Koopman operator based model predictive control of vehicle dynamics

Švec, Marko

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Marko Švec

KOOPMAN OPERATOR BASED MODEL PREDICTIVE CONTROL OF VEHICLE DYNAMICS

DOCTORAL THESIS

Zagreb, 2024

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Marko Švec

KOOPMAN OPERATOR BASED MODEL PREDICTIVE CONTROL OF VEHICLE DYNAMICS

DOCTORAL THESIS

Supervisor: Professor Jadranko Matuško, PhD

Zagreb, 2024

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Marko Švec

MODELSKO PREDIKTIVNO UPRAVLJANJE DINAMIKOM VOZILA TEMELJENO NA KOOPMANOVOM OPERATORU

DOKTORSKI RAD

Mentor: Prof. dr. sc. Jadranko Matuško

Zagreb, 2024.

Doctoral thesis is made at the University of Zagreb Faculty of Electrical Engineering and Computing, Department of Electric Machines, Drives and Automation

Supervisor: Professor Jadranko Matuško, PhD

Thesis contains 149 pages

Thesis no.: _____

ABOUT THE SUPERVISOR

Jadranko Matuško was born in Metković in 1975. He received his B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia, in 1999, 2003 and 2008, respectively. He spent the academic year 2011/2012 at the University of California, Berkeley, USA, as a visiting researcher.

Since October 2008, he has been with the Department of Electric Machines, Drives and Automation at FER. In January 2024 he was promoted to tenured full professor in the same department. He coordinated 2 national and participated in 14 national and 5 international scientific projects. He is a member of the Scientific Center of Excellence for Data Science and Cooperative Systems. He has published over 60 papers in journals and conference proceedings in the domains of optimal control, intelligent control and estimation, and mechatronic systems. He is an author of one national and one international patent.

Prof. Matuško is a member of IEEE and KoREMA. He has been a Proceedings editor for 9 international conferences and a Program Committee chair for 8 international conferences. He is a member of the editorial board of one scientific journal.

O MENTORU

Jadranko Matuško rođen je u Metkoviću 1975. godine. Diplomirao je, magistrirao i doktorirao na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER) 1999., 2003. odnosno 2008. godine. Akademsku 2011./2012. proveo je na usavršavanju na Sveučilištu u Berkeleyu, California, SAD.

Od listopada 2008. zaposlen je Zavodu za elektrostrojarstvo i automatizaciju Fakulteta elektrotehnike i računarstva kao docent. U siječnju 2024. godine izabran je u zvanje redovitog profesora u trajnom zvanju na istom Zavodu. Bio je voditelj dva domaća znanstveno istraživačka projekta te sudjelovao na 14 domaćih i 5 međunarodnih projekata. Član je znanstvenog centra izvrsnosti za znanost o podatcima i kooperativne sustave. Objavio je preko 60 radova u časopisima i zbornicima međunarodnih znanstvenih skupova iz područja optimalnog upravljanja, inteligentnog upravljanja i estimacije te mehatroničkih sustava. Autor je jednog domaćeg i jednog međunarodnog patenta.

Prof. Matuško član je stručnih udruga IEEE i KoREMA. Bio je urednik 9 zbornika radova znanstvenih skupova te bio predsjednik organizacijskog ili programskog odbora 8 znanstvenih skupova. Član je uredničkog odbora jednog znanstvenog časopisa.

ZAHVALA

Želio bih zahvaliti svome mentoru, profesoru Jadranku Matušku, koji mi je davne 2017. godine predložio da upišem doktorat. Sasvim je evidentno da bez njega ne bi bilo ni ove disertacije. Hvala profesore što ste mi omogućili da radim na zanimljivim projektima, podržavali moju znatiželju i minimalno me obasipali poslom koji nije vezan za moj doktorat, omogućivši mi tako da se posvetim vlastitom istraživanju. Tek sad shvaćam koliko mi je to zapravo značilo.

Posebnu zahvalu zaslužuje i profesor Šandor Ileš koji se prema meni odnosio kao prema vlastitom studentu, iako službeno nije imao nikakve obaveze pomagati mi. Hvala na svom utrošenom vremenu na ispravljanje mojih grešaka, na svim profesionalnim i osobnim savjetima i svom znanju koje si podijelio sa mnom. Uistinu je bio užitak raditi s tobom.

Hvala i svim kolegama sa ZESA-e. Kao jedan od rijetkih automatičara na zavodu punom elektrostrojara, imao sam priliku steći razna znanja van svoje domene i neopisivo sam zahvalan na toj prilici. Osim toga, kave su uvijek bile zabavne i pune smijeha.

Jedno veliko hvala mojim cimerima, a posebice Bojanu i Josipu, s kojima sam najviše profesionalno surađivao. Hvala na svoj pomoći i svemu što ste me naučili.

Hvala profesoru Zlatku Drmaču na objašnjenju Koopmanovog operatora. Hvala i Vítu Cibulki na svim raspravama o Koopmanovom operatoru i strojnom učenju.

Mojim dragim prijateljima, hvala na zabavnim trenucima punima smijeha i šale. Hvala na svim feštama, društvenim igrama i glupim forama koje su mi pomogle da nakratko pobjegnem od obaveza i stresa.

Hvala mojoj obitelji, posebice mojim roditeljima, Blaženki i Damiru, na svemu što su učinili za mene, i baki Danici na svoj podršci i odličnoj hrani. Pružili ste mi topli dom, sigurnost i beskrajnu ljubav. Bez vaše podrške sigurno ne bi bilo ni mog uspjeha i zbog svega toga bit ću vam zauvijek zahvalan. Dodatno, hvala mami što je slušala moja objašnjenja raznih matematičkih koncepata, stvarno je puno pomoglo.

Na kraju, želio bih zahvaliti mojoj Mariji. Hvala ti na svoj podršci, razumijevanju i prvenstveno strpljenju. Bez tvoje ljubavi, osmijeha i pametnih savjeta, ovaj period bi mi bio puno teži. Hvala ti što si me ohrabrivala kroz svaku fazu ovog izazova, slušala me kad bih beskrajno pričao o temama koje su zanimljive samo meni, i bila moj najveći oslonac kad mi je sve djelovalo preteško. Tvoja ljubav i vjera u mene značili su mi više nego što riječima mogu opisati.

ABSTRACT

Over the past decade, the global market for electric and hybrid vehicles has grown significantly, leading to technological innovations that are transforming the automotive industry. This increasing popularity has accelerated the development of autonomous vehicles and vehicle dynamics control systems, which require robust and efficient control strategies to ensure optimal performance.

This thesis investigates the use of the Koopman operator with model predictive control (MPC) to leverage performance and computational challenges in real-time vehicle dynamics control. The Koopman operator provides a linear representation of nonlinear systems and enables simplified control of complex dynamical systems. By integrating the Koopman operator into traditional nonlinear MPC (NMPC), a Koopman MPC (KMPC) framework is created that transforms nonlinear optimization problems into quadratic optimization problems. This change improves computational efficiency with minimal impact on control accuracy.

The thesis covers a number of topics that are essential for vehicle dynamics control systems. It begins with an examination of the key concepts of vehicle dynamics models and the fundamentals of model predictive control (MPC). It then introduces the theory of the Koopman operator and discusses various data-driven methods for identifying Koopman models, including extended dynamic mode decomposition (EDMD) and deep dynamic mode decomposition (Deep-DMD). In addition, a novel method called enhanced extended dynamic mode decomposition (E^2DMD) is introduced, together with three different numerical approaches to its implementation. In the remainder of the thesis, the application of the Koopman operator to basic vehicle models is investigated, integrating different Koopman-based models into the MPC framework. Furthermore, a torque vectoring algorithm using KMPC is developed and a comparative analysis with established MPC strategies is performed. Furthermore, the thesis describes the experimental validation of these control strategies using a scaled vehicle model on a treadmill. It details the setup, data acquisition and compares the performance of NMPC and KMPC offering insights into the effectiveness of these control strategies.

The results highlight the potential of Koopman-based controllers to bridge the gap between the high control performance of NMPC and the computational efficiency required for practical use. The thesis also presents new Koopman operator identification methods and a novel approach to generate learning data through nonlinear model identification as an intermediate step. These innovations show promising future directions for research and development in the field of vehicle dynamics control.

KEYWORDS: vehicle dynamics, model predictive control, direct yaw moment control, torque vectoring, Koopman operator, model identification, extended dynamic mode decomposition

SAŽETAK

MODELSKO PREDIKTIVNO UPRAVLJANJE DINAMIKOM VOZILA TEMELJENO NA KOOPMANOVOM OPERATORU

Tijekom posljednjih deset godina, globalni tržišni udio električnih i hibridnih vozila neprekidno raste. Proizvođači automobila i političari aktivno rade na daljnjem jačanju ovog trenda, što potvrđuju razne studije, kao i opsežna medijska pokrivenost i svakodnevna zapažanja. Ovaj trend ne samo da označava prelazak na drugačije načine prijevoza, već također otvara nove mogućnosti za daljnji razvoj sustava upravljanja dinamikom vozila. Također potiče razvoj autonomnih vozila, koja bi mogla revolucionirati naš pristup mobilnosti, logistici i sigurnosti. U ovom kontekstu, razvoj robusnih i učinkovitih upravljačkih strategija ključan je za iskorištavanje punog potencijala ovih tehnoloških napredaka.

Jedna od najsofisticiranijih tehnika u području automatike je model prediktivno upravljanje (MPC), koje optimizira upravljačke akcije na temelju predikcije budućih stanja sustava. MPC je poznat po svojoj sposobnosti da se izravno nosi s višedimenzionalnim sustavima i ograničenjima, što ga čini idealnim za kompleksne sustave dinamike vozila. Kada se koriste detaljni nelinearni modeli za prikaz dinamike vozila, rezultat je skup algoritama koje nazivamo nelinearni MPC (NMPC). NMPC se bavi nelinearnim optimizacijskim problemima koji mogu biti računski zahtjevni, što predstavlja izazov za aplikacije u realnom vremenu.

Koopmanov operator nudi obećavajuće rješenje ovog problema. Izvorno predstavljen u ergodičkoj teoriji, Koopmanov operator omogućuje linearnu reprezentaciju nelinearnih dinamičkih sustava transformiranjem prostora stanja u (teorijski beskonačno dimenzionalni) prostor osmotrivih funkcija. Ova transformacija omogućuje primjenu linearnih upravljačkih tehnika, čak i za nelinearne sustave, čime se znatno pojednostavljuje računska složenost. Međutim, u praksi se koristi konačno-dimenzionalna aproksimacija Koopmanovog operatora, što dovodi do novih potencijalnih problema.

Integracijom Koopmanovog operatora u MPC, tradicionalni NMPC algoritmi mogu se transformirati u tzv. Koopman MPC (KMPC) algoritme, koji mogu formirati linearne ili kvadratne optimizacijske probleme, što je znatno brže i lakše riješiti. Prema postojećoj literaturi, ovaj pristup zadržava većinu preciznosti povezane s NMPC-om zahvaljujući visokoj dimenzionalnosti prostora stanja, ali također značajno povećava brzinu rješavanja problema. Iz tog razloga korištenje MPC-a temeljenog na Koopmanovom operatoru, može postići "najbolje iz oba svijeta": održavanje visokih performansi u predikciji i upravljanju dinamikom vozila, uz poboljšanje računske učinkovitosti, što je ključno za primjene u stvarnom vremenu.

Glavna motivacija ovog rada je razviti upravljački okvir koji ne samo da zadovoljava visoke performanse modernih vozila, već i prevladava računske izazove u stvarnom vremenu. Istraživanje

ima za cilj pokazati izvedivost i prednosti ovog inovativnog pristupa. Razvojem algoritama, simulacija i praktičnom primjenom, cilj je otvoriti put responzivnijim, učinkovitijim i sigurnijim vozilima. Nadalje, integracijom tehnika dubokog učenja za aproksimaciju modela Koopmanovog operatora, ova disertacija također ima za cilj potaknuti daljnju suradnju između istraživača u području automatike i umjetne inteligencije - sinergiju koja će vjerojatno postati sve važnija u budućnosti.

Izvorni doprinosi ove disertacije odnose se na numeričke metode za identifikaciju Koopmanovog operatora i primjenu identificiranih modela u prediktivnim algoritmima za upravljanje dinamikom vozila. U suštini, predstavljaju fuziju metoda strojnog učenja i klasičnog modeliranja dinamičkih sustava, analizirajući prednosti i nedostatke svakog pristupa. Doprinos se sastoji od tri dijela, koja su, zajedno s kratkim objašnjenjima, navedena u nastavku.

1. Metoda za identifikaciju modela dinamike vozila temeljena na Koopmanovom operatoru, pogodna za primjenu u prediktivnim upravljačkim algoritmima.

   Ovaj dio doprinosa temelji se na prijedlogu nekoliko numeričkih metoda za identifikaciju modela temeljenih na Koopmanovom operatoru i istraživanju njihove učinkovitosti u modeliranju dinamičkih sustava, s naglaskom na dinamiku vozila. Osnovna ideja je zamijeniti postojeće nelinearne modele modelima zasnovanim na Koopmanovom operatoru kako bi se smanjila računska složenost prilikom izračunavanja predikcija, bez značajnog pogoršanja točnosti istih. Kao početna metoda korištena je proširena dinamička modalna dekompozicija (EDMD), na temelju koje su razvijene tri nove numeričke metode: redukcija vektora baznih funkcija diskretnim odabirom, algoritam učenja predikcije u više koraka i redukcija vektora baznih funkcija kao problem optimizacije hiperparametara. Ove metode zajednički se nazivaju poboljšana proširena dinamička modalna dekompozicija (E^2DMD). Osim navedenih, prilagođena je i korištena postojeća metoda poznata kao duboka dinamička modalna dekompozicija (Deep-DMD). Svi modeli temeljeni na Koopmanovom operatoru su evaluirani i uspoređeni međusobno, kao i s nelinearnim i s modelima temeljenim na klasičnoj linearnizaciji oko radne točke. Pokazali su obećavajuće rezultate, što opravdava njihovu primjenu u prediktivnim upravljačkim algoritmima.

2. Algoritam modelskog prediktivnog upravljanja distribucijom zakretnog momenta kotača s ciljem poboljšanja upravljivosti vozila, temeljen na modelu vozila identificiranom s Koopmanovim operatorom.

   Ovaj dio doprinosa zasniva se na korištenju modela temeljenih na Koopmanovom operatoru (razvijenih kao dio prvog dijela doprinosa) u algoritmima modelskog prediktivnog upravljanja za dinamiku vozila. Testirano je nekoliko pristupa, počevši od onih temeljenih na jednostavnim bicikl i dvotračnim modelima vozila, koji su poslužili kao dokaz koncepta. Pristup je dodatno generaliziran i primijenjen na prediktivno vektoriranje zakretnog momenta koristeći složenije modele, što je testirano u CarMaker-u, programskom alatu za simulaciju dinamike vozila visoke vjernosti. Regulatori temeljeni na Koopmanovom operatoru pokazali su dobre performanse i nisku računsku složenost, potvrđujući time hipotezu. U konačnici, razvijen je sličan upravljački algoritam i primijenjen na skaliranom vozilu koje se kreće na pokretnoj traci. Ovaj put, regulator temeljen na Koopmanovom

operatoru pokazao je ne samo dobru računsku izvedbu, već i veću otpornost na vremenska kašnjenja u usporedbi s nelinearnim regulatorom.

3. Tehnika za generiranje skupa podataka za učenje Koopmanovog operatora najprije stvaranjem nelinearnog modela iz eksperimentalnih podataka i zatim simulacijom različitih scenarija koristeći taj model.

   Glavna komponenta svake metode modeliranja temeljene na podacima su sami podaci. Skup podataka mora biti dovoljno velik i informativan, inače je gotovo nemoguće postići dobru točnost modela. Međutim, prilikom prikupljanja podataka iz simulacije ili eksperimenta, ponekad nije moguće isprobati sve zamislive kombinacije ulaznih signala ili obuhvatiti cijelu relevantanu regiju unutar prostora stanja. To može biti posljedica ograničenja sustava, lošeg dizajna eksperimenta ili može jednostavno biti preskupo. Ovaj doprinos predlaže korištenje identifikacije parametara standardnog nelinearnog modela kao međukoraka za generiranje skupa podataka. Prvo se prikuplja manji broj uzoraka iz eksperimenta. Zatim se odabire odgovarajući nelinearni model, a njegovi se parametri određuju na temelju prikupljenih podataka. To je iterativni proces u kojem se model može modificirati dok se ne postigne zadovoljavajuća točnost. Nakon toga, ovaj nelinearni model koristi se za simulaciju velikog broja trajektorija koje bi formirale novi skup podataka. Konačni model se zatim određuje na temelju ovog novog skupa podataka. Ovaj pristup koristi se za identifikaciju Koopmanovog operatora i pokazuje dobru izvedbu. Osim toga, provedena je analiza veličine skupa podataka gdje su eksperimentalni i simulacijski podaci izravno uspoređeni.

Disertacija se sastoji od sedam poglavlja, od kojih svako započinje sažetkom koji opisuje njegov sadržaj. Nakon toga slijedi sustavna prezentacija problema i pregled postojeće literature u području, ako je relevantno za poglavlje. Nakon glavne rasprave, dan je prikaz najvažnijih rezultata i doprinosa. Prvo poglavlje opisuje motivaciju za istraživanje provedeno u disertaciji, navodi dijelove doprinosa i objašnjava strukturu rada. Pregled ostalih poglavlja dan je u nastavku teksta.

POGLAVLJE 2. U ovom poglavlju izložene su matematičke osnove svih metoda i koncepata o kojima se raspravlja kasnije u disertaciji. Prvo su predstavljeni modeli dinamike vozila, kao što su dvotračni i bicikl model, kao i različite formulacije klizanja kotača. Zatim se ispituje složenost modeliranja sila na gumama i njihova ovisnost o različitim faktorima. U odjeljku o modelskom prediktivnom upravljanju razmatraju se njegovi temelji, primjene i problemi, s naglaskom na njegove linearne i nelinearne oblike. Poglavlje također obrađuje teorijski okvir Koopmanovog operatora i njegovu upotrebu u pretvaranju nelinearne dinamike u linearnu, ističući ograničenja zbog njegove (teorijske) beskonačne dimenzionalnosti. Na kraju, rasprava se usmjerava na sustave upravljanja dinamikom vozila, posebno aktivne sigurnosne sustave kao što su ABS, ESC i TV, naglašavajući njihovu važnost u poboljšanju stabilnosti i sigurnosti vozila.

POGLAVLJE 3. Ovo poglavlje bavi se različitim metodama identifikacije Koopmanovih modela temeljenih na podacima. U početku se raspravlja o dobro poznatim metodama, proširenoj dinamičkoj modalnoj dekompoziciji (EDMD) i dubokoj dinamičkoj modalnoj dekompoziciji (Deep-DMD), koje su detaljno objašnjene u prethodnim studijama. Zatim se uvodi nov pristup

nazvan poboljšana proširena dinamička modalna dekompozicija (E²DMD), koji predstavlja jedan od ključnih doprinosa ove disertacije. Objašnjava se logika ove metode, nakon čega slijedi prikaz tri numerička pristupa za razvoj takvih modela: redukcija vektora baznih funkcija diskretnim odabirom (E²DMD-DS) , algoritam učenja predikcije u više koraka (E²DMD-MS) i redukcija vektora baznih funkcija kao problem optimizacije hiperparametara (E²DMD-HO). Poglavlje također kratko spominje druge pristupe temeljene na podacima. Na kraju, ocjenjuju se različiti linearni prediktori primjenom na trima ustaljenim referentnim dinamičkim sustavima: Van der Polovom oscilatoru, prigušenom Duffingovom oscilatoru i bilinearnom motoru. Rezultati ovih simulacija zatim se uspoređuju i analiziraju.

Ključni nalazi uključuju dosljedno poboljšanje performansi prediktora kako se dimenzija prostora stanja povećava. Također su pronađene iznenađujuće varijacije u performansama Deep-DMD-a, što bi moglo ukazivati na probleme u arhitekturi ili dizajnu neuronske mreže. Rješavanje problema Deep-DMD-a također bi moglo poboljšati performanse E²DMD-MS metode, budući da se obje treniraju sličnim algoritmima. Očekuje se da će E²DMD-HO i E²DMD-MS imati slične performanse zbog korištenja afine transformacije, dok bi E²DMD-DS trebala biti manje učinkovita zbog binarnih vrijednosti i odsutnosti vektora pomaka. Međutim, rezultati ovog istraživanja, iz kojih se mogu izvući različiti zaključci, sugeriraju da metoda optimizacije korištena za učenje može značajno i neočekivano utjecati na performanse prediktora. Na kraju, bitno je napomenuti da odabir numeričke metode za učenje aproksimacije Koopmanovog operatora ovisi o dinamičkom sustavu koji je predmet istraživanja.

POGLAVLJE 4. Poglavlje istražuje kako se Koopmanov operator može koristiti za modeliranje i upravljanje dinamikom vozila koristeći osnovne modele vozila i uzdužno klizanje guma kao ulazni signal. Početni dio ispituje upotrebu EDMD algoritma za identifikaciju bicikl modela koji ne uključuje model gume. Ovaj model se zatim uspoređuje s modelima generiranim klasičnom linearizacijom oko radne točke. Daljnja validacija provodi se integracijom ovog modela u MPC algoritam i testiranjem na nelinearnom bicikl modelu vozila. Proces identifikacije modela i dizajn prediktivnog regulatora provjeravaju se korištenjem MATLAB-a i Simulinka. Sljedeći odjeljak ispituje primjenu Koopmanovog operatora za identifikaciju dvotračnog modela vozila, koji se zatim kombinira sa strategijom vektoriranja zakretnog momenta pomoću MPC-a. Ovaj dio poglavlja posebno se fokusira na poboljšanje stabilnosti ručno upravljanih vozila i predstavlja inovativan pristup primjeni izravnog upravljanja zakretnim momentom s linearnim MPC-om temeljenim na Koopmanovom operatoru. EDMD metoda ponovno se koristi za aproksimaciju modela, uzimajući u obzir potrebu za linearnim odnosom između propagacije stanja sustava i ulaza, kako je raspravljeno u prethodnom poglavlju. Ovaj linearni odnos, koji obično nije prisutan u modelima vozila, uspostavlja se konstrukcijom specifičnih nelinearnih transformacija. Nakon točnog razvoja modela, isti se koristi za razvoj KMPC algoritma, koji se zatim uspoređuje s linearnim vremenski promjenjivim MPC-om za različite predikcijske horizonte. KMPC postiže bolje rezultate u svim testnim scenarijima, kako u pogledu upravljačkih performansi, tako i u pogledu vremena izvođenja.

