mora savladati veći moment. Identifikacijom te prosljeđivanjem momenta koji aktuator trenutno očitava MPC može prilagoditi predikciju.

Velika prednost razvijenog MPC-a je što koristi prošireni model vozila jednadžbama kretanja. Tako generalizirani model omogućuje praćenje puno općenitijih referenci poput udaljenosti od referente staze. Na taj način se u zatvorenoj petlji može upravljati ne samo vladanjem vozila, nego i pozicijom vozila u prostoru na optimalan način. Za vožnju u prvom krugu centralnom putanjom, kao referenca se može zadati željena brzina koja može biti i velika. MPC će tada odvoziti prvi krug velikom brzinom, vozeći pri tome optimalne trkaće linije uz garanciju da neće izaći sa staze makar morao i usporiti, uz uvjet da percepcija i SLAM rade idealno. U kratko, cijelo kretanje vozila kroz stazu biti će optimalno samo uz zadavanje referenci koje bi trebao pratiti ($n = 0, \xi = 0, v = v_{max}$). To je izniman napredak u odnosu na generiranje referenci korištenjem algoritma čistog slijeđenja koje se odnose na lateralnu i longitudinalnu kontrolu. U tom slučaju MPC bi

upravljao optimalno samo vozilom, a ne njegovim kretanjem kroz prostor.

Za strategiju optimalne vožnje, optimizator trajektorije i razvijeni MPC vrlo dobro rade zajedno. Optimizator trajektorije generira optimalnu trajektoriju poštujući sva ograničenja vozila koja ne mogu biti definirana u linearnom MPC-u. Zbog toga što razvijeni MPC nema ograničenja na iskoristivost gume, ne bi trebao pokušavati voziti brže nego što je optimizator trajektorije zadao. Isto tako ne bi trebao prelaziti maksimalne kutne brzine gibanja koje je optimizator zadao. Na taj način, razvijeni MPC ne mora sadržavati ograničenja vozila direktno, već sama ograničenja optimizatora trajektorije predstavljaju radno područje MPC-a. A dobro kombiniranje dvaju algoritama može se iskoristiti jer koriste identične modele vozila. Preporuka je da se razvijeni MPC koristi prvenstveno za zadržavanje vozila unutar staze, te da se parametri biraju s ciljem što boljeg zadržavanja vozila unutar granica staze. Parametriranje trenutnog MPC-a s ciljem da vozi najbolje vrijeme će vrlo vjerojatno rezultirati rušenjem čunjeva.

Dodatni algoritmi koji pomažu MPC-u su kontrola proklizavanja i vektoriranje momenta. Radi njih MPC može zanemariti longitudinalno proklizavanje u modelu, a lakše postizati željenu kutnu brzinu. Preporuka je da se upravljački signal MPC-a zamjeni direktno sa vozačevim upravljačkim signalom unutar ECU modela. Na taj način upravljački signal MPC-a prolazi serijski kroz kontrolu proklizavanja pa kroz vektoriranje momenta te ispravljen dolazi do aktuatora za pogon. Vrijednost *APPS deratinga* može vrlo loše utjecati na MPC kontrolu. Upravljački signal za pogonske motore bio bi skaliran, bez da je to modelirano unutar MPC-a. Zbog toga je preporuka da kada se vozilo vozi u autonomnom načinu, vrijednosti *deratinga* ne budu definirane van MPC-a, nego u samom modelu MPC-a kroz ograničenja na upravljačke signale.

Prednost razvijanja MPC-a u YALMIP-u je što se može generirati C kod koji ne mora biti implementiran samo u ECU. Isti C kod može se koristiti i na računalu ako se odluči koristiti MPC na autonomnom računalu.