POGLAVLJE 5. Ovo poglavlje bavi se razvojem i primjenom algoritama prediktivnog vektoriranja zakretnog momenta koristeći Koopmanov operator. Započinje sveobuhvatnim pregledom literature o sustavima upravljanja dinamikom vozila, uspoređujući tradicionalne upravljačke

tehnike kao što su ABS, ESC i tempomat, koje se oslanjaju na heurističke algoritme ili jednostavne matematičke modele, sa sofisticiranijim nelinearnim modelima i upravljačkim strategijama. Ova rasprava ističe nedostatke postojećih pristupa i predlaže Koopmanov operator kao održiv alat za poboljšanje upravljanja vozilima u scenarijima u kojima nelinearni efekti dolaze do izražaja. Sljedeći dio poglavlja posvećen je upotrebi Koopmanovog operatora za modeliranje dinamike vozila. Obuhvaća identifikaciju parametara nelinearnog modela, nakon čega slijedi prikupljanje podataka i proces identifikacije Koopmanovih modela. Učinkovitost ovih modela ocjenjuje se korištenjem skupova podataka za treniranje i validaciju. U trećem odjeljku predstavljaju se linearni vremenski promjenjivi MPC (LTV-MPC), Koopman MPC i nelinearni MPC za vektoriranje zakretnog momenta te se raspravlja o njihovim razlikama i potencijalnim problemima. Nakon matematičke formulacije, ovi regulatori ocjenjuju se u simulatoru visoke vjernosti u četiri različita scenarija. Ishodi se temeljito analiziraju, pružajući kritičku usporedbu učinkovitosti i izvodljivosti korištenja sustava prediktivnog vektoriranja zakretnog momenta temeljenih na Koopmanovom operatoru u odnosu na druge metode.

Rezultati objašnjavaju kompromise između računske učinkovitosti i upravljačkih performansi. NMPC dosljedno pokazuje superiorne sposobnosti manevriranja, posebno u eksperimentima na visokim brzinama na stazi, što je potvrđeno dosljedno nižim normaliziranim vrijednostima funkcije cilja zatvorenog kruga. Međutim, to dolazi s višim računskim zahtjevima, što se odražava u dužim vremenima izvođenja, posebno kada se produlji predikcijski horizont. To naglašava kritični kompromis između upravljačkih performansi i računske izvedivosti, osobito u primjenama u stvarnom vremenu gdje je brzina izvođenja vrlo važna. S druge strane, regulatori temeljeni na Koopmanovom operatoru, posebno E^{2}DMD-MPC i Deep-DMD-MPC, nude uravnotežen kompromis između računske učinkovitosti i upravljačkih performansi. Njihova sposobnost postizanja konkurentne učinkovitosti uz značajno kraća vremena izvođenja čini ih kvalitetnom alternativom, posebno za scenarije u kojima su računalni resursi ograničeni ili je izvođenje u stvarnom vremenu ključno. Deep-DMD-MPC posebno pokazuje značajno poboljšanje u upravljačkoj učinkovitosti kako se predikcijski horizont povećava. Međutim, također pokazuje slabe performanse u nekim situacijama s malim predikcijskim horizontom, što ukazuje na nepredvidivost i kompleksnost identifikacije, vjerojatno zbog neuronske mreže koja je temelj modela. LTV-MPC, unatoč svojoj teorijskoj jednostavnosti i nižim računskim zahtjevima u odnosu na NMPC, pokazuje lošije performanse u pogledu upravljačkih performansi u većini scenarija. Njegove performanse su znatno bolje u eksperimentu na niskoj brzini na Nürburgringu, što sugerira je bolje prilagođen slučajevima gdje su nelinearnosti manje izražene ili predvidljive.

POGLAVLJE 6. Poglavlje istražuje upravljanje dinamikom vozila kroz eksperimente koristeći skalirani model vozila na pokretnoj traci kako bi se evaluirali upravljački algoritmi. Započinje prikazom eksperimentalnog postava i pregledom literature. Detaljno se opisuje identifikacija parametara nelinearnog modela vozila kroz izravne eksperimente, mjerenja i testove, uz validaciju usporedbom snimljenih signala s predikcijama modela. Nakon toga, fokus se prebacuje na identifikaciju Koopmanovog modela koristeći EDMD algoritam, uključujući generiranje podataka, strategije uzorkovanja i proces učenja modela. Učinkovitost modela demonstrira se korištenjem različitih skupova podataka i testnih scenarija, s posebnim naglaskom na to kako veličina i raspodjela skupova podataka utječu na točnost modela. Osim toga, ovo poglavlje ocjenjuje NMPC i KMPC strategije i predstavlja detaljne rezultate eksperimentalnih testova s oba kontrolera, uključujući

postavke, vrijeme izvršavanja i performanse za specifične manevre. Analiziraju se efekti različitih referenci i duljine predikcijog horizonta na performans regulatora te se eksperimentalni rezultati uspoređuju sa simulacijom.

Sveukupni zaključak ove analize je da KMPC može nadmašiti NMPC u pogledu stabilnosti i učinkovitosti, što je posebno primjetno kada su prisutna velika vremenska kašnjenja u sustavu. Iako NMPC ima veći potencijal pod simuliranim idealnim uvjetima, manje je učinkovit u stvarnim uvjetima zbog većih računskih zahtjeva i komunikacijskog kašnjenja. S druge strane, KMPC pokazuje niže iznose funkcije cilja zatvorenog kruga i značajno brža vremena izvođenja, što ga čini prikladnijim za primjene u stvarnom vremenu. Čak i kada se predikcijski horizont produži, što poboljšava performanse praćenja KMPC-a, povećanje vremena izvođenja ostaje unutar prihvatljivih granica za rad u stvarnom vremenu.

POGLAVLJE 7. Zadnje poglavlje sažima najvažnije rezultate i znanstvene doprinose te daje prijedloge za buduće pravce istraživanja.

Sveukupno, ova disertacija naglašava kritičnu važnost odabira prave upravljačke strategije koja ne samo da uzima u obzir nelinearnost dinamike vozila, već se i usklađuje s računalnim ograničenjima primjene u stvarnom vremenu. Stečeni uvidi naglašavaju potencijal regulatora temeljenih na Koopmanovom operatoru kao pristupa koji nudi strateški kompromis između visokih performansi NMPC algoritama i niske računske složenosti potrebne za praktičnu implementaciju. Nadalje, predstavljene su neke nove metode identifikacije Koopmanovog operatora, kao i novi pristup za generiranje podataka za učenje korištenjem identifikacije nelinearnih modela kao međukoraka.

KLJUČNE RIJEČI: dinamika vozila, modelsko prediktivno upravljanje, direktno upravljanje zakretnim momentom, vektoriranje zakretnog momenta, Koopmanov operator, identifikacija modela, proširena dinamička modalna dekompozicija

CONTENTS

1

Introduction

The introduction chapter outlines the motivation for the research conducted in the thesis, beginning with arguments supporting the growing need for advanced vehicle dynamics control algorithms. It introduces (nonlinear) model predictive control as a viable method to address these challenges. Additionally, it proposes the Koopman operator as a viable alternative to conventional nonlinear models and provides justification for this perspective. The chapter then describes the original contributions of the thesis and concludes with an overview of the thesis structure and a summary of the content of each chapter.

1.1 Motivation and problem statement

Over the last ten years, the global market share of electric and hybrid vehicles has been steadily rising. Manufacturers and policy makers are actively working to further strengthen this trend, as evidenced by various studies, e.g. [1, 2], as well as widespread media coverage and everyday observations. This trend not only signals a shift towards other modes of transport, but also opens up new avenues for the further development of vehicle dynamics control systems. It also promotes the development of autonomous vehicles [3], which are expected to revolutionize our approach to mobility, logistics and safety. In this context, the development of robust and efficient control strategies is crucial to exploit the full potential of these technological advances.

One of the most sophisticated techniques in the field of control engineering is model predictive control (MPC), which optimizes current control actions based on a prediction of future system states. MPC is known for its ability to deal explicitly with multivariable systems and constraints, making it ideal for complex vehicle dynamics systems. When detailed nonlinear models are used to capture vehicle dynamics, the approach evolves into nonlinear MPC (NMPC). NMPC deals with nonlinear optimization problems that can be computationally intensive, which is a challenge for real-time applications.

The Koopman operator offers a promising solution to this challenge. Originally introduced in ergodic theory, the Koopman operator provides a linear representation of nonlinear dynamical systems by transforming the state-space into a (theoretically infinite dimensional) space of observable functions. This transformation enables the application of linear control techniques, even for nonlinear systems, and thus considerably simplifies the computational complexity. The problem, of course, is that in practice a finite dimensional approximation of the Koopman operator is used, which introduces another set of potential issues.

By incorporating the Koopman operator into MPC, the traditional NMPC algorithms can be

transformed into so-called Koopman MPC (KMPC) algorithms, which can then form a linear or quadratic optimization problem that is much faster and easier to solve. According to the existing literature, this approach retains much of the precision associated with NMPC due to the high-dimensional lifting of the state-space, but gains significantly in computational speed. Thus, by using the Koopman operator-based MPC, one should achieve the "best of both worlds": maintaining high accuracy in predicting and controlling vehicle dynamics while improving computational efficiency, which is crucial for real-time applications.

The main motivation behind this thesis is to develop a control framework that not only meets the high performance requirements of modern vehicles, but also overcomes the computational challenges of real-time vehicle control. The research aims to demonstrate the feasibility and benefits of this innovative approach, which may bring the field of control engineering closer to a new standard for vehicle dynamics control. Through the development of algorithms, simulations and practical application, the goal is to pave the way for more responsive, efficient and safer vehicles. Furthermore, by integrating deep learning techniques to approximate Koopman operator models, this work also aims to encourage further collaboration between control engineering and artificial intelligence researchers - a synergy that is likely to become increasingly important in the future.

1.2 ORIGINAL CONTRIBUTIONS

The original contributions of the thesis revolve around numerical methods for Koopman operator identifcation and the application of the identified models in predictive algorithms for vehicle dynamics control. Essentially, they represent a fusion of machine learning methods and classical dynamic system modelling, analyzing the advantages and disadvantages of each approach. The contributions with a brief explanation follow below.

- A method for identifying a vehicle dynamics model based on the Koopman operator, suitable for applications in predictive control algorithms.

  The contribution is based on the proposal of several numerical methods for the identification of Koopman operator models and the investigation of their performance in the modelling of dynamical systems, with a focus on vehicle dynamics. The core idea is to replace existing nonlinear models with Koopman operator models in order to reduce the computational complexity while calculating predictions without significantly degrading the prediction accuracy. Extended dynamic mode decomposition (EDMD) is used as the starting method, on the basis of which three novel numerical methods are developed: basis function reduction by discrete selection, multiple step prediction learning algorithm and basis function reduction as a hyperparameter optimization problem. These methods are collectively referred to as enhanced extended dynamic mode decomposition (E^2DMD). In addition, an existing method called deep dynamic mode decomposition (Deep-DMD) is adapted and used. All Koopman-based models are evaluated and compared with each other and with nonlinear and Taylor linearization-based models. They showed promising results, which justifies their application in model predictive control algorithms.

- Model predictive control algorithm of wheel torque distribution with the aim of improving vehicle handling, based on a vehicle model identified with the Koopman operator.

  The contribution consists of using some of the better performing Koopman operator-based models (developed as a part of the first contribution) in model predictive control algorithms for vehicle dynamics. Several approaches are tested, starting with those based on simple bicycle and two-track vehicle models with multiple assumptions, that served as a proof-of-concept. The approach is further generalized and applied to predictive torque vectoring using higher complexity models, which is tested in CarMaker, a high-fidelity simulation software for vehicle dynamics. The Koopman operator-based controllers showed good performance and a really low computational cost, confirming the hypothesis. Finally, a similar control algorithm is developed and applied to a scaled vehicle running on a treadmill. This time, the Koopman-based controller showed not only good computational performance but also higher robustness against time delay compared to a nonlinear controller.

- A technique for generating a Koopman operator learning dataset by first creating a nonlinear model from experimental data and then simulating different scenarios with this model.

  The main component of any data-driven modelling method is the data itself. The dataset must be large and informative enough, otherwise it is almost impossible to achieve good model performance. However, when collecting the data from the simulation or experiment, it is sometimes not possible to try all conceivable input signal combinations or to capture the entire state-space region of interest. This may be due to system limitations or poor experiment design, or it may simply be too costly. This contribution proposes using a standard nonlinear model parameter identification as an intermediate step for dataset generation. First, a smaller number of samples are collected from the experiment. Then, a suitable nonlinear model is selected and its parameters are determined based on the collected data. This is an iterative process in which the model can be modified until satisfactory accuracy is achieved. Subsequently, this nonlinear model is used to simulate a large number of trajectories that would form the new dataset. The final model is then determined based on this new dataset. This approach is used for the identification of the Koopman operator and shows good performance. In addition, an analysis of the dataset size is performed where experimental and simulation data are directly compared.

1.3 OUTLINE OF THE THESIS

The thesis consists of seven chapters, each beginning with a summary outlining its content. This is followed by a systematic presentation of the problem and an overview of the existing literature in the field, where relevant to the chapter. After the main discussion, a summary of the most important results and contributions is given. The following section provides an overview of the thesis, accompanied by a summary of the content of the individual chapters.

CHAPTER 2. The chapter outlines the mathematical basis for all methods and concepts discussed later in the thesis. First, vehicle dynamics models, such as the two-track and the bicycle model, as well as different slip formulations are presented. Next, the complexity of modelling tire forces and their dependence on various factors are examined. The section on model predictive

control (MPC) looks at its fundamentals, applications and challenges, focusing on both its linear and nonlinear forms. The chapter also addresses the theoretical framework of the Koopman operator and its use in converting nonlinear dynamics into linear representations, noting the limitations due to its (theoretical) infinite dimensionality. Finally, the discussion turns to vehicle dynamics control systems, in particular active safety systems such as ABS, ESC and TV, and their importance in improving vehicle stability and safety is emphasized.

CHAPTER 3. This chapter deals with various methods for the data-driven identification of Koopman models. Initially, it discusses the well-established methods of extended dynamic mode decomposition (EDMD) and deep dynamic mode decomposition (Deep-DMD), which have been thoroughly detailed in previous studies. It then introduces a novel approach called enhanced extended dynamic mode decomposition (E^2DMD), marking one of the key contributions of this thesis. The logic behind this method is explained, followed by a presentation of three unique numerical techniques for developing such models: discrete basis function selection, a multiple step prediction learning algorithm, and basis function reduction viewed through the lens of hyperparameter optimization. The chapter also briefly mentions other data-driven approaches for identifying Koopman models. Finally, various linear predictors are evaluated by applying them to three recognized benchmark dynamical systems: the Van der Pol oscillator, the damped Duffing oscillator and the bilinear motor. The results of these simulations are then compared, analyzed and summarized in the concluding remarks.

CHAPTER 4. The chapter explores how the Koopman operator can be used to model and control vehicle dynamics using basic vehicle models and longitudinal tire slip as input. The initial section examines the use of the EDMD algorithm to identify a bicycle model that does not include a tire model. This model is then compared to the models generated by a Taylor expansion-based linearization. Further validation is performed by integrating this model into an MPC framework and testing it on a nonlinear bicycle vehicle model. Both the model identification process and the design of the predictive controller are verified using MATLAB and Simulink. The subsequent section examines the application of the Koopman operator to identify a two-track vehicle model, which is then combined with a torque vectoring control strategy through MPC. In particular, this part of the chapter focuses on enhancing the stability of manually steered vehicles and presents an innovative approach to applying direct yaw moment control with linear MPC based on the Koopman operator model. The EDMD method is again used to approximate the model, taking into account the need for a linear relationship between the propagation of the system states and the inputs, as discussed in a previous chapter. This linear relationship, which is not usually present in vehicles, is established by constructing specific nonlinear transformations. After accurately developing the model, it is used to develop a linear KMPC algorithm, which is then compared to a linear time-variant MPC over different prediction horizons.

CHAPTER 5. This chapter deals with the development and application of a model predictive torque vectoring algorithm using the Koopman operator. It begins with a comprehensive review of the literature on vehicle dynamics control systems, comparing traditional control techniques such as ABS, ESC and cruise control — which rely on heuristic algorithms or simple mathematical models — with more sophisticated nonlinear models and control strategies. This

discussion highlights the flaws of existing approaches and proposes the Koopman operator as a viable means of improving vehicle control in scenarios in which nonlinear effects occur. The next part of the chapter is devoted to the use of the Koopman operator for modelling vehicle dynamics. It covers the identification of nonlinear model parameters, followed by data acquisition and the process of identifying Koopman models. The effectiveness of these models is assessed using both training and validation datasets, with selected predictors included in the design of the control system. In the third section, the linear time-variant MPC, the Koopman MPC and the nonlinear MPC for torque vectoring are presented and their nuances and potential challenges are discussed. Following the theoretical presentation, these controllers are evaluated in a high-fidelity simulation setup in four different scenarios. The outcomes are thoroughly reviewed and discussed, providing a critical comparison of the effectiveness and feasibility of using Koopman-based predictive torque vectoring systems against other methods.

CHAPTER 6. The chapter investigates vehicle dynamics control through experiments using a scaled vehicle model on a treadmill to evaluate control algorithms. It begins with an overview of the experimental setup and a literature review. The identification of nonlinear vehicle model parameters through direct experiments, measurements and tests is described in detail, with validation by comparing these results with model predictions. Afterwards, the focus shifts to the identification of the Koopman model using the EDMD algorithm, including data generation, sampling strategies and the model learning process. The effectiveness of the model is demonstrated using different datasets and test scenarios, with a particular focus on how the size and distribution of the datasets affect model accuracy. In addition, this chapter evaluates the NMPC and KMPC strategies and presents detailed results from experimental tests with both controllers, including setup, execution times, and performance for specific manoeuvres. The effect of different references and prediction horizons on the control performance is analyzed and the experimental results are compared with the simulation.

CHAPTER 7. The final chapter summarises the most important results and scientific contributions of the thesis and makes suggestions for future research directions.

2

General background

This chapter provides an overview of the mathematical foundations of all the methods and concepts discussed later in the thesis. In the first sections, models of vehicle dynamics, including bicycle and two-track models, are introduced and alternative slip formulation is presented. The challenges of tire force modelling and its relationship to various influencing variables are mentioned. Model predictive control (MPC) and its logic, applications and challenges are described, with emphasis on its linear and nonlinear variants. The theoretical structure of the Koopman operator and its applications in converting nonlinear to linear dynamics are discussed, albeit with the practical limitations imposed by its infinite dimension. Finally, the chapter deals with vehicle dynamics control systems, focusing on active systems such as ABS, ESC and TV and emphasizing their role in improving vehicle stability and safety.

2.1 Vehicle dynamics models

The control strategies used in this thesis (namely the model predictive control described in Section 2.3) depend on accurate and reliable dynamical models to predict and optimize the future states of a system. This section therefore introduces two common vehicle dynamics models: the bicycle model and the two-track model. The bicycle model, known for its simplicity and computational efficiency, provides a basic approximation of vehicle behaviour. The two-track model, on the other hand, provides a more detailed representation by taking into account the width of a vehicle and the dynamics of the individual wheels.

2.1.1 *Bicycle model*

In this section, the single-track, or bicycle, model of a vehicle shown in Figure2.1is presented. The bicycle model reduces a vehicle to two wheels (front and rear) aligned along a single axis. This abstraction simplifies the equations of motion and allows a clearer focus on longitudinal and lateral dynamics without introducing additional complexity.

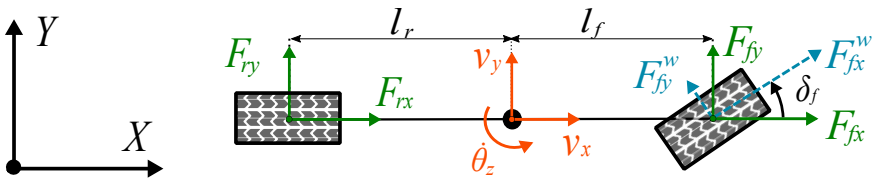

Figure 2.1: Bicycle model of a vehicle.

The continuous-time bicycle model is based on the model described in [4] and can be formulated as follows:

$$m\dot{v}_x = m\dot{\theta}_z v_y + F_{fx} + F_{rx} - \frac{1}{2} c_w \rho A_w v_x \sqrt{v_x^2 + v_y^2}, \tag{2.1}$$

$$m\dot{v}_y = -m\dot{\theta}_z v_x + F_{fy} + F_{ry} - \frac{1}{2} c_w \rho A_w v_y \sqrt{v_x^2 + v_y^2}, \tag{2.2}$$

$$J_z \ddot{\theta}_z = l_f F_{fy} - l_r F_{ry}, \tag{2.3}$$

$$J_w \dot{\omega}_\bullet = T_\bullet - f_{w\bullet} F_{\bullet z} - R_{w\bullet} F_{\bullet x}^w - b_{w\bullet} \omega_\bullet, \tag{2.4}$$

$$\dot{X} = v_x \cos\theta_z - v_y \sin\theta_z, \tag{2.5}$$

$$\dot{Y} = v_x \sin\theta_z + v_y \cos\theta_z. \tag{2.6}$$

The symbols $\bullet \in \{f, r\}$ denote the front and rear wheels. Within the given framework, the variables v_x, v_y, and θ_z represent the longitudinal velocity, the lateral velocity, and the yaw angle, respectively. The mass of the vehicle is denoted by m, the moment of inertia by J_z, and the wheel moment of inertia by J_w. The distances of the axles from the center of gravity (CoG) are represented by l_f and l_r. The drag force is described by the air density ρ, the drag coefficient c_w, and the area A_w exposed to the airflow. Only the airflow that is opposite to the trajectory of the vehicle's center of gravity is taken into account here. The forces acting on the vehicle body in the longitudinal and lateral directions are symbolized by $F_{\bullet x}$ and $F_{\bullet y}$, respectively. The wheel dynamics is described by the equation (2.4), which establishes the relationship between the wheel rotational velocities $\omega_\bullet$, the input wheel torques $T_\bullet$, the wheel radius $R_{w\bullet}$, the longitudinal wheel forces $F_{\bullet x}^w$, the wheel rolling resistance $f_{w\bullet}$, the wheel viscous friction $b_{w\bullet}$ and the vertical wheel forces $F_{\bullet z}$. In addition, the longitudinal and lateral position of the vehicle in the global coordinate system are denoted by the variables X and Y, respectively.

The forces acting on the axles in the vehicle coordinate system are described by the following equations:

$$\begin{aligned} F_{fx} &= F_{fx}^w \cos\delta_f - F_{fy}^w \sin\delta_f, \quad & F_{rx} &= F_{rx}^w, \\ F_{fy} &= F_{fx}^w \sin\delta_f + F_{fy}^w \cos\delta_f, \quad & F_{ry} &= F_{ry}^w. \end{aligned} \tag{2.7}$$

Here, δ_f represents the steering angle. Tire forces for a given axle are

$$\begin{aligned} F_{\bullet x}^w &= f_x^w(\alpha_\bullet, s_{\bullet x}, \mu, F_{\bullet z}), \\ F_{\bullet y}^w &= f_y^w(\alpha_\bullet, s_{\bullet x}, \mu, F_{\bullet z}), \end{aligned} \tag{2.8}$$

where f_x^w and f_y^w are the corresponding tire models, μ is the coefficient of friction (which is assumed to be the same for all wheels) and $F_{\bullet z}$ is the vertical wheel force. The longitudinal slip and lateral slip angle are:

$$s_{\bullet x} = \begin{cases} \frac{R_{w\bullet}\omega_\bullet - v_{\bullet x}^w}{R_{w\bullet}\omega_\bullet}, & v_{\bullet x}^w \le R_{w\bullet}\omega_\bullet, \quad R_{w\bullet}\omega_\bullet \neq 0, \\ \frac{R_{w\bullet}\omega_\bullet - v_{\bullet x}^w}{v_{\bullet x}^w}, & v_{\bullet x}^w > R_{w\bullet}\omega_\bullet, \qquad v_{\bullet x}^w \neq 0, \end{cases} \tag{2.9}$$

and

$$\alpha_\bullet = \arctan\left(\frac{v_{\bullet y}^w}{v_{\bullet x}^w}\right). \tag{2.10}$$

The velocities of the wheels can be determined using the following equations:

$$v_{fx}^w = \quad v_{fx}\cos\delta_f + v_{fy}\sin\delta_f, \quad v_{rx}^w = v_{rx}, \tag{2.11}$$

$$v_{fy}^w = -v_{fx}\sin\delta_f + v_{fy}\cos\delta_f, \quad v_{ry}^w = v_{ry}, \tag{2.12}$$

and the velocities $v_{\bullet x}$ and $v_{\bullet y}$ using:

$$v_{fx} = v_x, \quad v_{fy} = v_y + l_f\dot{\theta}_z, \quad v_{rx} = v_x, \quad v_{ry} = v_y - l_r\dot{\theta}_z. \tag{2.13}$$

In general, assuming flat road (road slope of $\phi = 0°$), and neglecting air drag and suspension model, vertical tyre load forces are:

$$F_{fz} = \frac{m}{2}\left(g\frac{l_r}{l_f+l_r} - a_x\frac{h}{l_f+l_r}\right), \quad F_{rz} = \frac{m}{2}\left(g\frac{l_f}{l_f+l_r} + a_x\frac{h}{l_f+l_r}\right), \tag{2.14}$$

where $a_x = \dot{v}_x - \dot{\theta}_z v_y$ is the longutidinal acceleration, h height of the center of gravity (CoG), and g the gravitational constant. However, in this thesis, load transfer effects are disregarded, and the calculation of vertical forces is performed as:

$$F_{fz} = \frac{mgl_r}{l_f+l_r}, \quad F_{rz} = \frac{mgl_f}{l_f+l_r}. \tag{2.15}$$

2.1.2 *Two-track model*

Contrary to the previous section, here the two-track is described and depicted in Figure2.2. This model assumes that the vehicle has four independent wheels and has front-wheel steering. The model can be formulated as follows:

$$m\dot{v}_x = \quad m\dot{\theta}_z v_y + F_{flx} + F_{frx} + F_{rlx} + F_{rrx} - \frac{1}{2}c_w\rho A_w v_x\sqrt{v_x^2+v_y^2}, \tag{2.16}$$

$$m\dot{v}_y = -m\dot{\theta}_z v_x + F_{fly} + F_{fry} + F_{rly} + F_{rry} - \frac{1}{2}c_w\rho A_w v_y\sqrt{v_x^2+v_y^2}, \tag{2.17}$$

$$J_z\ddot{\theta}_z = l_f\left(F_{fly}+F_{fry}\right) - l_r\left(F_{rly}+F_{rry}\right) + w\left(-F_{flx}+F_{frx}-F_{rlx}+F_{rrx}\right), \tag{2.18}$$

$$J_w\dot{\omega}_{\bullet\star} = T_{\bullet\star} - f_{w\bullet}F_{\bullet\star z} - R_{w\bullet}F_{\bullet\star x}^w - b_{w\bullet}\omega_{\bullet\star}, \tag{2.19}$$

$$\dot{X} = v_x\cos\theta_z - v_y\sin\theta_z, \tag{2.20}$$

$$\dot{Y} = v_x\sin\theta_z + v_y\cos\theta_z. \tag{2.21}$$

The symbols $\bullet \in \{f, r\}$ denote the front and rear wheels, while $\star \in \{l, r\}$ correspond to the left and right wheels. Most of the variables are the same as in Section2.1.1, with a few additions. The marking w stands for half the track width of the vehicle. The forces acting on the vehicle body in the longitudinal and lateral directions are symbolized by $F_{\bullet\star x}$ and $F_{\bullet\star y}$ respectively. In the wheel dynamics equation, the wheel rotational velocities are $\omega_{\bullet\star}$, the input wheel torques $T_{\bullet\star}$, the longitudinal wheel forces $F_{\bullet\star x}^w$ and the vertical wheel forces $F_{\bullet\star z}$.

The longitudinal and lateral forces impacting the vehicle's chassis are denoted by

$$\begin{aligned} F_{f\star x} &= F_{f\star x}^w\cos\delta_f - F_{f\star y}^w\sin\delta_f, \quad F_{r\star x} = F_{r\star x}^w, \\ F_{f\star y} &= F_{f\star x}^w\sin\delta_f + F_{f\star y}^w\cos\delta_f, \quad F_{r\star y} = F_{r\star y}^w. \end{aligned} \tag{2.22}$$

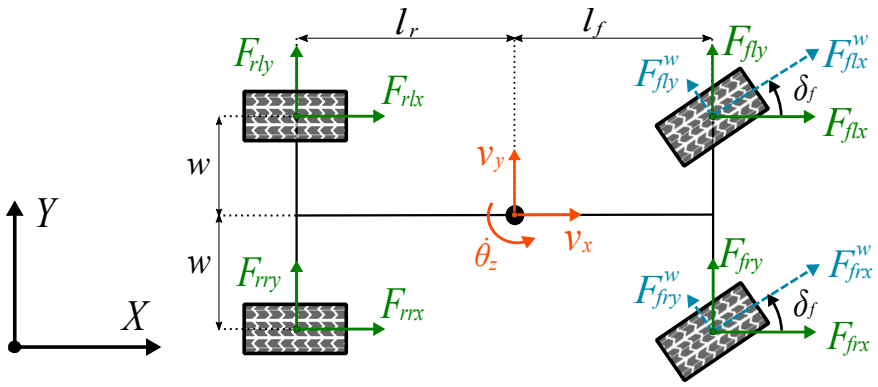

Figure 2.2: Two-track model of a vehicle.

The tire forces are nonlinear and can be described by the following models:

$$\begin{aligned} F^w_{\bullet\star x} &= f^w_x(\alpha_{\bullet\star}, s_{\bullet\star x}, \mu, F_{\bullet\star z}), \\ F^w_{\bullet\star y} &= f^w_y(\alpha_{\bullet\star}, s_{\bullet\star x}, \mu, F_{\bullet\star z}). \end{aligned} \tag{2.23}$$

The longitudinal slip, $s_{\bullet\star x}$, and the lateral slip angle, $\alpha_{\bullet\star}$, are commonly defined as:

$$s_{\bullet\star x} = \begin{cases} \frac{R_{w\bullet}\omega_{\bullet\star} - v^w_{\bullet\star x}}{R_{w\bullet}\omega_{\bullet\star}}, & v^w_{\bullet\star x} \le R_{w\bullet}\omega_{\bullet\star}, \quad R_{w\bullet}\omega_{\bullet\star} \neq 0, \\ \frac{R_{w\bullet}\omega_{\bullet\star} - v^w_{\bullet\star x}}{v^w_{\bullet\star x}}, & v^w_{\bullet\star x} > R_{w\bullet}\omega_{\bullet\star}, \qquad v^w_{\bullet\star x} \neq 0, \end{cases} \tag{2.24}$$

and

$$\alpha_{\bullet\star} = \arctan\left(\frac{v^w_{\bullet\star y}}{v^w_{\bullet\star x}}\right). \tag{2.25}$$

Under the same assumptions as in (2.14), vertical tyre load forces are:

$$F_{flz} = \frac{m}{2}\left(g\frac{l_r}{l_f + l_r} - a_x\frac{h}{l_f + l_r} - a_y\frac{h}{w}\right), \tag{2.26a}$$

$$F_{frz} = \frac{m}{2}\left(g\frac{l_r}{l_f + l_r} - a_x\frac{h}{l_f + l_r} + a_y\frac{h}{w}\right), \tag{2.26b}$$

$$F_{rlz} = \frac{m}{2}\left(g\frac{l_f}{l_f + l_r} + a_x\frac{h}{l_f + l_r} - a_y\frac{h}{w}\right), \tag{2.26c}$$

$$F_{rrz} = \frac{m}{2}\left(g\frac{l_f}{l_f + l_r} + a_x\frac{h}{l_f + l_r} + a_y\frac{h}{w}\right), \tag{2.26d}$$

where $a_x = \dot{v}_x - \dot{\theta}_z v_y$ is the longutidinal and $a_y = \dot{v}_y + \dot{\theta}_z v_x$ lateral acceleration. Once the load transfer effects are disregarded the forces are:

$$F_{f\star z} = \frac{mgl_r}{2(l_f + l_r)}, \quad F_{r\star z} = \frac{mgl_f}{2(l_f + l_r)}. \tag{2.27}$$

The velocities of the wheels in wheel coordinate system are:

$$v^w_{f\star x} = \; v_{f\star x}\cos\delta_f + v_{f\star y}\sin\delta_f, \quad v^w_{r\star x} = v_{r\star x}, \tag{2.28}$$

$$v^w_{f\star y} = -v_{f\star x}\sin\delta_f + v_{f\star y}\cos\delta_f, \quad v^w_{r\star y} = v_{r\star y}. \tag{2.29}$$

The velocities $v_{\bullet\star x}$ and $v_{\bullet\star y}$ can be expressed as:

$$v_{flx} = v_x - w\dot{\theta}_z, \quad v_{fly} = v_y + l_f\dot{\theta}_z, \tag{2.30}$$
$$v_{frx} = v_x + w\dot{\theta}_z, \quad v_{fry} = v_y + l_f\dot{\theta}_z, \tag{2.31}$$
$$v_{rlx} = v_x - w\dot{\theta}_z, \quad v_{rly} = v_y - l_r\dot{\theta}_z, \tag{2.32}$$
$$v_{rrx} = v_x + w\dot{\theta}_z, \quad v_{rry} = v_y - l_r\dot{\theta}_z. \tag{2.33}$$

2.1.3 *Alternative slip formulation*

To enhance the numerical stability of the discussed vehicle models, Micheli et al. proposed a different slip equation as a solution to the issues of dividing by small numbers in equations (2.24) and (2.25), as documented in [5]. This modification becomes especially beneficial under low-speed scenarios. Inspired by their method, here a new definition for longitudinal slip is offered, expressed as:

$$s_{\bullet\star x} = \begin{cases} s^{drive}_{\bullet\star}, & v^w_{\bullet\star x} \leq R_{w\bullet}\omega_{\bullet\star}, \\ s^{brake}_{\bullet\star}, & v^w_{\bullet\star x} > R_{w\bullet}\omega_{\bullet\star}, \end{cases} \tag{2.34}$$

with

$$s^{drive}_{\bullet\star} = \frac{\left(R_{w\bullet}\omega_{\bullet\star} - v^w_{\bullet\star x}\right) R_{w\bullet}\omega_{\bullet\star}}{\left(R_{w\bullet}\omega_{\bullet\star}\right)^2 + \varepsilon_0}, \tag{2.35a}$$
$$s^{brake}_{\bullet\star} = \frac{\left(R_{w\bullet}\omega_{\bullet\star} - v^w_{\bullet\star x}\right) v^w_{\bullet\star x}}{{v^w_{\bullet\star x}}^2 + \varepsilon_0}, \tag{2.35b}$$

and the slip angle as

$$\alpha_{\bullet\star} = \arctan\left(\frac{v^w_{\bullet\star y} v^w_{\bullet\star x}}{{v^w_{\bullet\star x}}^2 + \varepsilon_0}\right). \tag{2.36}$$

Note that ε_0 can be set to arbitrary value, which ever works the best for a given application. The effects of a various ε_0 values are visible in Figure2.3. This shows that the proposed change only affects the model at velocities close to zero.

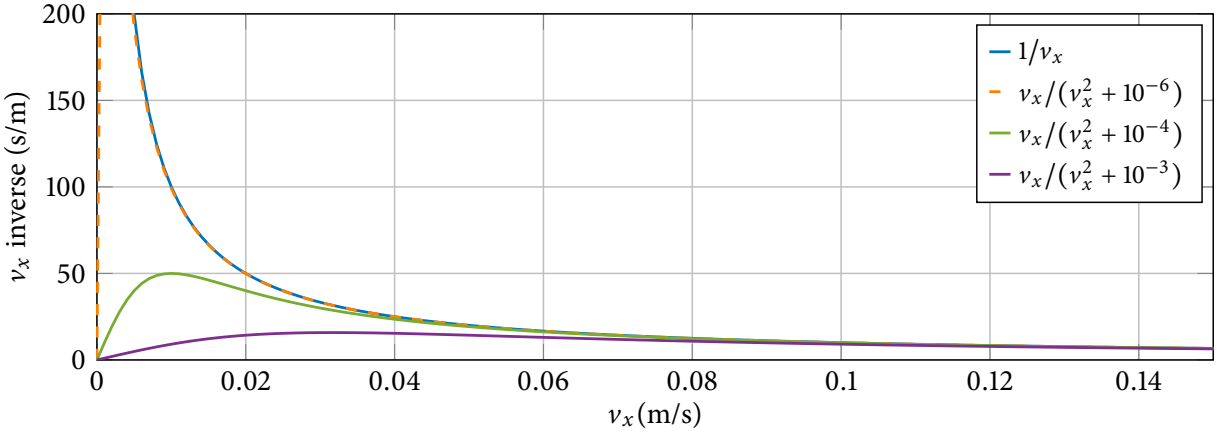

Figure 2.3: Modified slip denominators for different coefficient ε_0.

2.2 TIRE FORCE MODELLING

Components that can significantly affect dynamic behaviour of a vehicle are its tires. In fact, tire-ground interaction is the only source of force and torque generation between the vehicle and the road, so it is important to model it accurately. This chapter focuses on tire models, which aim to describe the complex relationships between forces, torques, and the numerous variables that influence them.

2.2.1 *Magic Formula for tire modelling*

There are plenty of tire force models suggested in the literature, such as the brush tire model [6] or the TMeasy [7] model. However, one of the most widely adopted tire models in both academia and industry is the Magic Formula. This empirical model, developed by Pacejka [8], is known for its ability to fit a wide range of tire characteristics. The Magic Formula is mainly used for describing the lateral and longitudinal forces acting on a tire.

The basic form of the Magic Formula for tire force $F(\sigma)$ is given by:

$$F(\sigma) = D \sin\left(C \arctan\left(B\sigma - E(B\sigma - \arctan(B\sigma))\right)\right), \tag{2.37}$$

where σ represents the slip generalization, i.e. $\sigma \in \{s_x, \alpha\}$, depending on whether we are modelling longitudinal or lateral force. In other words, $F_x = F(s_x)$, while $F_y = F(\alpha)$. The tire model coefficients are denoted by B, C, D, and E. The coefficients are usually determined through curve fitting of empirical tire data, and are generally assumed to be different for longitudinal and lateral forces. These parameters can also vary depending on several conditions like load, inflation pressure, and tire temperature, among others. The Magic Formula's adaptability and accuracy in various operating conditions make it a go-to choice for realistic vehicle simulations.

2.2.2 *Linear tire model*

While the Magic Formula provides an excellent approximation of real-world tire behaviour, there are instances where a simpler, linear model may suffice, especially in control system design or for real-time simulations. A linear tire model can be derived based on the Magic Formula.

The linearized tire model can be obtained by taking the first-order Taylor series expansion of the Magic Formula around a small slip value:

$$F \approx F_0 + \left.\frac{dF}{d\sigma}\right|_{\sigma=\sigma_0} (\sigma - \sigma_0) \tag{2.38}$$

In this equation, F_0 is the force at the nominal (or operating) slip value σ_0, and $\frac{dF}{d\sigma}$ is the rate of change of lateral force with respect to the slip at σ_0.

The linear tire model assumes that this rate of change, often called the longitudinal stiffness for longitudinal forces and cornering stiffness for lateral forces, remains constant over a small range of slip values. Usually, one chooses origin ($s_{x0} = 0$ and $\alpha_0 = 0°$) as a stationary point, in which case the linear tire models can be written as:

$$F_x^{lin} = \left.\frac{dF_y}{ds_x}\right|_{s_x=s_{x0}} = C_x s_x, \quad F_y^{lin} = \left.\frac{dF_y}{d\alpha}\right|_{\alpha=\alpha_0} = -C_y \alpha. \tag{2.39}$$

By linearizing the Magic Formula, one can create a simplified model that is computationally efficient and suitable for control systems design. However, it is important to note that the linear model may not be accurate for larger deviations in the operating conditions, especially when it comes to high-speed cornering or rapid changes in road surface friction.

2.2.3 *Tire force coupling*

Earlier sections explored the production of lateral and longitudinal tire forces when either pure longitudinal slip or pure slip angle is present. However, when both types of slip coexist, adjustments to the tire force equations are required. This is to ensure that the vector sum of the generated forces does not surpass the maximum allowable limit. This phenomenon can be effectively represented using the concept of the friction circle [9].

The friction circle serves as a graphical tool to visualize the limits of a tire's grip. Within this circle, both longitudinal (acceleration and braking) and lateral (cornering) forces operate. The relationship between these forces is known as tire force coupling and can be mathematically expressed as:

$$F_x^2 + F_y^2 \leq (\mu F_z)^2. \tag{2.40}$$

Here, F_x represents the longitudinal forces, F_y represents the lateral forces, and μF_z is the maximum frictional force that the tire can generate.

In practical terms, this equation implies that as you use more of the tire's capacity for acceleration (F_x), you have less available for cornering (F_y), and vice versa. Understanding this balance is critical for vehicle control and is foundational in both automotive design and high-performance driving.

2.2.4 *Piecewise linear tire model*

In order to keep the simplicity of the linear tire model, but also to account for maximum achievable force value, here a piecewise linear approach is proposed. The model limits the linear tire model to its maximum value and introduces longitudinal and lateral force coupling to emulate the friction circle. The model can be represented by the following equations:

$$\begin{aligned} F_x^{sat} &= \max\left(-\mu F_z, \min\left(C_x s_x, \mu F_z\right)\right), \\ F_y^{sat} &= \max\left(-\mu F_z, \min\left(-C_y \alpha, \mu F_z\right)\right), \end{aligned} \tag{2.41a}$$

$$\begin{gathered} \beta = \arctan\left(\frac{F_y^{sat}}{F_x^{sat}}\right), \quad \|F\| = \min\left(\sqrt{\left(F_x^{sat}\right)^2 + \left(F_y^{sat}\right)^2}, \mu F_z\right), \\ F_x = \|F\| \cos\beta, \quad F_y = \|F\| \sin\beta. \end{gathered} \tag{2.41b}$$

This formulation simplifies the process of model identification by reducing the number of parameters compared to more complex models such as the Magic Formula. Despite the reduction in parameters, this approach still provides a sufficiently accurate representation within the desired operating region for the applications demonstrated in this disertation.

Graphical comparison between described tire models can be seen in Figure2.4.

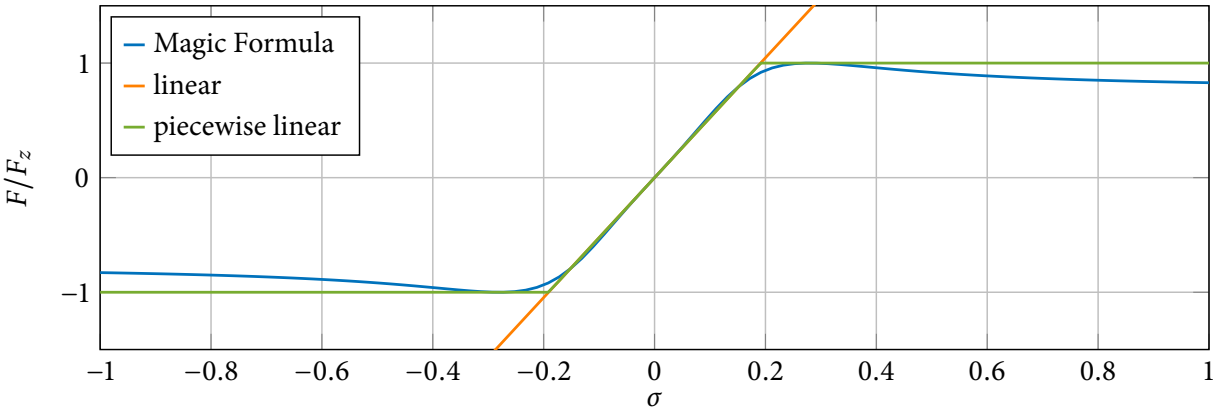

Figure 2.4: Comparison of normalized forces for different tire models without coupling effect (either pure longitudinal slip or pure slip angle).