Prelaskom vozila na pogon na sva 4 kotača kontrola i modeliranje biti će puno jednostavnije i točnije. Uz pretpostavku da je zahtjev za moment pogonskih motora gotovo trenutačan te da se koči samo korištenjem regenerativnog kočenja, sve greške pri modeliranju kočnica te pogonskih motora mogu se izbjeći. Pogonski motori mogu puno

točnije ostvariti zadani moment kočenja i ubrzavanja nego raspisivanje sila na kotače kroz diskove kočnica i faktore trenja ili struje motora kao što je izvedeno u ovome radu.

Simulacijama je validiran rad samog algoritma pod utjecajem linearizacije i diskretizacije kao i načinu implementacije MPC-a. Stoga sve greške koje dolaze netočnim modeliranjem vozila nemaju utjecaj u simulacijama. Testiranje MPC-a nad točnijim modelom vozila nije izvršeno. Implementirani nelinearni model vozila vrlo je točan te jednostavni modeli vozila koji su implementirani u *Simulink* sučelju nisu puno točniji. Vrlo detaljan model vozila VulpesD razvijen je unutar simulatora *FSB Racing Teama* za potrebe simulacija mnogih drugih dijelova vozila. Prije implementacije MPC u vozilo VulpesD potrebno je potvrditi da MPC radi zadovoljavajuće i u simulatoru.

9. Zaključak

U ovome radu razvijeni su i testirani adaptivni MPC te LTV MPC. Opisan je model vozila na kojem bi se MPC trebao koristiti da bi se što bolje mogao napraviti matematički model vozila, kao i granice koje su definirane unutar MPC-a. Navedeni su algoritmi upravljanja vozilom koji se inače koriste. Model je lineariziran i diskretiziran metodama koje osiguravaju najbrže moguće pretprocesiranje podataka za MPC, tako da se algoritam može efikasnije izvoditi u stvarnom vremenu. Analizirani su razni optimizacijski algoritmi nakon čega je odabir najboljeg optimizatora potvrđen testiranjem i simulacijom.

MPC je testiran na razne uvjete poput šuma, kašnjenja signala, aktiviranja ograničenja te vraćanja vozila na ispravan put. Model vozila odabran je na način da što točnije modelira vladanje vozila te korelira sa modelom vozila za optimizaciju trajektorije. Komentirane su razne implementacije MPC-a unutar vozila VulpesD te je svaka odluka obrazložena i analizirana.

Razvijeni MPC jako dobro prati reference te u idealnim slučajevima bez šuma i kašnjenja radi gotovo kao nelinearni MPC. Uz samu prednost što upravlja vozilom, MPC direktno upravlja kretanjem vozila u prostoru. Odabir parametara MPC-a vrlo je intuitivan, a kroz simulacije su potvrđeni utjecaji raznih parametara na upravljanje vozila.

Literatura

[1] Formula Student Germany, 2024., registrirani timovi s autonomnim vozilom. [Mrežno]. Adresa: https://www.formulastudent.de/teams/fsd/

[2] ——, "Formula student rules", 2024. [Mrežno]. Adresa: https://www.formulastudent.de/fileadmin/user_upload/all/2024/rules/FS-Rules_2024_v1.1.pdf

[3] F. Christ, A. Wischnewski, A. Heilmeier, i B. Lohmann, "Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients", *Vehicle System Dynamics*, 2019. https://doi.org/10.1080/00423114.2019.1704804

[4] D. S. Lal, V. A., i G. Selvaraj, "Lateral control of an autonomous vehicle based on pure pursuit algorithm", Amritapuri, Kerala, India, 2017.

[5] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, i S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing", u *Proceedings of the 2007 American Control Conference*, Marriott Marquis Hotel at Times Square, New York City, USA, srpanj 11-13 2007.

[6] D. Đurašinović, "Slijeđenje zadane putanje cestovnog vozila zasnovano na modelskom prediktivnom upravljanju", Diplomski rad, Sveučilište u Zagrebu, Zagreb, Hrvatska, kolovoz 2023.