2.3 MODEL PREDICTIVE CONTROL

Model predictive control (MPC) is a versatile and advanced method in the area of control systems, often outperforming controllers designed using other techniques. Predominantly used for multiple-input, multiple-output (MIMO) systems, it offers a robust framework for dealing with a diverse range of complex scenarios. The idea emerged in the 1960s, but the real success in process industry and theoretical development began in the 1980s [10].

The fundamental logic of MPC resonates with intuitive human decision-making processes. This process involves using a mathematical model to predict system behaviour for a finite number of future time steps. Simultaneously, the controller chooses the most effective future path according to a defined performance measure or cost that needs to be minimized. This concept is known as a receding horizon control (RHC) strategy, where the system state is continuously measured or estimated and new optimal control inputs are calculated at each sampling point on a receding horizon (Figure2.5).

In contrast to MPC, conventional control methods react to errors that occur without predicting the future behaviour of a system. They also do not take into account any constraints, whereas MPC explicitly includes these in its design. Constraints can be physical (like actuator limits), performance-based (such as allowable overshoot), or even safety-related (like temperature or pressure tolerances). Unlike classical control methods that need to keep set points far from these limits, potentially leading to suboptimal operation, MPC computes the optimal set point and operational behaviour while respecting the constraints.

MPC employs algorithms for solving constrained finite-horizon optimal control problems. Advanced computational algorithms have greatly enhanced the speed and reliability of the mathematical calculations essential for MPC. This is crucial, especially when the system has to adapt to unexpected changes or disturbances.

Despite its high performance and systematic constraint handling, MPC does pose some challenges. Ensuring real-time execution, closed-loop stability, robustness, and feasibility can be complex, given the intricate mathematics involved. However, thanks to significant advancements in computer hardware, software, sensors, and communication technologies, the benefit-to-cost ratio for implementing such computationally demanding systems has greatly improved. These technological breakthroughs have not only made MPC more accessible but also suitable for various

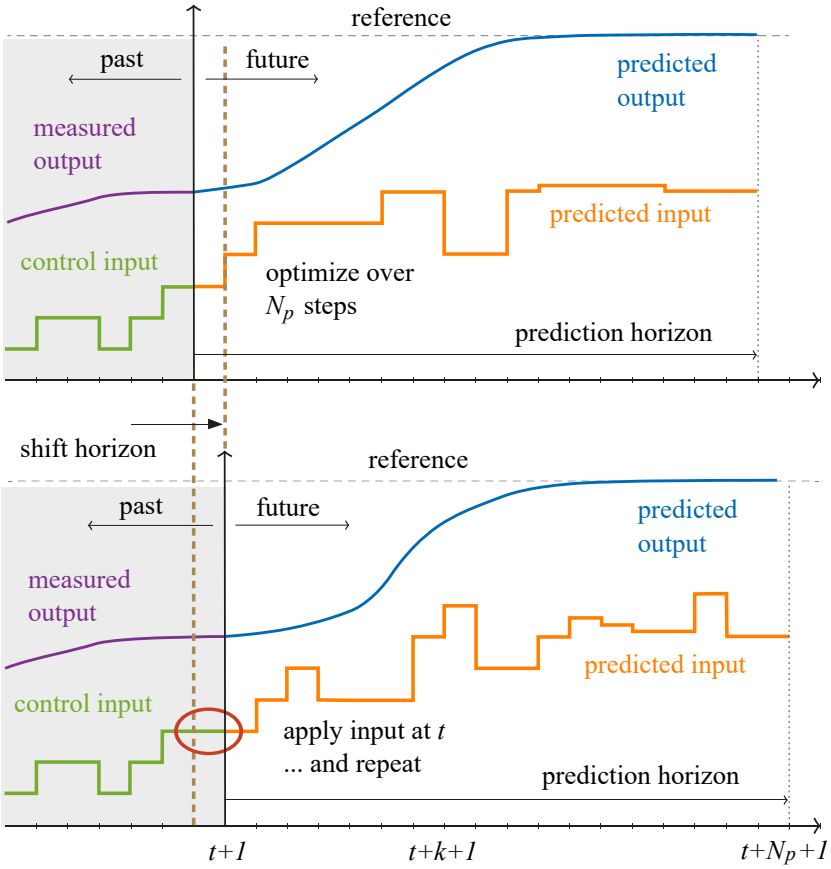

Figure 2.5: Model Predictive Control working principle.

applications beyond the process industry, such as automotive, power systems, and computer control, even down to the milli, micro, and nanosecond timescales.

In summary, model predictive control offers a dynamic, adaptable, and highly efficient approach to system control. It is particularly effective in managing complex MIMO systems and stands out for its ability to consider constraints explicitly, making it invaluable across diverse industries and applications.

2.3.1 *Linear quadratic regulator*

The simple optimal control algorithm, which has no constraints other than the system dynamics and has a quadratic cost function, is called a linear quadratic regulator (LQR). Its task can be interpreted as the regulation of the system state x_k to the origin while minimizing the control effort. For more details on LQR, please refer e.g. to [11].

Finite-horizon discrete-time linear quadratic regulator (FDLQR) is a receding horizon controller with state feedback control law $\mathbf{u} = \kappa(\mathbf{x})$ computed by minimizing the following cost function:

$$\begin{aligned} J(\mathbf{x}_t) := \min_{\mathbf{u}_t,\ldots,\mathbf{u}_{t+N-1}} \quad & \mathbf{x}_{t+N}^T P \mathbf{x}_{t+N} + \sum_{k=t}^{t+N-1} \mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k, \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad k = t, \ldots, t+N-1, \\ & \mathbf{x}_t = \mathbf{x}(t), \end{aligned} \tag{2.42}$$

where $\mathbf{x}_k$ is the state and $\mathbf{u}_k$ the control input at the prediction step k, $P = P^T \succeq 0$, $Q = Q^T \succeq 0$ and $R = R^T > 0$ are weight matrices of corresponding dimensions, $N \in \mathbb{N}$ is prediction horizon and $\mathbf{x}(t)$ (known) initial state value.

In case the prediction horizon is infinite, i.e. $N \to \infty$, one gets what is called discrete linear quadratic regulator (DLQR) and quite often this is the version reffered to when LQR is mentioned in the literature.
The optimization problem (2.42) now becomes

$$\begin{aligned} J_\infty(\mathbf{x}_t) := \min_{\{\mathbf{u}_t,\dots\}} \quad & \sum_{k=t}^{\infty} \mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k, \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad k = t, \dots \\ & \mathbf{x}_t = \mathbf{x}(t). \end{aligned} \tag{2.43}$$

The optimal control law can be expressed as

$$\mathbf{u}_t = -K_\infty \mathbf{x}_t, \tag{2.44}$$

with

$$K_\infty = \left[R + B^T P_\infty B\right]^{-1} B^T P_\infty A, \tag{2.45}$$

where P_∞ is the solution of the discrete algebraic Riccati equation:

$$P_\infty = Q + A^T S P_\infty \left[I_n + B R^{-1} B^T P_\infty\right]^{-1} A. \tag{2.46}$$

Additionally to positive definitness condition of matrices Q and R, in order to guarantee that a unique solution exists that asymptotically stabilizes the closed-loop system, the pair (A, B) is has to be stabilizable and the pair (A, Q) has to be detectable. Under those assumptions the following holds:

- The equation (2.46) has a unique positive semi-definite solution, $P_\infty \succeq 0$.
- With the control law (2.44) and control matrix (2.45) the closed-loop system

$$\mathbf{x}_{t+1} = \left(A - BK_\infty\right) \mathbf{x}_t \tag{2.47}$$

  is asymptotically stable.

- The optimal cost function value is given by

$$J_\infty^*(\mathbf{x}_t) = \mathbf{x}_t^T P_\infty \mathbf{x}_t, \tag{2.48}$$

  which is, at the same time, Lyapunov function of the closed-loop system.

General convention is to ommit ∞ symbol when expressing Lyapunov function and control law, in other words $P = P_\infty$ and $K = K_\infty$.

2.3.2 *Linear model predictive control*

Linear model predictive control (LMPC) problem can be defined similarly to the LQR problem (2.42) with additional linear inequality constraints included:

$$\begin{aligned} J(\mathbf{x}_t) := \min_{\mathbf{u}_t,\dots,\mathbf{u}_{t+N-1}} \quad & J_f(\mathbf{x}_{t+N}) + \sum_{k=t}^{t+N-1} J_s(\mathbf{x}_k, \mathbf{u}_k), \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad k = t, \dots, t+N-1, \\ & (\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{X} \times \mathbb{U}, \\ & \mathbf{x}_{t+N} \in \mathbb{X}_f, \\ & \mathbf{x}_t = \mathbf{x}(t). \end{aligned} \tag{2.49}$$

In this problem $J_s(\mathbf{x}_k, \mathbf{u}_k)$ is the stage cost and $J_f(\mathbf{x}_{t+N})$ is a terminal cost function. State and input constraints are represented by polyhedral sets $\mathbb{X}$ and $\mathbb{U}$, while $\mathbb{X}_f \subseteq \mathbb{X}$ is a polyhedral terminal target set or terminal constraint. The solution of the problem are the optimal state sequence $[\mathbf{x}_t, \mathbf{x}_{t+1}, \dots, \mathbf{x}_{t+N}]$ and the optimal control input sequence $[\mathbf{u}_t, \mathbf{u}_{t+1}, \dots, \mathbf{u}_{t+N-1}]$ [12].

Most common choices for the stage and terminal cost are quadratic and linear norm cost:

- **Quadratic cost**: The stage and terminal cost are given by quadratic functions

$$J_s(\mathbf{x}, \mathbf{u}) := \|\mathbf{x}\|_Q^2 + \|\mathbf{u}\|_R^2 = \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}, \quad J_f(\mathbf{x}) := \|\mathbf{x}\|_P^2 = \mathbf{x}^T P \mathbf{x}, \tag{2.50}$$

  where $P = P^T \succeq 0$, $Q = Q^T \succeq 0$ and $R = R^T \succ 0$ are weight matrices of corresponding dimensions. MPC with quadratic cost can be translated into a quadratic program(QP).

- **Linear norm cost**: The stage and terminal cost are formed by l_1- or l_∞-norms

$$J_s(\mathbf{x}, \mathbf{u}) := \|Q\mathbf{x}\|_p + \|R\mathbf{u}\|_p, \quad J_f(\mathbf{x}) := \|P\mathbf{x}\|_p \quad \text{with } p \in \{1, \infty\}. \tag{2.51}$$

  Matrices Q and R are assumed to be non-singular, whereas P is assumed to have a full column rank. Optimization problem with linear norm cost can be translated into a linear program(LP).

It has been stated in the section2.3.1that, for the special case of an infinite horizon ($N \to \infty$) the closed-loop system with this controller has some useful properties, such as unique solution and guaranteed stability. The RHC law does not necessarily have those properties. Moreover, it can be proven that, in general, stability and feasibility are not ensured by the RHC law [13]. The following theorem states the necessary assumptions for proving stability and feasibility.

Theorem 2.1. *Let $\mathbb{X}_0$ be the set of feasible initial states and $\mathbf{x}^+$ the state at the next sampling time. Consider that the following assumptions hold:*

1. *The stage cost is a positive definite function, i.e. it is strictly positive and zero only at the origin.*
2. *The terminal set is positively invariant under the local control law $\kappa(\mathbf{x})$:*

$$\mathbf{x}^+ = A\mathbf{x} + B\kappa(\mathbf{x}) \in \mathbb{X}_f \quad \forall \mathbf{x} \in \mathbb{X}_f.$$

   All state and input constraints are satisfied in $\mathbb{X}_f$:

$$\mathbb{X}_f \subseteq \mathbb{X}, \quad \kappa(\mathbf{x}) \in \mathbb{U} \quad \forall \mathbf{x} \in \mathbb{X}_f.$$

3. *The terminal cost is a continuous Lyapunov function in the terminal set* $\mathbb{X}_f$*:*

$$J_f(\mathbf{x}^+) - J_f(\mathbf{x}) \leq -J_s(\mathbf{x}, \kappa(\mathbf{x})) \quad \forall \mathbf{x} \in \mathbb{X}_f.$$

Then the closed-loop system under the MPC control law is stable in $\mathbb{X}_0$*.*

Since terminal sets require advanced tools to compute and reduce region of attraction, they can usually be ommited [14].

To sumarize, when linear MPC is mentioned, this usually means that the optimization problem consisting has a quadratic cost function, linear equality constraints (linear system dynamics) and linear inequality constraints (polyhedral sets). Such optimization problem has the following formulation:

$$\begin{aligned} J(\mathbf{x}_t) := \min_{\mathbf{u}_t,\ldots,\mathbf{u}_{t+N-1}} \quad & \mathbf{x}_{t+N}^T P \mathbf{x}_{t+N} + \sum_{k=t}^{t+N-1} \mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k, \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad k = t, \ldots, t+N-1, \\ & F\mathbf{x}_k + G\mathbf{u}_k \leq \mathbf{h}, \\ & F_N \mathbf{x}_{t+N} \leq \mathbf{h}_N, \\ & \mathbf{x}_t = \mathbf{x}(t), \end{aligned} \tag{2.52}$$

with F and G and F_N being matrices and $\mathbf{h}$ and $\mathbf{h}_N$ constraint vector of corresponding dimensions.

2.3.3 *Dense vs. sparse formulations*

To solve MPC problems numerically, one typically utilizes a pre-existing solver. For LMPC as represented in equation (2.52), a solver designed for quadratic programming, such as simplex or barrier solver from Gurobi [15] or OSQP [16], is employed. In this context, the MPC problem should be shaped into a standard quadratic program as depicted below:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\mathbf{z}^T H_{qp} \mathbf{z} + \mathbf{f}_{qp}^T \mathbf{z}, \\ \text{s.t.} \quad & A_{in}\mathbf{z} \leq \mathbf{b}_{in}, \\ & A_{eq}\mathbf{z} = \mathbf{b}_{eq}. \end{aligned} \tag{2.53}$$

In the above equation, H_{qp} represents a positive definite cost matrix $\mathbf{f}_{qp}$ is a linear cost vector, A_{in} and A_{eq} denote linear inequality and equality matrices, while $\mathbf{b}_{in}$ and $\mathbf{b}_{eq}$ are the inequality and equality vectors, respectively. Two different approaches which can be used are here reffered to as the sparse and dense MPC formulations.

Dense formulation: Lets define agrregated state and input vectors as $\mathbf{X}_t = \left[\mathbf{x}_t^T, \mathbf{x}_{t+1}^T, \ldots \mathbf{x}_{t+N}^T\right]^T$ and $\mathbf{U}_t = \left[\mathbf{u}_t^T, \mathbf{u}_{t+1}^T, \ldots \mathbf{u}_{t+N-1}^T\right]^T$. Then one can rewrite the state propagation equality from (2.52) as

$$\mathbf{X}_t = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{U}_t, \tag{2.54}$$

with matrices

$$\mathbf{A} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ B & 0 & \ldots & 0 \\ AB & B & \ldots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & \ldots & B \end{bmatrix}, \tag{2.55}$$

where I is an identity matrix of appropriate size. Combining (2.52) with (2.54), the optimization problem becomes the following:

$$\min_{\mathbf{U}_t,\ \mathbf{x}_t} \quad \mathbf{x}_t^T(\mathbf{A}^T\mathbf{Q}\mathbf{A})\mathbf{x}_t + \mathbf{U}_t^T(\mathbf{B}^T\mathbf{Q}\mathbf{B} + \mathbf{R})\mathbf{U}_t \tag{2.56a}$$

$$\text{s.t.} \quad \mathbf{F}\mathbf{A}\mathbf{x}_t + (\mathbf{F}\mathbf{B} + \mathbf{G})\mathbf{U}_t \leq \mathbf{H}, \tag{2.56b}$$

$$\mathbf{x}_t = \mathbf{x}(t). \tag{2.56c}$$

In the equations (2.56a) - (2.56c), bold symbols denote extended matrices and vectors $\mathbf{Q} = \text{diag}(Q, Q, \ldots, P)$, $\mathbf{R} = \text{diag}(R, R, \ldots, R)$, $\mathbf{F} = \text{diag}(F, F, \ldots, F_N)$, $\mathbf{G} = \text{diag}(G, G, \ldots, G)$ and $\mathbf{H} = \left[\mathbf{h}, \mathbf{h}, \ldots, \mathbf{h}_N\right]^T$ of the corresponding dimesion. In such a formulation, the cost function and constraints contain only the initial state vector $\mathbf{x}_t$ and the input vector sequence $\mathbf{U}_t$, which reduces the size of the optimizer. This QP can be transformed into (2.53) via the following equations:

$$\begin{aligned}
\mathbf{z} &= \left[\mathbf{x}_t^T, \mathbf{U}_t^T\right]^T, \\
H_{qp} &= 2\ \text{diag}\left(\mathbf{A}^T\mathbf{Q}\mathbf{A},\ \mathbf{B}^T\mathbf{Q}\mathbf{B} + \mathbf{R}\right), \quad \mathbf{f}_{qp} = \mathbf{0}, \\
A_{in} &= \text{diag}\left(\mathbf{F}\mathbf{A},\ \mathbf{F}\mathbf{B} + \mathbf{G}\right), \quad \mathbf{b}_{in} = \mathbf{H}, \\
A_{eq} &= \left[I,\ \mathbf{0}\right], \quad \mathbf{b}_{eq} = \mathbf{x}(t).
\end{aligned}$$

The dense formulation results in a smaller optimizer vector size, but may be numerically less stable due to the numerous numerical operations required to determine the optimization vectors and matrices.

Sparse formulation: In contrast to the dense formulation, where the goal is to reduce the size of the optimizer by incorporating the system dynamics into the cost and constraint matrices, the sparse formulation aims to take advantage of a matrix sparsity structure that some solvers can use to solve quadratic programs more efficiently. The main idea is to add all the states and inputs to the optimizer, which leads to the following:

$$\begin{aligned}
\mathbf{z} &= \left[\mathbf{x}_t^T, \mathbf{x}_{t+1}^T, \ldots \mathbf{x}_{t+N}^T, \mathbf{u}_t^T, \mathbf{u}_{t+1}^T, \ldots \mathbf{u}_{t+N-1}^T\right]^T, \\
H_{qp} &= 2\ \text{diag}\left(Q, Q, \ldots, P, R, R, \ldots R\right), \quad \mathbf{f}_{qp} = \mathbf{0}, \\
A_{in} &= \begin{bmatrix} F & 0 & \ldots & 0 & G & 0 & \ldots & 0 \\ 0 & F & \ldots & 0 & 0 & G & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & F_N & 0 & 0 & \ldots & 0 \end{bmatrix}, \quad \mathbf{b}_{in} = \begin{bmatrix} \mathbf{h} \\ \mathbf{h} \\ \vdots \\ \mathbf{h}_N \end{bmatrix}, \\
A_{eq} &= \begin{bmatrix} I & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ A & -I & 0 & \ldots & 0 & B & 0 & \ldots & 0 \\ 0 & A & -I & \ldots & 0 & 0 & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & A & -I & 0 & 0 & \ldots & B \end{bmatrix}, \quad \mathbf{b}_{eq} = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}.
\end{aligned}$$

Although it can be efficiently solved by QP solvers that can use the sparsity pattern to speed up the computations, the sparse formulation usually leads to a large optimizer vector.

Both formulations have their advantages and disadvantages, and it is up to the user to choose which one is more appropriate for a given application. However, it is important to note that the sparse formulation can be advantageous when the state-space is relatively small, while for applications with a large number of states, the dense formulation is usually more efficient.

2.3.4 *Nonlinear model predictive control*

Nonlinear model predictive control (NMPC) has emerged as a compelling alternative to its linear counterpart for solving complex control problems that involve nonlinear system dynamics, which are quite comon in engineering and science. As the name suggests, NMPC employs a nonlinear model of the system to predict future states and calculate the control inputs that optimize a particular objective function. As in LMPC, objective function is usually a composite metric that aims to minimize tracking errors, control effort, and potentially other objectives.

The mathematical foundation of NMPC involves solving a constrained nonlinear optimization problem at each time step. The problem can be formally written as follows:

$$\begin{aligned} J(\mathbf{x}_t) := \min_{\mathbf{u}_t,\ldots,\mathbf{u}_{t+N-1}} \quad & J_f(\mathbf{x}_{t+N}) + \sum_{k=t}^{t+N-1} J_s\left(\mathbf{x}_k, \mathbf{u}_k\right) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad k = t, \ldots, t+N-1, \\ & (\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{X} \times \mathbb{U}, \\ & \mathbf{x}_{t+N} \in \mathbb{X}_f \\ & \mathbf{x}_t = \mathbf{x}(t), \end{aligned} \tag{2.57}$$

where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ is the nonlinear system dynamics and $J_s\left(\mathbf{x}_k, \mathbf{u}_k\right)$ and $J_f(\mathbf{x}_{t+N})$ (potentially) nonlinear stage and terminal cost, respectively. State set $\mathbb{X}$ and input set $\mathbb{U}$, as well as terminal set $\mathbb{X}_f$ can now be described by nonlinear inequalites and don't have to be convex. Since the stability and feasibility assumptions presented in the Section2.3.2did not rely on linearity, results can be directly extended to the problem2.57. However, computing the function $J_f(\mathbf{x}_{t+N})$ and the set $\mathbb{X}_f$ can be very difficult. For NMPC, this is usually computed based on linearization, and the terminal set and terminal cost for the linearized system are used around the equilibrium.

Just as it is the case with LMPC, to avoid generalization using set notation, NMPC can be written as:

$$\begin{aligned} J(\mathbf{x}_t) := \min_{\mathbf{u}_t,\ldots,\mathbf{u}_{t+N-1}} \quad & J_f(\mathbf{x}_{t+N}) + \sum_{k=t}^{t+N-1} J_s\left(\mathbf{x}_k, \mathbf{u}_k\right) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad k = t, \ldots, t+N-1, \\ & \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq \mathbf{0}, \\ & \mathbf{h}_N(\mathbf{x}_{t+N}) \leq \mathbf{0}, \\ & \mathbf{x}_t = \mathbf{x}(t). \end{aligned} \tag{2.58}$$

In the previous equation, $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)$ and $\mathbf{h}_N(\mathbf{x}_{t+N})$ are vector-valued (nonlinear) functions representing stage and terminal constraints.

Due to the nonlinear nature of the optimization problem, solving NMPC in real time becomes computationally demanding. Some of the common techniques to solve these challenges are mentioned here:

- *Interior-point methods*: Interior-point methods are designed to handle inequality constraints effectively by transforming them into equality constraints. A barrier function is introduced to the original problem, thereby creating a new problem that only involves equality constraints. The solutions to the modified problem converge to the original problem as the barrier parameter approaches zero [17].

- *Sequential quadratic programming (SQP)*: One of the most commonly used methods for solving nonlinear optimization problems is sequential quadratic programming. SQP approximates the nonlinear problem by solving a sequence of quadratic sub-problems. This is done by linearizing the constraints and approximating the objective function around the current point. Then, the quadratic problem is solved, and the solution serves as the initial guess for the next iteration. This process continues until convergence criteria are met [18].

- *Real-time iteration (RTI) scheme*: In real-time applications where computational resources are limited, the real-time iteration scheme can be particularly useful. The RTI scheme splits the nonlinear optimization problem into preparation and feedback phase. Preparation phase includes warm-starting the optimization using initial guess based on the solution from the previous iteration. During that time, measurement and/or state estimation is performed to obtain the latest information about the system states, after which the quadratic sub-problem is solved. Contraray to SQP in which the optimization is done to full convergence, here only one iteration of the underlying optimization algorithm is applied. The process is repeated in real time, allowing NMPC to be used in fast-changing systems [18].

- *Approximation methods*: When the control problem is too complex to be solved exactly in real time, approximation methods such as explicit NMPC [19] or machine learning-based approaches can be used (as in [20] or [21]). These methods aim to approximate the optimal control law either by partitioning the state-space or by training a model to emulate the control behaviour.

In summary, NMPC provides a flexible and powerful framework to control nonlinear systems effectively. However, solving the associated nonlinear optimization problem in real time remains a challenge. A variety of methods are available to tackle this challenge, each with its pros and cons. The choice of method often depends on the specific requirements of the application, such as the need for real-time performance, the complexity of the nonlinearities, and the availability of computational resources.

2.3.5 *Reference tracking*

Many real-world control applications are focused on following a desired reference trajectory rather than merely stabilizing around a specific steady state. To accomplish this, the standard MPC algorithm can be adapted to include a penalty term for deviations from both the desired state and input references. The modified cost function is expressed as:

$$
\begin{aligned}
J(\mathbf{x}_t, \mathbf{X}_{ref}, \mathbf{U}_{ref}) := \min_{\mathbf{u}_t,\dots,\mathbf{u}_{t+N-1}} \quad & J_f(\mathbf{x}_{t+N} - \mathbf{x}^{ref}_{t+N}) + \sum_{k=t}^{t+N-1} J_s\left(\mathbf{x}_k - \mathbf{x}^{ref}_k, \mathbf{u}_k - \mathbf{u}^{ref}_k\right), \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad k = t, \dots, t+N-1, \\
& (\mathbf{x}_k, \mathbf{u}_k) \in \mathbb{X} \times \mathbb{U}, \\
& \mathbf{x}_{t+N} \in \mathbb{X}^{ref}_f \\
& \mathbf{x}_t = \mathbf{x}(t).
\end{aligned}
\tag{2.59}
$$

Equation (2.59) shows linear MPC with time dependant input and state reference signals. In it, $\mathbf{x}^{ref}_k$ and $\mathbf{u}^{ref}_k$ are state and input references time step k and $\mathbb{X}^{ref}_f$ is an invariant terminal target

set that is parametrized by the reference. State and input reference sequences are marked by $\mathbf{X}_{ref} = [\mathbf{x}_t^{ref}, \mathbf{x}_{t+1}^{ref}, \ldots, \mathbf{x}_{t+N}^{ref}]$ and $\mathbf{U}_{ref} = [\mathbf{u}_t^{ref}, \mathbf{u}_{t+1}^{ref}, \ldots, \mathbf{u}_{t+N}^{ref}]$, respectively.

Alternatively, one can also penalize the output $\mathbf{y}_k = C\mathbf{x}_k$ for deviating from its desired reference $\mathbf{y}_k^{ref}$. Both of these strategies produce comparable outcomes but are reliant on the accuracy of the model being used. If the model is not precise, the output will not closely follow the desired reference. Various techniques for achieving accurate, offset-free tracking of references have been studied, as indicated in literature such as [12] and the references therein.

2.3.6 *Soft constraints*

Constraints within control algorithm can be categorized as either hard or soft. Hard constraints usually result from the physical limitations of the actuators or the system as a whole. These are non-negotiable limits that must never be exceeded. Soft constraints, on the other hand, are more flexible and relate to preferred performance and safety limits of the system. While hard constraints are usually tied to inputs and strictly must not be violated, system constraints are generally soft. A violation of these soft constraints is permitted for a short period of time, usually due to errors in the system modelling or external disturbances. In standard MPC formulations, such as2.49or 2.57, any breach of these constraints would render the problem infeasible. To overcome this, soft constrained MPC introduces slack variables, denoted as ε_k, into the cost function:

$$
\begin{aligned}
J(\mathbf{x}_t) := \min_{\substack{\mathbf{u}_t,\ldots,\mathbf{u}_{t+N-1} \\ \varepsilon_t,\ldots,\varepsilon_{t+N-1}}} \quad & J_f(\mathbf{x}_{t+N}) + J_\varepsilon(\varepsilon_{t+N}) + \sum_{k=t}^{t+N-1} J_s(\mathbf{x}_k, \mathbf{u}_k) + J_\varepsilon(\varepsilon_k) \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad k = t, \ldots, t+N-1, \\
& F\mathbf{x}_k + G\mathbf{u}_k \leq \mathbf{h} + \varepsilon_k, \\
& F_N\mathbf{x}_{t+N} \leq \mathbf{h}_N + \varepsilon_{t+N}, \\
& \mathbf{x}_t = \mathbf{x}(t),
\end{aligned}
\tag{2.60}
$$

If no constraints are violated, the slack ε_k variables will be forced by the cost term $J_\varepsilon(\varepsilon_k)$ to have zero value, making the soft constrained MPC formulation equivalent to the standard MPC. However, if constraints are only mildly exceeded, the problem remains solvable [22].

2.4 KOOPMAN OPERATOR

The last section mentions both the advantages and weaknesses of linear and nonlinear MPC. Ideally, one would like to have the predictive capabilities of NMPC while maintaining the simplicity and efficiency of LMPC. The Koopman operator presents an approach that effectively combines these properties. Recent developments acknowledge the Koopman operator theory as a novel perspective on dynamical systems, highlighting the evolution of measurements $g(\mathbf{x})$ of a state $\mathbf{x}$. Bernard O. Koopman, in 1931, ilustrated the ability to represent a nonlinear dynamical system via an infinite-dimensional linear operator within a Hilbert space of measurement functions tied to the system's state [23], which was later generalized by Koopman and von Neumann to systems with continuous eigenvalue spectrum [24]. Known as the Koopman operator, its linearity and spectral decomposition provide a comprehensive insight into the behaviour of nonlinear systems. However, its infinite-dimensional character, which is a result of unbounded degrees of

freedom which describe all conceivable measurement functions g of the state vector $\mathbf{x}$, reduces its practicality.

For a long time there was practially no advancement in the filed, until Mezić and his collaborators brought it to life at the begining of this century [25, 26, 27, 28, 29]. Ever since then, the primary research focus is to formulate matrix-based, finite-dimensional approximations of the Koopman operator, with the eventual aim of developing globally linear models of nonlinear systems. The attraction of translating nonlinear dynamics into linear terms is a result of the superior estimation and prediction capabilities of nonlinear models, along with the numerous control methods available for linear systems. Nonetheless, developing a finite-dimensional version of the Koopman operator is pragmatically challenging, requiring a precise determination of a large, yet finite, number of unknown observable functions [30].

2.4.1 *Theoretical background*

The Koopman operator advances measurement functions of the state with the flow of the dynamics. We examine complex-valued measurement functions, denoted $g : X \to \mathbb{C}$ which belong to an infinite-dimensional Hilbert space. Often referred to as observables, these functions are distinct from the unrelated concept of the observability in control theory.

To begin with, lets consider a discrete-time autonomous system:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{f}(\mathbf{x}_t), \\ \mathbf{y}_t &= \mathbf{h}(\mathbf{x}_t), \end{aligned} \tag{2.61}$$

where the current state vector is $\mathbf{x}_t \in \mathbb{X} \subset \mathbb{R}^{n_x}$, the subsequent state vector $\mathbf{x}_{t+1} \in \mathbb{X} \subset \mathbb{R}^{n_x}$ and the current output vector $\mathbf{y}_t \in \mathbb{Y} \subset \mathbb{R}^{n_y}$. The Koopman operator $\mathcal{K} : \mathcal{H} \to \mathcal{H}$ is an infinite-dimensional linear operator that acts on measurement functions g according to:

$$\mathcal{K}g = g \circ \mathbf{f}, \tag{2.62}$$

where $\circ$ is the composition operator. For a discrete-time system (2.61), this becomes:

$$\mathcal{K}g(\mathbf{x}_t) = g(\mathbf{f}(\mathbf{x}_t)) = g(\mathbf{x}_{t+1}), \tag{2.63}$$

Concisely, the Koopman operator formulates an infinite-dimensional linear dynamical system that advances the state observation $g_t = g(\mathbf{x}_t)$ to the succeeding time step:

$$g(\mathbf{x}_{t+1}) = \mathcal{K}g(\mathbf{x}_t). \tag{2.64}$$

It is crucial to note that this is applicable for any observable function g and for any state $\mathbf{x}_t$. Schematic overview of the method is depicted in Figure2.6.

The Koopman operator is linear, which follows from the linearity of the addition operation in function spaces:

$$\begin{aligned} \mathcal{K}(\alpha_1 g_1(\mathbf{x}) + \alpha_2 g_2(\mathbf{x})) &= \alpha_1 g_1(\mathbf{f}(\mathbf{x})) + \alpha_2 g_2(\mathbf{f}(\mathbf{x}_k)) \\ &= \alpha_1 \mathcal{K} g_1(\mathbf{x}) + \alpha_2 \mathcal{K} g_2(\mathbf{x}). \end{aligned} \tag{2.65}$$

The linear dynamical system (2.64) is analogous to the dynamical systems (2.61). It is noteworthy that the original state $\mathbf{x}$ could serve as the observable, with the infinite-dimensional operator

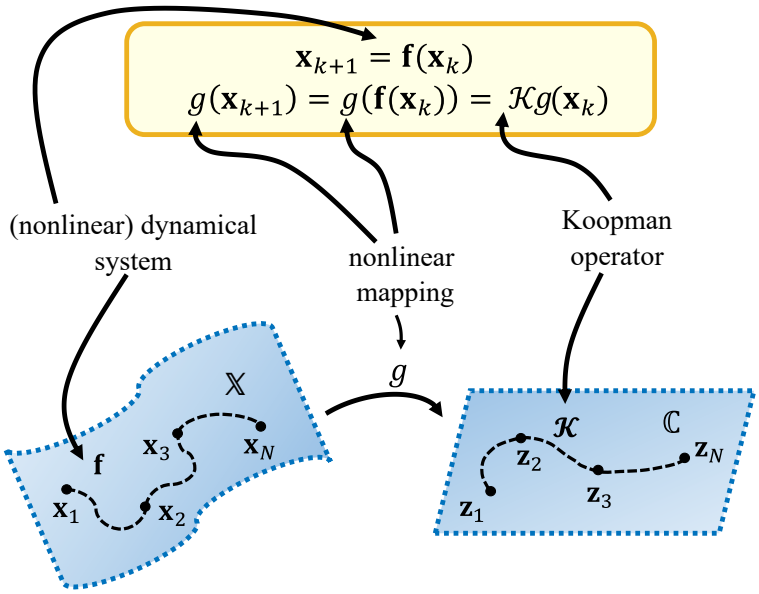

Figure 2.6: Schematic illustration the Koopman operator.

$\mathcal{K}$ still advancing this function. Nonetheless, while the observable $g = \mathbf{x}$ might present a straightforward representation within a selected basis for Hilbert space, it may evolve into an exceedingly complex form upon successive iterations through the dynamics. Thus, deriving a representation for $\mathcal{K}$ may not be simple or straightforward.

For sufficiently smooth dynamical systems, it is also possible to define the continuous-time analogue of the Koopman dynamical system

$$\dot{g}(\mathbf{x}) = \mathcal{K}g(\mathbf{x}). \tag{2.66}$$

Although continuous-time Koopman systems won't be discussed in this disertation, the following example tries to clarify the main idea of the Koopman operator for the case where its finite dimenstional representation exists [31].

Example 2.1. *Consider an example system with a single fixed point, described as follows:*

$$\begin{aligned} \dot{x}_1 &= \mu x_1 \\ \dot{x}_2 &= \lambda(x_2 - x_1^2) \end{aligned} \tag{2.67}$$

When $\lambda < \mu < 0$, this system demonstrates a slow attracting manifold specified by $x_2 = x_1^2$. By incorporating the nonlinear measurement $g = x_1^2$ into the state $\mathbf{x}$, one can construct a three-dimensional Koopman invariant subspace. Within this framework, the system's behaviour is represented by linear dynamics:

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix}. \tag{2.68}$$

2.4.2 *Koopman operator eigenfuncions*

The Koopman operator possesses linearity, an attribute that is notably advantageous, yet its infinite-dimensionality introduces challenges in representation and computation. Rather than

encapsulating the progression of all measurement functions within a Hilbert space, applied Koopman analysis attempts to pinpoint crucial measurement functions that linearly evolve with the flow of the dynamics. Eigenfunctions of the Koopman operator provide a unique set of measurements that exhibit linear behaviour over time. Indeed, a main motivation for adopting the Koopman framework lies in its capacity to simplify the dynamics through the eigen-decomposition of the operator.

A discrete-time Koopman eigenfunction $\phi(\mathbf{x})$ corresponding to eigenvalue λ satisfies

$$\phi(\mathbf{x}_{t+1}) = \mathcal{K}\phi(\mathbf{x}_t) = \lambda\phi(\mathbf{x}_t). \tag{2.69}$$

A central challenge in modern applied dynamical systems is how to obtain Koopman eigenfunctions either from data or from analytical expressions. The identification of these eigenfunctions enables globally linear representations of strongly nonlinear systems. In other words, nonlinear dynamics become entirely linear when expressed in eigenfunction coordinates represented by $\phi(\mathbf{x})$. For instance, any conserved quantity within a dynamical system, like the constant function $\phi = 1$, is a Koopman eigenfunction and corresponds to an eigenvalue $\lambda = 0$ for any dynamical system.

Additionally, depending on the dynamical system, there may be a finite set of generator eigenfunction elements that may be used to construct all other eigenfunctions. In discrete time, we find that the product of two eigenfunctions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ is also an eigenfunction

$$\mathcal{K}(\phi_1(\mathbf{x})\phi_2(\mathbf{x})) = \phi_1(\mathbf{f}(\mathbf{x}))\phi_2(\mathbf{f}(\mathbf{x})) = \lambda_1\lambda_2\phi_1(\mathbf{x})\phi_2(\mathbf{x}) \tag{2.70}$$

corresponding to a new eigenvalue $\lambda_1\lambda_2$ given by the product of the eigenvalues of $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$.

2.4.3 *Koopman operator for non-autonomous systems*

The general representation of a discrete-time nonlinear non-autonomous dynamical system is expressed as follows:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{y}_t &= \mathbf{h}(\mathbf{x}_t). \end{aligned} \tag{2.71}$$

In this equation, the current state is $\mathbf{x}_t \in \mathbb{X} \subset \mathbb{R}^{n_x}$, the subsequent state $\mathbf{x}_{t+1} \in \mathbb{X} \subset \mathbb{R}^{n_x}$, the current input $\mathbf{u}_t \in \mathbb{U} \subset \mathbb{R}^{n_u}$ and the current output $\mathbf{y}_t \in \mathbb{Y} \subset \mathbb{R}^{n_y}$. The transition is denoted by $\mathbf{f}$, and the output mapping by $\mathbf{h}$. The augmented state evolution is described by the equation

$$\chi_{t+1} = \hat{\mathbf{f}}(\chi_t) = \begin{bmatrix} \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \\ \mathcal{S}\mathbf{U}_t \end{bmatrix}. \tag{2.72}$$

Here $\chi_t = \begin{bmatrix} \mathbf{x}_t & \mathbf{U}_t \end{bmatrix}^T$ represents the extended state vector and $\mathbf{U}_t = \begin{bmatrix} \mathbf{u}_t, \mathbf{u}_{t+1}, \ldots \mathbf{u}_\infty \end{bmatrix} \in l(\mathbb{U})$ an infinite dimensional sequence of input vectors. $\mathcal{S}$ denotes the left shift operator, where $\mathcal{S}\mathbf{u}_t = \mathbf{u}_{t+1}$. The Koopman operator, denoted as $\mathcal{K} : \mathcal{H} \to \mathcal{H}$, is associated with the dynamics given by equation (2.72), and it is defined as

$$\mathcal{K}\psi(\chi_t) = \psi(\hat{\mathbf{f}}(\chi_t)) = \psi(\chi_{t+1}). \tag{2.73}$$

In the above equation, $\psi : \mathbb{R}^{n_x} \times l(\mathbb{U}) \to \mathbb{C}$ is a lifting function belonging to the function space $\mathcal{H}$, which remains invariant under the action of the Koopman operator. For further insights on this subject, please refer to [32].

2.4.4 *Applications*

Over the past decade, there has been a growing focus on researching the use of Koopman operator methods for state estimation and control. Pioneering work in the field of estimation includes the concept of the Koopman Observer Form (KOF), introduced by Surana et al. in [33] and [34], where the potential of the Koopman operator to construct KOF was demonstrated in a general setting. This approach extends the scope of linear observers similar to Luenberger/Kalman to a wider range of applications.

The application of the Koopman operator in control systems, including practical control scenarios, is described in detail in works such as [35]. These include studies such as experimental applications of the Koopman operator in active learning for control and the application of Koopman-based control in ultrahigh-precision positioning. Additional application examples are closed-loop control of a robot with spherical casing [36], fractional order PID control of a MEMS gyroscope [37] and many others.

Brunton and colleagues [31] were among the first to address the problem of optimal control via the Koopman operator, focusing on the selection of observable functions that enable the application of optimal linear control strategies to nonlinear problems. Their study emphasized the effectiveness of nonlinear observable subspaces in creating Koopman operator-based optimal control laws for fully nonlinear systems using linear optimal control techniques such as LQR.

In the field of predictive control, Korda and Mezić [32] provide a comprehensive theoretical framework for MPC using a Koopman operator-derived linear model (KMPC). Additionally, Zhang [38] and Mamakoukas [39], along with their teams, further investigate robust MPC techniques based on the Koopman model. So far, KMPC has found its application in various engineering fields. The study by Narasingam et al. [40] effectiveness of KMPC in controlling a continuously stirred tank reactor process, achieving robust closed-loop stability. Similarly, Korda et al. [41] applied KMPC for the transient stabilization of a power system. In [42], the authors also demonstrated the application of finite control set KMPC in the control of electrical drives, which is a pioneering real-world application in power electronics. In the field of robotics, Bruder et al. [43] implemented KMPC for soft robot trajectory tracking illustrating its superiority in guiding a robot along a predefined path compared to a standard linear model-based MPC controller. Further experimental validation was done in [44], where a variant of MPC called quasi-LPV MPC was used and tested on a 3-degree-of-freedom gyroscope and demonstrated excellent tracking capabilities. The method described in this paper extends the concept of KMPC by converting the Koopman model into a quasi-LPV model through linearization instead of using it directly. KMPC was also used in nonlinear flow control [45] and model identification as well as feedback control of a hydraulic fracturing process [46]. Chen et al. [47] introduced a data-driven predictive control strategy using the Koopman model for automatic train contol systems to address challenges related to operational safety, comfort, and parking precision amid uncertain train dynamics and actuator constraints. Wang et al. [48] proposed a method combining deep neural networks with Koopman theory for linear modelling and control of nonlinear robotic systems, which improved accuracy and control in mobile robot experiments beyond conventional approaches without the need for pre-existing knowledge of the system dynamics. Furthermore, the authors in [49] addressed the complexity of controlling soft actuators with pronounced nonlinearity and proposed an improved KMPC framework for effective model-based control. Finally, the work described in [50] discusses

the challenges associated with accurate modelling and control of infinite-degree-of-freedom continuum manipulators with high flexibility using KMPC, while in [51] the authors propose a multi-criteria optimization of HVAC operation of buildings using Koopman predictive control and deep learning.

Many different use cases are mentioned in this overview, but it still only represents a small part of the diverse and rapidly changing application possibilities of the Koopman operator. A detailed overview of vehicle dynamics control with the Koopman operator is presented in Chapter5.

2.5 VEHICLE DYNAMICS CONTROL SYSTEMS

The safety mechanisms in motor vehicles can be divided into passive and active systems, each of which offers different ways of dealing with collisions and preventing accidents. Passive systems, such as seat belts and airbags, are activated in the event of an accident in order to prevent injuries. Active systems, on the other hand, work preventively and use various electronically controlled modules to avoid accidents. The latter use a series of sensors and algorithms that manipulate a vehicle's components to assist the driver in maintaining control. This section presents some standard vehicle dynamics control systems, a subset of active safety functions. While these systems differ in their mechanisms, they all aim to improve the stability and traction of the vehicle by monitoring and modulating its operating parameters and assisting the driver [9,52,53].

Anti-lock braking system (ABS): This system prevents the wheels from locking under heavy braking and ensures that traction is maintained, preventing the vehicle from skidding uncontrollably. The ABS adjusts the brake fluid pressure in real time to prevent the wheels from locking and to improve the driver's control of the vehicle. It integrates speed sensors, electronic control units and hydraulic actuators to efficiently modulate brake pressure to maintain optimum friction between the tyres and the road, ensuring a shorter stopping distance and sustained steerability during emergency braking.