[7] M. Schwenzer, M. Ay, T. Bergs, i D. Abel, "Review on model predictive control: an engineering perspective", *The International Journal of Advanced Manufacturing Technology*, 2021. https://doi.org/10.1007/s00170-021-07682-3

[8] J. Löfberg, “Yalmip : A toolbox for modeling and optimization in matlab”, u *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[9] M. Hovd, “A brief introduction to model predictive control”, ožujak 2004.

[10] J. Kabzan, M. I. Valls, V. J. F. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, i R. Siegwart, “Amz driverless: The full autonomous racing system”, *Journal of Field Robotics*, 2020. https://doi.org/10.1002/rob.21977

[11] S. Caron, A. Zaki, P. Otta, D. Arnström, J. Carpentier, F. Yang, i P.-A. Leziart, “qpbenchmark: Benchmark for quadratic programming solvers available in Python”, 2024. [Mrežno]. Adresa: https://github.com/qpsolvers/qpbenchmark

[12] P. F. Lima, “Predictive control for autonomous driving with experimental evaluation on a heavy-duty construction truck”, doktorska disertacija, KTH Royal Institute of Technology, Stockholm, Sweden, 2016.

[13] I. H. Gundlach, “Zeitoptimale trajektorienplanung für automatisiertes fahren bis in den fahrdynamischen grenzbereich”, doktorska disertacija, Technische Universität Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, 2020.

[14] D. Groznica, “Konstrukcija kočnog sustava bolida formule”, Završni rad, Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje, Zagreb, Hrvatska, 2019.

[15] F. Kolarić, “Estimacija uzdužne brzine vozila korištenjem adaptivnog kalmanovog filtra”, Zagreb, Hrvatska, 2022.

[16] T. Barfoot i C. Clark, “Motion planning for formations of mobile robots”, *Robotics and Autonomous Systems*, sv. 46, str. 65–78, 2004.

Sažetak

Upravljanje gibanjem trkaćeg bolida koristeći modelsko prediktivno upravljanje

Matej Lovreković

Cilj je ovog diplomskog rada implementirati upravljanje gibanjem autonomnog mobilnog vozila u obliku trkaćeg bolida s pogonom na stražnjim kotačima. U radu je implementirano modelsko prediktivno upravljanje za slijeđenje zadane putanje i brzine vozila kroz stazu. Izvedena je analitička linearizacija modela vozila koja osigurava da je model uvijek lineariziran oko trenutne radne točke. Simulacijama je validiran linearizirani model. Modeliranjem i upravljanjem aktuatora ostvareno je izravno upravljanje pozicijom aktuatora za skretanje, momentom aktuatora za kočenje te momentom pogonskih motora. Implementirani su adaptivni MPC te LTV MPC. Algoritmi su validirani simulacijama unutar *Simulink* sučelja. Analizirano je upravljanje u različitim scenarijima poput šumovitog okruženja, kašnjenja signala te aktiviranja definiranih ograničenja upravljanja.

Ključne riječi: Autonomni trkaći bolid; Modelsko prediktivno upravljanje; Formula Student; Modeliranje; Simulacija; Adaptivni MPC; LTV MPC

Abstract

Motion control of a racing car using model predictive control

Matej Lovreković

The aim of this thesis is to implement motion control for an autonomous mobile vehicle in the form of a rear-wheel drive racing car. In this work, model predictive control is implemented in order to follow a given path and vehicle speed through the track. An analytical linearization of the vehicle model is derived, ensuring that the model is always linearized at the current operating point. The linearized model is validated through simulations. By modeling and controlling the actuators, direct control of the actuator position for steering, actuator torque for braking, and torque of the drive motors is achieved. Adaptive MPC and LTV MPC are implemented. The algorithms are validated through simulations within the *Simulink* interface. Vehicle control is analyzed in various scenarios, such as noisy environments, signal delays, and the activation of defined control constraints.

Keywords: Autonomous racing vehicle; Model predictive control; Formula Student; Modeling; Simulation; Adaptive MPC; LTV MPC