Electronic stability control (ESC): ESC regulates vehicle stability by preventing the vehicle from skidding when cornering sharply or on slippery surfaces. It uses sensors to monitor the condition of the vehicle and uses the data to apply different braking forces to the individual wheels and steer the vehicle in the desired direction. ESC automatically brakes individual wheels to counteract understeer or oversteer, ensuring that the vehicle maintains the trajectory desired by the driver, especially in situations where vehicle stability is at risk. Various manufacturers develop their own versions of ESC that are tailored to their vehicle design.

Active steering system (ASS): Active steering (or steer-by-wire) changes the angle and/or torque of the vehicle's steering wheel to assist the driver in manoeuvreing. The angle of the front wheels is calculated by combining two factors. One aspect is influenced by the driver's input via the steering wheel, while the other is controlled by the steer-by-wire control system. This system subtly modifies the driver's steering movements to prevent skidding, while ensuring that the vehicle maintains the path desired by the driver without noticeable intervention. The driver's tasks can generally be divided into two main areas. Following the path, which is perceived as the main task, and damping disturbances. If the steer-by-wire system is designed to take care of the latter,

the driver can concentrate better on the important task of following the chosen path.

Torque vectoring (TV): If the ESC is activated and the differential brake is applied while the vehicle is accelerating, this can reduce acceleration and not produce the longitudinal response desired by the driver. A viable solution to this problem could be an active torque distribution system for all-wheel drive (AWD), also known as active torque vectoring. The term "all-wheel drive" means that the torque is distributed to all four wheels. The integration of differentials on the front and rear axles and a transfer case [1] enables the AWD function. The differentials on each axle ensure that the left and right wheels turn at different speeds, which is crucial when cornering, as one wheel travels along a path with larger radius and has to turn faster. The active torque distribution system uses the differentials to distribute the required torque individually to each wheel so that the yaw rate can be controlled without having to apply the brakes. This system is becoming increasingly popular in modern electric vehicles, especially those with a wheel-integrated motor structure where the motors can be independently controlled [54,55].

Direct yaw moment control (DYC): A system that uses the longitudinal forces of the tires to control the lateral movement of the vehicle. The term is often used to describe a category of stability control systems that includes both ESC and TV as well as other similar systems [56]. However, car manufacturers use different terms for yaw stability control systems, each of which may have its own characteristics and a slightly different meaning. In this thesis, the terms TV and DYC are used as synonyms.

Traction control system (TCS): The TCS prevents the wheels from spinning during acceleration by modulating the engine power to the wheels. With the help of sensors that monitor the wheel speed, the system reduces excessive power that could cause the wheels to spin and ensures optimum contact and traction on the road. This is particularly important when accelerating on uneven or slippery roads to ensure that the vehicle maintains its stability and stays on the intended path.

Cruise control system (CC): Cruise control is mainly used on open highways and maintains the speed set by the driver without the driver having to constantly press the accelerator pedal.

Adaptive cruise control (ACC): ACC extends the basic functions of a conventional cruise control system. It works like a standard cruise control system when no obstacles are detected in the direction of travel and maintains a constant speed. If a vehicle ahead is detected, especially on highways, ACC switches from a speed maintenance algorithm to a distance control algorithm. Vehicles using ACC must be equipped with sensors such as radar or lidar to detect vehicles ahead. In addition, ACC not only controls the throttle, but also applies the brakes in its distance control mode.

[1] An intermediate gearbox that transfers power from the transmission to the driven axles of four-wheel drive, all-wheel drive and other multi-axle vehicles.

2.6 SUMMARY

The chapter provides a comprehensive overview of the various aspects of vehicle dynamics, control systems and modelling methods. It begins with vehicle dynamics models, explaining the concepts of the two-track and the bicycle model, followed by an introduction of alternative slip formulation. The following sections address the problem of tire force modelling, examining various methods for accurately representing the complex relationships between tire forces, torques and other influencing variables.

Model predictive control (MPC) is examined, outlining its fundamentals, applications and challenges. This section discusses its inherent logic, which mimics human decision-making processes by using mathematical models to predict system behaviour and optimize control inputs based on a defined performance measure. The different versions of MPC, such as linear model predictive control and nonlinear model predictive control, are described in detail, including their formulations, limitations and computational approaches.

Another section introduces the Koopman operator and discusses its theoretical framework and practical applications in transforming nonlinear into linear dynamical systems. The theory revolves around the evolution of measurements and operates in an infinite-dimensional Hilbert space of measurement functions. While the Koopman operator is linear and its spectral decomposition provides revealing insights into the behaviour of nonlinear systems, its practical applicability is limited by its infinite-dimensional nature.

Finally, the chapter provides an insight into vehicle dynamics control systems. Various active control systems such as the anti-lock braking system (ABS), the electronic stability control (ESC) and the torque vectoring system (TV) are presented. In addition, their role in increasing vehicle stability, ensuring traction and avoiding accidents by dynamically modulating the vehicle's operating parameters is emphasised.

3

Data-driven Koopman identification

The third chapter deals with different techniques for data-driven identification of Koopman models. The first two sections introduce the widely recognized extended dynamic mode decomposition and deep dynamic mode decomposition, which are extensively documented in the literature. These concepts are then extended to what is referred to as enhanced extended dynamic mode decomposition, one of the fundamental contributions of this thesis. The rationale behind this approach is elaborated, and three distinct numerical methods for obtaining such models are presented and explained: basis function reduction by discrete selection, multiple step prediction learning algorithm and basis function reduction as a hyperparameter optimization problem. Additionally, some other data-driven Koopman identification approaches are briefly mentioned. The concluding section provides an evaluation of various linear predictors using three established benchmark dynamical systems: the Van der Pol oscillator, the damped Duffing oscillator and the bilinear motor. The simulation results are compared and analyzed, and concluding remarks are made.

3.1 Extended dynamic mode decomposition

In their work [57], the authors introduced extended dynamic mode decomposition (EDMD) as a method that utilizes data to approximate the Koopman operator using a dictionary of basis functions. It is said that these functions span the subspace of observables.

3.1.1 *EDMD for autonomous systems*

For the uncontrolled (autonomous) system (2.61), the approximation is done by solving the optimization problem

$$\min_{A} \sum_{j=1}^{N_K} \left\| \Phi(\mathbf{x}_{t+1}^j) - A\Phi(\mathbf{x}_t^j) \right\|_2^2, \tag{3.1}$$

where $\mathbf{z}_t^j = \Phi(\mathbf{x}_t^j) = \begin{bmatrix} \phi_1(\mathbf{x}_t^j) & \dots & \phi_{n_\psi}(\mathbf{x}_t^j) \end{bmatrix}^T \in \mathbb{R}^{n_\phi}$ is a vector of lifting (basis) functions for the j-th sample. The states $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ are obtained by simulating the nonlinear system model and N_K is the cardinality of the simulated dataset (number of simulated time steps). Important to note is that the states may, but do not have to be a part of a trajectory. It is sufficient to perform N_K different one-step simulations with random samples from the state-space, i.e. from the intervals of interest which are subsets of the state-space, as initial conditions.

The dynamics (2.61) is approximated by a discrete-time linear system with matrix $A \in \mathbb{R}^{n_\phi \times n_\phi}$:

$$\mathbf{z}_{t+1} = A\mathbf{z}_t. \tag{3.2}$$

If the vector of lifting functions is the state vector itself, i.e. $\mathbf{z}_t = \mathbf{x}_t$, the method is called dynamic mode decomposition (DMD) [58].

3.1.2 *EDMD for non-autonomous systems*

For the controlled (non-autonomous) system (2.71), that approximation can be achieved by solving the following optimization problem:

$$\min_{A,B} \sum_{j=1}^{N_K} \left\| \Phi(\mathbf{x}_{t+1}^j) - [A\ B]\, \Psi(\mathbf{x}_t^j, \mathbf{u}_t^j) \right\|_2^2, \quad \Psi(\mathbf{x}_t^j, \mathbf{u}_t^j) = \left[\Phi(\mathbf{x}_t^j)^T\ (\mathbf{u}_t^j)^T \right]^T. \tag{3.3}$$

Here, $\mathbf{z}_t^j = \Phi(\mathbf{x}_t^j) = [\phi_1(\mathbf{x}_t^j) \quad \dots \quad \phi_{n_\phi}(\mathbf{x}_t^j)]^T \in \mathbb{R}^{n_\phi}$ again represents a vector of lifting or basis functions, and $\mathbf{u}_t^j$ denotes an input vector for the jth sample. The states $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ are obtained either by collecting data from the real system or by simulating the nonlinear model of a system. N_K represents the number of simulated time steps or the cardinality of the dataset. The dynamics (2.71) is approximated by a discrete-time linear system with matrices $A \in \mathbb{R}^{n_\phi \times n_\phi}$ and $B \in \mathbb{R}^{n_\phi \times n_u}$:

$$\begin{aligned} \mathbf{z}_{t+1} &= A\mathbf{z}_t + B\mathbf{u}_t, \\ \hat{\mathbf{y}}_t &= C\mathbf{z}_t. \end{aligned} \tag{3.4}$$

The output estimate is $\hat{\mathbf{y}}_t \in \mathbb{R}^{n_y}$, while the output mapping is determined by the matrix $C \in \mathbb{R}^{n_y \times n_\phi}$. The matrix C is obtained by minimizing the least square cost

$$\min_{C} \sum_{j=1}^{N_K} \left\| \mathbf{y}_t^j - C\Phi(\mathbf{x}_t^j) \right\|_2^2, \tag{3.5}$$

This step is the same for both autonomous and non-autonomous systems.

The analytical solution of the problem (3.3) is

$$[A\ B] = V W^T (W W^T)^\dagger, \tag{3.6}$$

where † denotes the Moore-Penrose Pseudoinverse and

$$\begin{aligned} V &= \left[\Phi(\mathbf{x}_{t+1}^1)\ \Phi(\mathbf{x}_{t+1}^2) \dots \Phi(\mathbf{x}_{t+1}^{N_K}) \right] \mathbb{R}^{n_\Phi \times N_K}, \\ W &= \left[\Psi(\mathbf{x}_t^1, \mathbf{u}_t^1)\ \Psi(\mathbf{x}_t^2, \mathbf{u}_t^2) \dots \Psi(\mathbf{x}_t^{N_K}, \mathbf{u}_t^{N_K}) \right] \in \mathbb{R}^{(n_\Phi + n_u) \times N_K}. \end{aligned}$$

In an equivalent way we can obtain matrix C, i.e. solution to (3.5), as

$$C = Y Z^T (Z Z^T)^\dagger, \tag{3.7}$$

where

$$\begin{aligned} Y &= \left[\mathbf{y}_t^1\ \mathbf{y}_t^2 \dots \mathbf{y}_t^{N_K} \right] \in \mathbb{R}^{n_y \times N_K}, \\ Z &= \left[\Phi(\mathbf{x}_t^1)\ \Phi(\mathbf{x}_t^2) \dots \Phi(\mathbf{x}_t^{N_K}) \right] \in \mathbb{R}^{n_\Phi \times N_K}. \end{aligned}$$

The mapping $\Psi(\mathbf{x}_t^j, \mathbf{u}_t^j)$ in the optimization problem (3.3) is structured such that it has a computable solution and a linear predictor (3.4) is obtained [32]. In other words, this requires that the system dynamics from (2.71) has the following form:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + B\mathbf{u}_t. \tag{3.8}$$

This is an important constraint that affects the way in which the state and input space can be defined in order for the method to be applicable.

3.1.3 *EDMD for general nonlinear systems*

In case one would like to use EDMD to model a system where the form (3.8) is not satisfied (e.g. a bilinear system), one possible approach is to rewrite it in the extended space $\tilde{\mathbf{x}} = \begin{bmatrix}\mathbf{x}^T \ \mathbf{u}^T\end{bmatrix}^T$ and treat the control input as a state variable. With this approach, the original discrete-time system

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t),\\ \mathbf{u}_{t+1} &= \mathbf{u}_t + \Delta\mathbf{u}_t,\end{aligned} \tag{3.9}$$

is rewritten as follows

$$\tilde{\mathbf{x}}_{t+1} = \mathbf{f}(\tilde{\mathbf{x}}_t) + B\Delta\mathbf{u}_t, \tag{3.10}$$

where the change of the control input is considered as the new control input. With such an approach, it is possible to lift the autonomous dynamics using the Koopman operator, and the overall system dynamics remain linear for both the control input and the state. It is important to note that, if some of the inputs satisfy requested linear relationship, they don't have to be included in the extended space.

3.1.4 *Basis functions*

Similar to other spectral methods, the precision and convergence speed of the EDMD rely on the chosen set of basis functions, which define the subspace of observables. This set, often referred to as the dictionary, is typically selected by an engineer who is creating the system identification algorithm. In [57] the authors suggest various potential choices basis functions, including polynomials [59], Fourier modes [60], radial basis functions [61], and spectral elements [62].

The best selection of these functions is usually dependent on the specific dynamics of the system under study and the chosen data sampling methodology. Therefore, it is also beneficial to incorporate functions that reflect the inherent dynamics of a system, as Korda et. al. did in [41]. In the context of this thesis, the focus is on polynomials and thin spline radial basis functions, in addition to functions that are intrinsic to vehicle dynamics.

Polynomial basis comprises monomials of the state vector elements, as described by the following equation:

$$P_d = \{\prod_{j=1}^{n_x} x_j^{v_j} \mid v_j \in \mathbb{N} \cup \{0\}, \ \sum_{j=1}^{n_x} v_j \leq d\}, \tag{3.11}$$

where d denotes the order of the basis P_d. They have the capability to interpolate various nonlinear functions, but the their dimensionality increases rapidly as the order increases.

Example 3.1. *Let us consider a state vector* $\mathbf{x} \in \mathbb{R}^2$ *and a polynomial basis of order* $d = 3$. *In this case, the newly generated extended state-space vector is equal to*

$$\Phi(\begin{bmatrix}x_1 & x_2\end{bmatrix}^T) = \begin{bmatrix}x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 & 1\end{bmatrix}^T. \tag{3.12}$$

The order of the basis elements within the vector can be different (here the original states are at the beginning for the sake of simplicity).

Thin spline radial basis functions (RBFs) are powerful tools for multidimensional interpolation, with applications ranging from numerical analysis to machine learning and data-driven modelling. Among the different types of RBFs, thin spline radial basis functions are particularly notable for their smoothing properties and computational efficiency.

Thin spline radial basis functions are defined as functions that depend only on the radial distance from a center point $\mathbf{c}_j$:

$$\phi_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_j\|_2^2 \log \|\mathbf{x} - \mathbf{c}_j\|_2 \tag{3.13}$$

To prevent the squared term from disproportionately influencing the value of the function, it is necessary to normalize the function arguments to fit within a unit hypercube.

Thin spline radial basis functions offer a robust and flexible approach to interpolation problems. Their ability to produce smooth interpolants with minimal curvature makes them a tool of choice in many fields which require the reconstruction of functions from scattered data.

3.2 DEEP DYNAMIC MODE DECOMPOSITION

In this section, the main idea of the deep dynamic mode decomposition (Deep-DMD) algorithm is described. Recently, a lot of similar approaches emerged in the literature, e.g. [63], [64], [65], [66] and [67], but the one described here is inspired by [68].

In the context of EDMD, the selection of basis functions is a manual process undertaken by the algorithm designer. However, in Deep-DMD, this task is automated through the implementation of a deep neural network (DNN). The primary function of the DNN in Deep-DMD is to autonomously determine the subspace of the Koopman operator. This differs from the other approaches where DNN is used for system identification because those methods usually directly model the propagation, whereas Deep-DMD models state vector encoding and propagation is done in a linear fashion. For analytical purposes, the dynamics (2.71) approximated by Deep-DMD can be expressed in the following manner:

$$\begin{aligned} \Phi_e(\mathbf{x}_{t+1}, \theta_e) &= [A\ B]\, \Psi_e(\mathbf{x}_t, \mathbf{u}_t, \theta_e), \\ \hat{\mathbf{y}}_t &= C\Phi_e(\mathbf{x}_t, \theta_e), \end{aligned} \tag{3.14}$$

where $\Psi_e(\mathbf{x}_t, \mathbf{u}_t, \theta_e) = \left[\Phi_e(\mathbf{x}_t, \theta_e)^T\ \mathbf{u}_t^T\right]^T \in \mathbb{R}^{n_{\Phi_e}+n_u}$. The extended state is denoted as $\mathbf{z}_t = \Phi_e(\mathbf{x}_t, \theta_e) \in \mathbb{R}^{n_{\Phi_e}}$, with Φ_e being the encoder neural network, parametrized by θ_e which includes weights and biases.

Figure3.1depicts the Deep-DMD algorithm utilizing an neural network encoder framework. This structure features an encoder, composed of fully-connected neural network layers, responsible for converting the initial state into a higher-dimensional space. The weights A and B are connected to the encoder's final layer and operate without activation functions. These weights are ideally trained concurrently with the encoder. However, if the encoder faces unforeseen errors or a vanishing gradient problem, A and B can be retrained with the frozen encoder network in the final stage of training, potentially reducing the effectiveness of Deep-DMD to that of standard EDMD. Specifically, the output at time t for any hidden layer l can be expressed as:

$$\mathbf{h}_t^l = \sigma^l\left(W^l \mathbf{h}_t^{l-1} + \mathbf{b}^l\right). \tag{3.15}$$

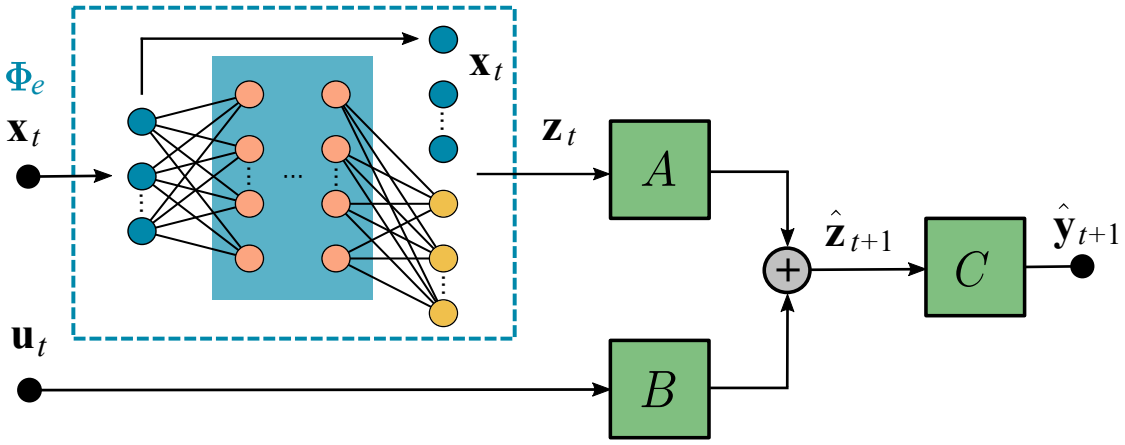

Figure 3.1: Deep-DMD framework diagram for a single step prediction.

In this equation, $W^l \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\mathbf{b}^l \in \mathbb{R}^{n_l}$ represent the weight matrix and the bias vector of the l-th hidden layer, respectively, where n_l denotes the number of neurons in that layer. σ^l is the activation function of the hidden layer l, which is one of the L layers of the encoder, where L denotes the total number of encoder layers. The initial input for the first layer is the state vector itself, i.e. $\mathbf{h}_t^0 = \mathbf{x}_t$. In cases where the lifted state vector also contains the original state vector, the output of the encoder is given as follows:

$$\Phi_e(\mathbf{x}_t, \theta_e) = \left[\mathbf{x}_t^T \ (\mathbf{h}_t^L)^T\right]^T. \tag{3.16}$$

Here, $\mathbf{h}_t^L \in \mathbb{R}^{n_{\Phi_e} - n_x}$ represents the output from the final neural network layer of the encoder.

3.2.1 *Multiple step prediction error minimization*

Instead of minimizing single step prediction errors (3.3) and (3.5), Deep-DMD minimizes the multi step prediction error, which should increase numerical stability of the algorithm and chances of finding a good local optimum (which is a known issues in neural network training). The output prediction in p steps is

$$\hat{\mathbf{y}}_{t+p} = C\hat{\mathbf{z}}_{t+p}, \tag{3.17}$$

where $\hat{\mathbf{z}}_{t+p}$ is the p step approximation starting from $\mathbf{x}_t$:

$$\hat{\mathbf{z}}_{t+p} = A^p \Phi(\mathbf{x}_t) + \sum_{j=1}^{p} A^{j-1} B \mathbf{u}_{t+p-j}. \tag{3.18}$$

The cost function for a given time step t is given as a sum of multiple components, each of which serves a different purpose:

$$L^t = \alpha_1 L^t_{o,x} + \alpha_2 L^t_{x,x} + \alpha_3 L^t_{x,o} + \alpha_4 L^t_{\infty}. \tag{3.19}$$

The first component accounts for the reconstruction error in the original state-space:

$$L^t_{o,x} = \frac{1}{p} \sum_{j=1}^{p} \|\mathbf{y}_{t+j} - C\Phi(\mathbf{x}_t)\|_2^2. \tag{3.20}$$

The second one is the mean of prediction errors along the p time steps:

$$L^t_{x,x} = \frac{1}{p}\sum_{j=1}^{p} \|\mathbf{y}_{t+j} - C\hat{\mathbf{z}}_{t+j}\|_2^2. \tag{3.21}$$

To ensure a low prediction error in the lifted state-space, the following cost function is minimized:

$$L^t_{x,o} = \frac{1}{p}\sum_{j=1}^{p} \|\Phi(\mathbf{x}_{t+j}) - \hat{\mathbf{z}}_{t+j}\|_2^2. \tag{3.22}$$

To remove the high amplitude outliers, l^∞ norm is used, i.e.

$$L^t_{\infty} = \frac{1}{p}\sum_{j=1}^{p} \|\mathbf{y}_{t+j} - C\Phi(\mathbf{x}_t)\|_\infty + \frac{1}{p}\sum_{j=1}^{p} \|\mathbf{y}_{t+j} - C\hat{\mathbf{z}}_{t+j}\|_\infty. \tag{3.23}$$

Finally, to avoid overfitting, regularization cost is added:

$$L_{reg} = \|G\|_2^2 + \|h\|_2^2 + \|A\|_2^2 + \|B\|_2^2 + \|C\|_2^2. \tag{3.24}$$

The final cost function, averaged over N_K training samples equals

$$L = \frac{1}{N_K}\sum_{t=1}^{N_K} L^t + \alpha_5 L_{reg}, \tag{3.25}$$

where scalar weights $\alpha_1, \ldots, \alpha_5$ determine relative influence of a given component on the complete cost. Weights are chosen depending on a specific application, and some of them can be set to zero if appropriate.

3.2.2 *Learning algorithm*

Learning is performed in batch mode. The data is split into training and validation sets, and early stopping is implemented as a regularization technique to prevent overfitting (together with the regularization described in (3.24)). The implementation is described in Algorithm1. In it, e_{max} is the maximum number of training epochs, v_p is the violation patience, i.e. the number of epochs for which the algorithm "tolerates" that the validation loss does not decrease, n_e is the evaluation epoch number, b_s is the batch size and l_r the learning rate.

EDMD model retraining can be done after the learning process has been completed. The main idea is to perform the original EDMD least square minimization (3.6) and (3.7), but this time the previously learned deep encoder (3.16) is used as the basis function. The reason for this are possible numerical problems that can occur when executing Algorithm1.

3.3 ENHANCED EXTENDED DYNAMIC MODE DECOMPOSITION

The equations (3.3) and (3.5) demonstrate how the EDMD approximation can be computed easily and quickly. However, the method suffers from the curse of dimensionality, which can cause even larger numerical errors due to the enormous size of the vectors $\Phi(\mathbf{x}_t)$ and $\Psi(\mathbf{x}_t, \mathbf{u}_t)$ [69]. Additionally, it only minimizes a single step prediction error, which can become problematic as the numerical errors multiply with increasing prediction horizon. On the other hand, Deep-DMD,

Algorithm 1: Deep-DMD learning procedure

Data: X_{train}, U_{train}, Y_{train}, X_{valid}, U_{valid}, Y_{valid}
Init: A, B, C, θ_e, e_{max}, $\alpha_1 \dots \alpha_5$, p, v_p, n_e, b_s, l_r
Result: $A_{best}, B_{best}, C_{best}, \theta_{e,best}$

$train\_data \leftarrow \{X_{train}, U_{train}, Y_{train}\}$;
$validation\_data \leftarrow \{X_{valid}, U_{valid}, Y_{valid}\}$;
$model \leftarrow \{A, B, C, \theta_e\}$;
$v \leftarrow 0$;
$model\_best \leftarrow model$;
$L_{best} \leftarrow \infty$;
for $e = 0 : e_{max}$ **do**
  if $e \mod n_e == 0$ **then**
    $L_{valid} \leftarrow loss(validation\_data, p)$;
    if $L_{valid} < L_{best}$ **then**
      $L_{best} \leftarrow L_{valid}$;
      $model\_best \leftarrow model$;
      $v \leftarrow 0$;
    else
      $v \leftarrow v + 1$;
    end
    if $v == v_p$ **then**
      break;
    end
  end
  $batch\_data = create\_batches(train\_data, b_s)$;
  for $b = 0 : b_s$ **do**
    $L_{batch} \leftarrow loss(batch\_data[b], p)$;
    $model.backward(L_{batch})$; `//calculate gradients`
    $model.update(optimizer(l_r))$; `//update parameters`
  end
end

since it contains a neural network based encoder, has a high chance of experiencing convergence problems if all the hyperparameters are not chosen very carefully.

Here a method called enhanced extended dynamic mode decomposition (E²DMD) , which has the potential to handle all of these problems, is presented. The approximated model resulting from E²DMD can be written as:

$$\begin{aligned} \mathbf{w}_{t+1} &= A\mathbf{w}_t + B\mathbf{u}_t \\ \hat{\mathbf{y}}_t &= C\mathbf{w}_t, \end{aligned} \tag{3.26}$$

where $\mathbf{w}_t = \tilde{\Phi}(\mathbf{x}_t)$ is the reduced dimension encoding of the basis function vector $\mathbf{z}_t$, such as the

one used in (3.4). System matrices are, as for EDMD, obtained by minimizing the cost function

$$\min_{A,B} \sum_{j=1}^{N_K} \|\tilde{\Phi}(\mathbf{x}_{t+1}^j) - [A\ B]\,\tilde{\Psi}(\mathbf{x}_t^j, \mathbf{u}_t^j)\|_2^2, \quad \tilde{\Psi}(\mathbf{x}_t^j, \mathbf{u}_t^j) = \left[\tilde{\Phi}(\mathbf{x}_t^j)^T\ (\mathbf{u}_t^j)^T\right]^T, \tag{3.27}$$

and output matrix by minimizing

$$\min_{C} \sum_{j=1}^{N_K} \|\mathbf{y}_t^j - C\tilde{\Phi}(\mathbf{x}_t^j)\|_2^2. \tag{3.28}$$

The E²DMD algorithm was first proposed in [70], and its more detailed description is given in the following text.

3.3.1 *Basis function dimension reduction*

E²DMD reduces the dimension of the originally set basis function vector $\mathbf{z}_t \in \mathbb{R}^{n_\phi}$ to an arbitrary dimension n_w such that $n_x < n_w < n_\phi$:

$$\mathbf{w}_t = G\mathbf{z}_t + \mathbf{h}, \tag{3.29}$$

resulting in a new basis function vector $\mathbf{w}_t \in \mathbb{R}^{n_w}$, thus avoiding the curse of dimensionality. Mapping (3.26), uses the optimal matrix $G \in \mathbb{R}^{n_w \times n_\Phi}$ and the vector $\mathbf{h} \in \mathbb{R}^{n_w}$ to extract as much useful information as possible from the vector $\mathbf{z}_t$.

The new basis vector can also directly contain the original state vector, i.e.

$$\mathbf{w}_t = \left[\mathbf{x}_t^T\ \tilde{\mathbf{w}}^T\right]^T, \tag{3.30}$$

where in this case the reduced basis vector $\tilde{\mathbf{w}}_t \in \mathbb{R}^{n_w - n_x}$ is obtained from

$$\tilde{\mathbf{w}}_t = G_w\mathbf{z}_t + \mathbf{h}_w, \tag{3.31}$$

with the optimal matrix $G_w \in \mathbb{R}^{(n_w - n_x)\times n_\Phi}$ and the vector $\mathbf{h} \in \mathbb{R}^{n_w - n_x}$.

The diagram of the method is shown in Figure3.2, while the numerical matrix calculation is described in the following lemma.

Lemma 3.1. *Let us define matrices $V \in \mathbb{R}^{(n_\Phi + n_u)\times N_K}$, $W \in \mathbb{R}^{n_\Phi \times N_K}$, $Z \in \mathbb{R}^{n_\Phi \times N_K}$ and $Y \in \mathbb{R}^{n_y \times N_K}$ as in (3.6) and (3.7), reduction matrix $G \in \mathbb{R}^{n_w \times n_\Phi}$ and bias vector $\mathbf{h} \in \mathbb{R}^{n_w}$. In addition, lets define the bias matrix $H = [\mathbf{h}\,\mathbf{h}\ldots\mathbf{h}] \in \mathbb{R}^{n_w \times N_K}$, the block diagonal matrix $\mathbf{G} = diag(G,\ I_u)$ consisting of the matrix G and the identity matrix $I_u \in \mathbb{R}^{n_u \times n_u}$, and the bias matrix extended with zeros $\mathbf{H} = \left[H^T\ \mathbf{0}^T\right]^T \in \mathbb{R}^{(n_w + n_u)\times N_K}$. The analytical solution of the problem (3.27) can then be written as*

$$[A\ B] = (GV + H)(W^T\mathbf{G}^T + \mathbf{H}^T)((\mathbf{G}W + \mathbf{H})(W^T\mathbf{G}^T + \mathbf{H}^T))^\dagger,$$

Similarly, the solution of (3.28) is

$$C = Y(Z^TG^T + H^T)((GZ + H)(Z^TG^T + H^T))^\dagger.$$

Proof. *The matrices V, W, Y and Z are defined for the basis $\Phi(\mathbf{x})$ as*

$$\begin{aligned} V &= \left[\Phi(\mathbf{x}_{t+1}^1)\ \Phi(\mathbf{x}_{t+1}^2)\dots\Phi(\mathbf{x}_{t+1}^{N_K})\right], \\ W &= \left[\Psi(\mathbf{x}_t^1,\mathbf{u}_t^1)\ \Psi(\mathbf{x}_t^2,\mathbf{u}_t^2)\dots\Psi(\mathbf{x}_t^{N_K},\mathbf{u}_t^{N_K})\right], \\ Y &= \left[\mathbf{y}_t^1\ \mathbf{y}_t^2\dots\mathbf{y}_t^{N_K}\right], \\ Z &= \left[\Phi(\mathbf{x}_t^1)\ \Phi(\mathbf{x}_t^2)\dots\Phi(\mathbf{x}_t^{N_K})\right], \end{aligned} \tag{3.32}$$

and for such formulation, the system matrices A_o, B_o and C_o can be found using:

$$\begin{aligned} [A_o\ B_o] &= VW^T(WW^T)^\dagger, \\ C_o &= YZ^T(ZZ^T)^\dagger. \end{aligned} \tag{3.33}$$

These matrices represent the solution of the problems (3.3) and (3.5), i.e. the approximation of the Koopman operator with EDMD (without dimension reduction). Let us define transformed matrices $\tilde{V}$, $\tilde{W}$ and $\tilde{Z}$, which represents the system with a reduced basis $\tilde{\Phi}(\mathbf{x})$:

$$\begin{aligned} \tilde{V} &= \left[\tilde{\Phi}(\mathbf{x}_{t+1}^1)\ \tilde{\Phi}(\mathbf{x}_{t+1}^2)\dots\tilde{\Phi}(\mathbf{x}_{t+1}^{N_K})\right], \\ \tilde{W} &= \left[\tilde{\Psi}(\mathbf{x}_t^1,\mathbf{u}_t^1)\ \tilde{\Psi}(\mathbf{x}_t^2,\mathbf{u}_t^2)\dots\tilde{\Psi}(\mathbf{x}_t^{N_K},\mathbf{u}_t^{N_K})\right], \\ \tilde{Z} &= \left[\tilde{\Phi}(\mathbf{x}_t^1)\ \tilde{\Phi}(\mathbf{x}_t^2)\dots\tilde{\Phi}(\mathbf{x}_t^{N_K})\right]. \end{aligned} \tag{3.34}$$

For the reduced system, the matrices A, B and C are calculated as:

$$\begin{aligned} [A\ B] &= \tilde{V}\tilde{W}^T(\tilde{W}\tilde{W}^T)^\dagger, \\ C &= Y\tilde{Z}^T(\tilde{Z}\tilde{Z}^T)^\dagger. \end{aligned} \tag{3.35}$$

The basis reduction transformation (3.29) is equivalent to

$$\tilde{\Phi}(\mathbf{x}_t) = G\Phi(\mathbf{x}_t) + \mathbf{h}. \tag{3.36}$$

By substituting vectors $\tilde{\Phi}(\mathbf{x})$ in (3.34) by (3.36) and defining $\hat{\mathbf{h}} = \left[\mathbf{h}^T\ \mathbf{0}^T\right]^T$, we get

$$\begin{aligned} \tilde{V} &= \left[G\Phi(\mathbf{x}_{t+1}^1) + \mathbf{h},\ G\Phi(\mathbf{x}_{t+1}^2) + \mathbf{h},\dots G\Phi(\mathbf{x}_{t+1}^{N_K}) + \mathbf{h}\right], \\ \tilde{W} &= \left[\mathbf{G}\Psi(\mathbf{x}_t^1,\mathbf{u}_t^1) + \hat{\mathbf{h}},\ \mathbf{G}\Psi(\mathbf{x}_t^2,\mathbf{u}_t^2) + \hat{\mathbf{h}}\dots\mathbf{G}\Psi(\mathbf{x}_t^{N_K},\mathbf{u}_t^{N_K}) + \hat{\mathbf{h}}\right], \\ \tilde{Z} &= \left[G\Phi(\mathbf{x}_t^1) + \mathbf{h},\ G\Phi(\mathbf{x}_t^2) + \mathbf{h},\dots G\Phi(\mathbf{x}_t^{N_K}) + \mathbf{h}\right]. \end{aligned} \tag{3.37}$$

Since the reduction transformation is affine, (3.37) can be written as

$$\begin{aligned} \tilde{V} &= G\left[\Phi(\mathbf{x}_{t+1}^1),\ \Phi(\mathbf{x}_{t+1}^2),\dots\Phi(\mathbf{x}_{t+1}^{N_K})\right] + \left[\mathbf{h}\,\mathbf{h}\dots\mathbf{h}\right], \\ \tilde{W} &= \mathbf{G}\left[\Psi(\mathbf{x}_t^1,\mathbf{u}_t^1)\ \Psi(\mathbf{x}_t^2,\mathbf{u}_t^2)\dots\Psi(\mathbf{x}_t^{N_K},\mathbf{u}_t^{N_K})\right] + \left[\hat{\mathbf{h}}\,\hat{\mathbf{h}}\dots\hat{\mathbf{h}}\right], \\ \tilde{Z} &= G\left[\Phi(\mathbf{x}_t^1)\ \Phi(\mathbf{x}_t^2)\dots\Phi(\mathbf{x}_t^{N_K})\right] + \left[\mathbf{h}\,\mathbf{h}\dots\mathbf{h}\right], \end{aligned} \tag{3.38}$$

which, after combining the initial definitions with (3.32), becomes:

$$\tilde{V} = GV + H, \quad \tilde{W} = \mathbf{G}W + \mathbf{H}, \quad \tilde{Z} = GZ + H. \tag{3.39}$$

By substituting (3.39) into (3.35), we get

$$\begin{aligned} [A\ B] &= (GV + H)(W^T\mathbf{G}^T + \mathbf{H}^T)((\mathbf{G}W + \mathbf{H})(W^T\mathbf{G}^T + \mathbf{H}^T))^\dagger, \\ C &= Y(Z^TG^T + H^T)((GZ + H)(Z^TG^T + H^T))^\dagger, \end{aligned} \tag{3.40}$$

which proves the lemma.

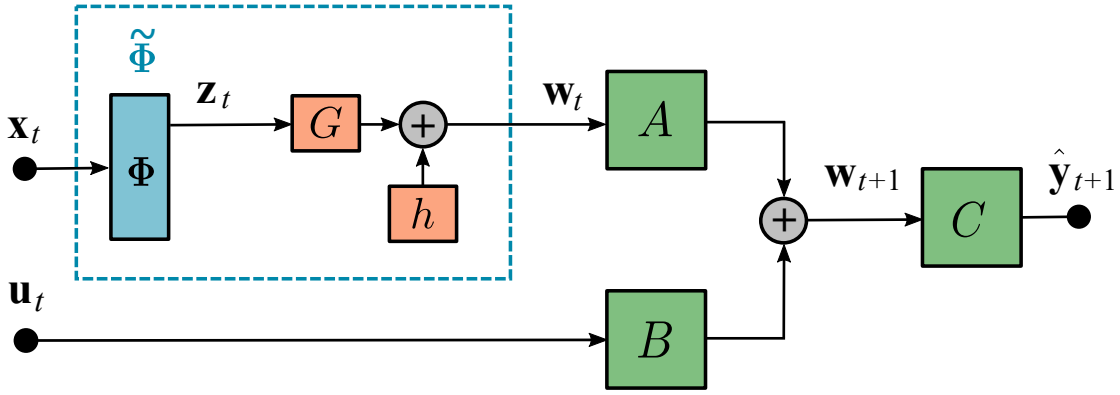

Figure 3.2: E²DMD framework diagram for a single step prediction.

Lemma3.1shows how the procedure for calculating Koopman model matrices for E ²DMD method looks like. However, the problem is obviously nonlinear since the optimization variables G and $\mathbf{h}$ cannot be expressed explicitly. In the rest of the section, three different optimization methods are proposed to solve this problem.

3.3.2 *Basis function reduction by discrete selection*

One of the ways to train E²DMD is to omit the bias vector $\mathbf{h}$ and constrain the reduction matrix G so that a binary selection matrix is created, which is referred to as G_b. The term binary selection matrix describes a matrix that consists exclusively of zeros and ones and is used to select certain elements from a vector. When multiplied by a vector, this matrix produces a new vector whose elements are a subset of the original vector. In addition, all its rows are unique, i.e. each row selects a different element from the original vector. The equation (3.29) then becomes

$$\mathbf{w}_t = G_b \mathbf{z}_t, \tag{3.41}$$

while the solution to the problems (3.27) and (3.28) is given by the Corollary3.1. In addition, Example3.2illustrates the form which the matrix G_b should have.

Corollary 3.1. *Let us define matrices $V \in \mathbb{R}^{(n_\Phi + n_u)\times N_K}$, $W \in \mathbb{R}^{n_\Phi \times N_K}$, $Z \in \mathbb{R}^{n_\Phi \times N_K}$ and $Y \in \mathbb{R}^{n_y \times N_K}$ as in (3.6) and (3.7) and the reduction matrix $G_b \in \mathbb{R}^{n_w \times n_\Phi}$. In addition lets define the block diagonal matrix $\mathbf{G}_b = diag(G_b, I_u)$ consisting of the matrix G_b and the identity matrix $I_u \in \mathbb{R}^{n_u \times n_u}$. The analytical solution of the problem (3.27) can then be written as*

$$[A\ B] = G_b V W^T \mathbf{G}_b^T (\mathbf{G}_b W W^T \mathbf{G}_b^T)^\dagger.$$

Equivalently, the solution of (3.28) is

$$C = Y Z^T G_b^T (G_b Z Z^T G_b^T)^\dagger.$$

Proof. *Consider the equations proved in Lemma3.1. Substituting the matrix G with the binary selection matrix G_b and setting the bias vector $h = \mathbf{0}$ directly proves the corollary.*

Example 3.2. *Let us define original basis vector* $\mathbf{z} = \begin{bmatrix} z_1 \; z_2 \; z_3 \; z_4 \; z_5 \end{bmatrix}^T$ *and reduced basis vector* $\mathbf{w} = \begin{bmatrix} w_1 \; w_2 \; w_3 \end{bmatrix}^T = \begin{bmatrix} z_2 \; z_5 \; z_1 \end{bmatrix}^T$. *The corresponding binary selection matrix equals*

$$G_b = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

To find the optimal selection matrix G_b, one can choose any discrete optimization algorithm, but here the iterated local search with simulated annealing was applied. The main idea behind the algorithm is to initialize the matrix G_b as a unit matrix concatenated with a zero matrix and then randomly swap its columns to find the best n_w-dimensional choice of all n_Φ basis vector components. Additionally, Metropolis heuristics, the reheating trick and random reordering of the matrix columns were applied to increase the chances of finding a good solution [71]. A good solution is the one that has the lowest mean multiple step prediction error evaluated on the validation set. This error is described as

$$L = \frac{100}{N_{valid} p} \sum_{j=1}^{N_{valid}} \sum_{k=0}^{p-1} \frac{\|\hat{\mathbf{y}}_k^j - \mathbf{y}_k^j\|_2}{\|\mathbf{y}_k^j\|_2}, \tag{3.42}$$

where $\mathbf{y}_k^j$ is the real and $\hat{\mathbf{y}}_k^j$ predicted output value for j-th trajectory at step k, N_{valid} is the number of samples in the validation set and p the number of prediction steps. The following text gives a brief introduction to the concepts described, while the pseudocode for the optimization is given in Algorithm2. Within this algorithm, the number of training epochs is represented as e_{max}, the violation patience as v_p, the evaluation epoch number as n_e, the initial temperature as T_{init}, and the temperature reduction coefficient as α_{temp}.

Iterated local search (ILS) is a metaheuristic algorithm used to solve optimization problems, especially those where the search for an optimal solution is complex or computationally expensive. The main idea behind ILS is the repeated application of a local search algorithm from different starting points in the search space. It is particularly effective because it combines the depth of local search (thoroughly exploiting the current area of the search space) with the breadth of global search (exploring new and potentially more promising areas of the search space through perturbation). This makes it a powerful tool for problems where the solution landscape is rugged with many local optima, such as planning, routing, or combinatorial optimization tasks. A more detailed explanation can be found in [72].

Simulated annealing (SA) is an optimization technique based on the annealing process in metallurgy. In annealing, a material is heated and then slowly cooled to reduce defects and increase its strength. Simulated annealing applies this concept to find an approximate solution to an optimization problem, which is particularly useful for finding a global optimum in a large search space with many local optima. It is especially powerful for problems where the solution space is large, such as scheduling, routing or optimising complex functions. Its ability to avoid local optima makes it a preferred choice in many real-world applications, although it is not guaranteed to find the absolute best global solution. The theory explaining the algorithm and some examples are given in [71].

Algorithm 2: Simulated annealing optimization of a binary selection matrix

Data: Z_{train}, U_{train}, Y_{train}, Z_{valid}, U_{valid}, Y_{valid}
Init: $e_{max}, v_p, n_e, T_{init}, \alpha_{temp}, p$
Result: $A_{glob\_best}, B_{glob\_best}, C_{glob\_best}, G_{b,glob\_best}$

$train\_data \leftarrow \{Z_{train}, U_{train}, Y_{train}\}$;
$validation\_data \leftarrow \{Z_{valid}, U_{valid}, Y_{valid}\}$;
$G_b \leftarrow [I,\ 0]$; `//identity matrix concatenated with zeros`
$G_{b,best},\ L_{best} \leftarrow G_b,\ \infty$;
$G_{b,glob\_best},\ L_{glob\_best} \leftarrow G_{b,best},\ L_{best}$;
$v \leftarrow 0$;
$T \leftarrow T_{init}$;
for $e = 0 : e_{max}$ **do**
    if $e \mod n_e == 0$ **then**
        $G_b \leftarrow randomize\_all\_columns(G_b)$;
    else
        $G_b \leftarrow randomly\_swap\_two\_columns(G_b)$; `//local search`
    end
    $A, B, C \leftarrow calculate\_model(train\_data, G_b)$;
    $L_{valid} \leftarrow loss(validation\_data, A, B, C, G_b, p)$;
    if $L_{valid} < L_{best}$ **then**
        $G_{b,best},\ L_{best} \leftarrow G_b,\ L_{valid}$;
    else
        `//Metropolis heuristics`
        if $rand\_number < exp(-(L_{valid} - L_{best})/T)$ **then**
            $G_{b,best},\ L_{best} \leftarrow G_b,\ L_{valid}$;
        end
    end
    if $L_{best} < L_{glob\_best}$ **then**
        $G_{b,glob\_best},\ L_{glob\_best} \leftarrow G_{b,best},\ L_{best}$;
        $v \leftarrow 0$;
    else
        $v \leftarrow v + 1$;
    end
    if $v < v_p$ **then**
        $T \leftarrow \alpha_{temp} T$; `//reduce the temperature`
    else
        $T \leftarrow T_{init}$; `//reinitialize the temperature`
    end
end
$A_{glob\_best}, B_{glob\_best}, C_{glob\_best} \leftarrow calculate\_model(train\_data, G_{b,glob\_best})$;

3.3.3 *Multiple step prediction learning algorithm*

The second approach to train E²DMD proposed in the thesis is to use multiple step prediction error minimization as in3.2.1. The procedure is the same as for Deep-DMD, only the structure of the encoder changes. The learning procedure is specified in Algorithm3.

Algorithm 3: E²DMD multiple step minimization learning procedure

Data: $X_{train}, U_{train}, Y_{train}, X_{valid}, U_{valid}, Y_{valid}$
Init: $A, B, C, G, h, e_{max}, \alpha_1 \dots \alpha_5, p, v_p, n_e, b_s, l_r$
Result: $A_{best}, B_{best}, C_{best}, G_{best}, h_{best}$

$train\_data \leftarrow \{X_{train}, U_{train}, Y_{train}\}$;
$validation\_data \leftarrow \{X_{valid}, U_{valid}, Y_{valid}\}$;
$model \leftarrow \{A, B, C, G, h\}$;
$v \leftarrow 0$;
$model\_best \leftarrow model$;
$L_{best} \leftarrow \infty$;
for $e = 0 : e_{max}$ **do**
    `//gradient descend and early stopping logic is the same as in Algorithm1`
end

Although this method is implemented similarly to Deep-DMD, it does not require any form of neural network, making it easier and faster to train. As with Deep-DMD, EDMD model retraining can be performed after running the Algorithm3, if required.

3.3.4 *Basis function reduction as a hyperparameter optimization problem*

In the previous text, when solving the E²DMD problem, all matrices were considered to be a solution of the same optimization problem. Alternative point of view is to consider the matrix G and the vector $\mathbf{h}$ from the affine transformation (3.29) as hyperparameters of the underlying EDMD optimization problem.

Hyperparameters are the parameters of the algorithm that are not learned from the data, but set prior to the training process. In the case of neural networks, these are the learning rate, the number of hidden layers and neurons in a neural network, regularization terms and many others. In contrast to the model parameters, which are learned during training, the hyperparameters are (traditionally) set by the engineer and can significantly influence the performance of the model. In hyperparameter optimization, numerical algorithms are used to automatically find the best (or more often good) hyperparameter combination, which is crucial in machine learning to improve the performance of the models. It also significantly reduces the time required for model tuning. The flowchart of a typical hyperparameter optimization algorithm is depicted in Figure3.3.

Various methods can be used for this purpose, including grid and random search, Bayesian optimization, genetic algorithms, tree-structured Parzen estimator (TPE) [73], covariance matrix adaptation evolution strategy (CMA-ES) [74] and others.

Tree-structured Parzen estimator is a Bayesian optimization approach which models the probability distribution of hyperparameters as a function of the results. It divides the search space

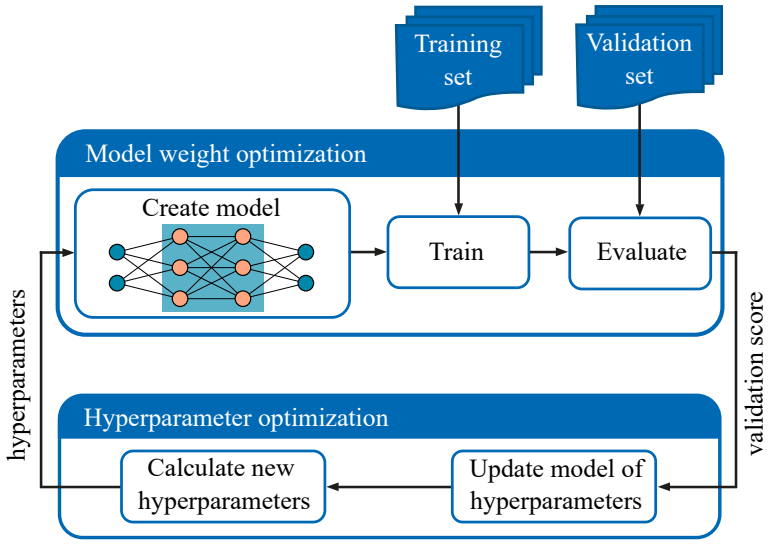

Figure 3.3: The flowchart for hyperparameter optimization.

into regions and uses Parzen window estimators to approximate the distribution within these regions. The process is iterative and focuses more on promising regions over time. It is more efficient compared to random search, adjusts the search based on previous results, improves over iterations, and is suitable for high-dimensional spaces. However, it is more complex to implement than simpler methods such as grid or random search, the initial definition of the search space can affect efficiency and it is computationally intensive. For more detailed explanation, please refer to [73] and the references therein.

In this thesis, hyperparameter optimization is viewed as an outer loop in the learning process, where the basis reduction is performed in this outer loop (selection of hyperparameters G and $\mathbf{h}$), while the EDMD problem is solved in the inner loop. The algorithm minimizes the mean multiple step prediction error (3.42) and uses the Optuna framework [75] and its implementation of the TPE method [76,77] with a maximum of n_t trials. [1]

3.4 OTHER APPROACHES

The existing body of literature includes a variety of methods for numerically determining the Koopman operator model, with some notable examples highlighted here.

One such technique is Generalized Laplacian Analysis (GLA), as discussed in [78]. GLA enables the calculation of Koopman modes for a function vector, provided the Koopman eigenvalues are known, by utilizing time series data. However, the functions must be within the span of the eigenfunctions for GLA to be effective. Since GLA imposes significant prerequisites and has its limitations, it is more suited as an analytical tool than a computational one, as the authors note in [27]. A study done by Korda and Mezić in [79] offers a more practical approach, constructing Koopman operator eigenfunctions directly from data, assuming known eigenvalues. This method is optimization-based and does not require dictionary selection. Another approach, Data-Driven Encoding (DDE) of the Koopman operator, calculates inner products from data of nonlinear

[1] Although the simulated annealing approach in Algorithm2could also be considered as hyperparameter optimization, it is this change of viewpoint that allows the optimization problems to be separated and allows existing hyperparameter optimization frameworks to be used for this task.

dynamic systems. An efficient algorithm for computing these inner products is presented in [80]. In [81], the authors introduce Koopmanizing flows, a novel continuous-time framework for learning linear predictors in nonlinear dynamics. This method involves a latent linear system, related diffeomorphically, unfolding into a linear predictor through a monomial basis. It simultaneously learns the lifting, linear dynamics, and state reconstruction, with an unconstrained Hurwitz matrices parameterization ensuring asymptotic stability regardless of operator approximation accuracy. The Freeman method, as described by Cibulka et al. in [82], presents a dictionary-free Koopman model with a nonlinear input transformation. This approach differs from others by eliminating the need for a dictionary and incorporating a nonlinear input transformation, enhancing prediction accuracy with minimal ad hoc adjustments. It also facilitates input quantization and exploits symmetries to reduce computational costs, both offline and online.

Though these methods are valuable, they are beyond the scope of this thesis and are mentioned merely for reference.

3.5 SIMULATION RESULTS

In this section, the comparison of different linearization-based approaches is presented:

1. classical EDMD-based Koopman model,
2. Deep-DMD-based model,
3. discrete selection based E ^{2}DMD (described in 3.3.2, referred to as E ^{2}DMD-DS),
4. multiple step prediction minimizing E^{2}DMD (described in 3.3.3, referred to as E ^{2}DMD-MS),
5. E^{2}DMD trained with hyperparameter optimization approach (described in 3.3.4, refered to as E^{2}DMD-HO),
6. model based on local linearization of the dynamics at the origin (refered to as LIN 0),
7. model based on local linearization of the dynamics at a given initial condition $\mathbf{x}_0$ (refered to as LIN $\mathbf{x}_0$).

The approaches are evaluated on three dynamical systems that often serve as benchmarks for such problems, similarly as was done in [32] and [79]. These systems and data collection methods are as follows:

- **Van der Pol oscillator** with dynamics given by

$$\begin{aligned} \dot{x}_1 &= 2x_2, \\ \dot{x}_2 &= -0.8x_1 + 2x_2 - 10x_1^2 x_2 - u. \end{aligned} \tag{3.43}$$

  In order to collect the data required for model identification, the system dynamics are sampled using the Runge-Kutta method of order four with a discretization interval of $T_s = 0.01$ seconds. This process involves simulating 1000 different trajectories, each over a span of 200 sampling intervals, which is equal to 2 seconds per trajectory. For each trajectory, the control input is a randomly generated signal that is uniformly distributed in the unit interval $[-1, 1]$. The starting points of these trajectories are also determined randomly, with their initial conditions uniformly distributed within the unit box, i.e. $[-1, 1]^2$.

- **Damped Duffing oscillator** described by

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -0.5x_2 - x_1(4x_1^2 - 1) + 0.5u. \end{aligned} \tag{3.44}$$

  The procedure for data collection is done using the same method and parameters as for Van der Pol oscillator.

- **Bilinear motor** characterized by the following equations:

$$\begin{aligned} \dot{x}_1 &= -\frac{R_a}{L_a}x_1 - \frac{k_m}{L_a}x_2 u - \frac{u_a}{L_a}, \\ \dot{x}_2 &= -\frac{B}{J}x_2 - \frac{k_m}{J}x_1 u - \frac{\tau_l}{J}. \end{aligned} \tag{3.45}$$

  In the above equation, x_1 represents the rotor current, x_2 is the angular velocity and u is the stator current, which is the control input. The parameters of the system are as follows $L_a = 0.314$, $R_a = 12.345$, $k_m = 0.253$, $J = 0.00441$, $B = 0.00732$, $\tau_l = 1.47$, $u_a = 60$. This model is particularly characterised by its bilinear nature, which connects the state and the control input. The control input u is physically limited to the range $u \in [-4, 4]$, but is rescaled to $[-1, 1]$ for the analysis.

  To derive the Koopman operator, the scaled dynamics of the system is discretized using the Runge-Kutta method of order four with a discretization interval $T_s = 0.01$ seconds. This involves simulating 1000 trajectories, each consisting of 200 sampling steps, also equivalent to 2 seconds per trajectory. The control input for each trajectory is a randomly generated signal, uniformly distributed within the interval $[-1, 1]$. The initial conditions for these trajectories are also randomly set, uniformly distributed within the $[-1, 1]^2$ unit box.

In the following text, the procedures for lifting system dynamics and learning different linear predictors are given, together with simulation experiment description and detailed statistics.

3.5.1 *Learning algorithm setup*

As mentioned earlier, a total of $N_s = 200000$ samples were generated for each system. This dataset was subsequently split into training and validation sets, with the training set consisting of 85% of the total data, which equates to $N_{train} = 170000$ samples, and the validation set containing $N_{valid} = 30000$ samples. Moreover, each method employed in this section requires an additional set of parameters, outlined as follows:

- **EDMD**: trained exclusively using the training data (no consideration for validation data as it is not part of the standard algorithm),

- **Deep-DMD**: encoder with a total of $L = 5$ layers, where $n_1 = 32$, $n_2 = 64$, $n_3 = 128$, $n_4 = 64$ and $n_5 = n_{\Phi_e} - n_x$; activation function for the layers n_1 to n_4 is the rectified linear unit (ReLU) and for n_5 the hyperbolic tangent; weights $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = 0.3$, $\alpha_4 = 10^{-9}$ and $\alpha_5 = 10^{-9}$, maximum number of training epochs $e_{max} = 10000$, number of prediction steps $p = 25$, violation patience $v_p = 5$, number of evaluation epochs $n_e = 5$, batch size $b_s = 64$ and learning rate $l_r = 10^{-4}$; EDMD retraining was used; state included in the lifted state vector;

- **E²DMD-DS**: number of training epochs $e_{max} = 10000$, violation patience $v_p = 100$, randomization epoch number $n_e = 100$, initial temperature $T_{init} = 1000$, temperature reduction coefficient $\alpha_{temp} = 0.99$, and a number of prediction steps $p = 200$; state included in the lifted state vector;
- **E²DMD-MS**: employs the same parameters as Deep-DMD (those that are applicable); EDMD retraining was used; state included in the lifted state vector;
- **E²DMD-HO**: number of trials $n_t = 250$ and prediction horizon $p = 200$, encoder weigth and bias values limited to $[-1,\ 1]$; state included in the lifted state vector.

3.5.2 *Van der Pol oscillator simulation*

In the study of the Van der Pol oscillator, two distinct sets of basis functions were employed:

1. Thin plate spline radial basis functions (RBFs) with centers selected via a uniform distribution in the unit box. The total number of different RBFs is 100, with the state also included in the basis function vector ($\phi_1 = x_1$, $\phi_2 = x_2$). This leads to a lifted state- space dimension of $n_\Phi = 102$.

2. Polynomial basis of order $d = 15$, resulting in a lifted state-space dimension of $n_\Phi = 136$.

The following experiments compare all linearization approaches mentioned at the beginning of the section, for both RBF and polynomial basis functions. The initial conditions and the control input for the simulation are random and uniformly distributed within the $[-0.7, 0.7]^2$ box and $[-1, 1]$ unit interval, respectively. The sample time equals $T_s = 0.01$ s, while the simulation lasts for the total of $T_{sim} = 3$ seconds. The quality of the predictors is evaluated using mean normalized prediction error (MNPE):

$$MNPE = \frac{100}{N_{sim}} \sum_{k=0}^{N_{sim}-1} \frac{\|\hat{\mathbf{y}}_k - \mathbf{y}_k\|_2}{\|\mathbf{y}_k\|_2}, \tag{3.46}$$

with $\hat{\mathbf{y}}_k$ being the predicted and $\mathbf{y}_k$ real output value at time step k. Output mapping is an identity matrix, i.e. $\mathbf{y} = [x_1\ x_2]^T$.

MNPE was computed by conducting 5000 simulation runs, and the following tables present its mean, median, minimum, and maximum values. They serve to better illustrate the error distribution, but for predictor performance evaluation only mean and median will be used, since extreme values can occur due to unusually "hard" trajectory or numerical error.

Table3.1provides a comparison of various linear predictors. Firstly, it is evident that all Koopman models exhibit significantly superior performance, characterized by considerably lower MNPE, compared to models linearized around the origin or initial state. This observation alone justifies the utilization of EDMD and Deep-DMD methods, although it is worth noting that such findings are well-documented and extensively studied in existing literature (refer to, for example, [32] and [48]). Furthermore, for nominal basis vector sizes both EDMD approaches demonstrate slightly superior performance compared to Deep-DMD, with the EDMD utilizing a polynomial basis showing the best performance when considering mean and median error metrics. It is possible that this is influenced by the stochastic nature of RBFs and the neural network encoder, although a comprehensive analysis of this aspect is beyond the scope of this thesis. A graphical representation of a single run validating these results can be found in Figure3.4.

Table 3.1: MNPE comparison of different linear predictors for Van der Pol oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
EDMD-RBF ($n_\Phi = 102$)	16.227	14.029	3.5815	157.18
EDMD-Poly ($n_\Phi = 136$)	14.511	13.009	5.2381	138.62
Deep-DMD ($n_{\Phi_e} = 100$)	19.147	17.032	3.4076	107.84
LIN 0	1254.3	1141.2	7.7125	3213.1
LIN x_0	48043	301.9	8.6791	$1.4519 \cdot 10^6$

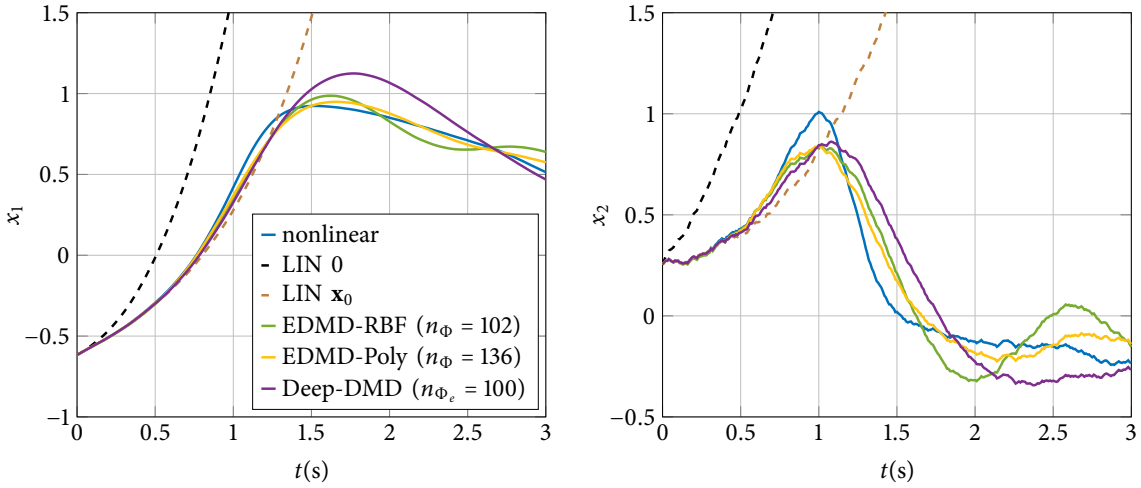

Figure 3.4: Predictor comparison for Van der Pol oscillator with $\mathbf{x}_0 = [-0.62\ 0.26]^T$ and random input signal.

The following text presents an in-depth experimental investigation of the proposed E²DMD methods. In Table3.2, you can find the errors associated with multiple predictors, all operating in an extended state-space with a dimension of $n_w = 5$. Additionally, Figure3.5illustrates a single trajectory. It is noteworthy that all of these predictors exhibit substantial errors and are capable of accurately predicting true trajectories only over a limited prediction horizon.

Table 3.2: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 5$ for Van der Pol oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 5$)	44.66	41.083	10.291	194.61
RBF basis				
E²DMD-DS ($n_w = 5$)	50.017	41.054	20.061	272.42
E²DMD-MS ($n_w = 5$)	88.706	80.925	22.992	251.44
E²DMD-HO ($n_w = 5$)	48.399	42.664	18.673	152.42
Polynomial basis				
E²DMD-DS ($n_w = 5$)	41.259	36.258	18.85	169.13
E²DMD-MS ($n_w = 5$)	61.232	53.94	12.826	238.41
E²DMD-HO ($n_w = 5$)	52.863	47.164	21.97	156.31

Notably, only the subset of states sampled from the polynomial basis vector (E²DMD-DS) eventually converges to the actual trajectory, but this result does not justify the utilization of such a model. In terms of mean and median errors, Deep-DMD outperforms most of the predictors, with the exception of E²DMD-DS using an RBF basis.

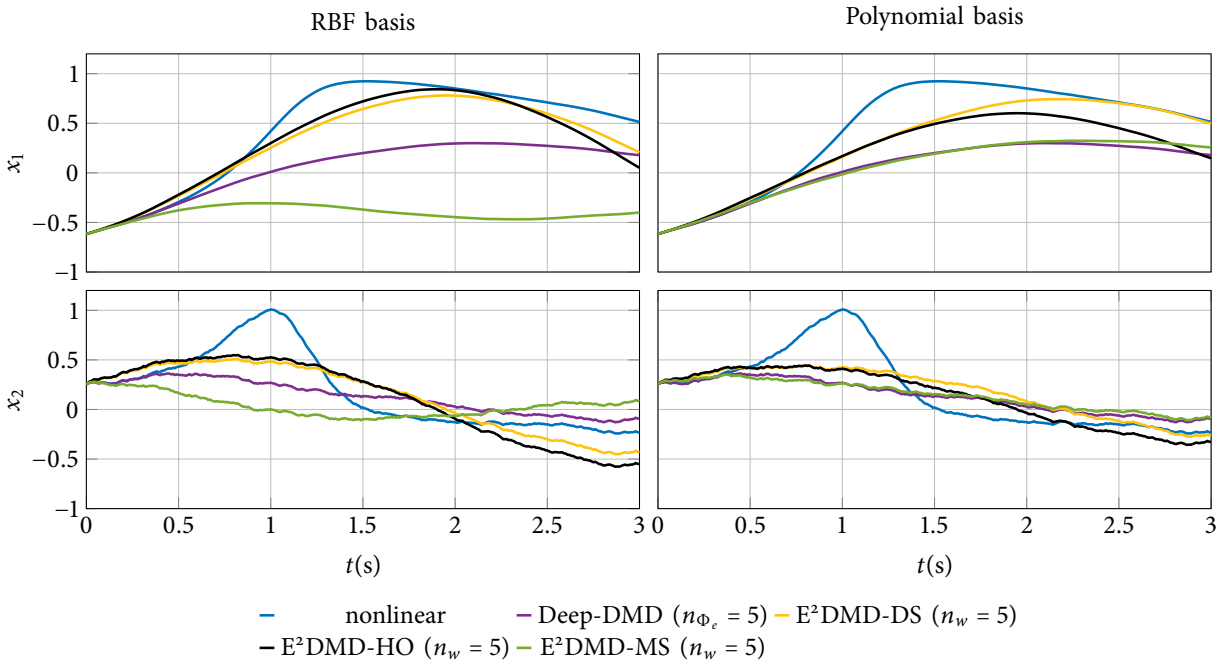

Figure 3.5: Comparison of different Koopman models with reduced state-space of size $n_w = 5$ for Van der Pol oscillator with $\mathbf{x}_0 = [-0.62\ 0.26]^T$ and random input signal.

Errors and trajectories for experiments conducted using predictors of dimension $n_w = 25$ are available in Table3.3and Figure3.6. These results demonstrate lower MNPE when compared to predictors with a dimension of $n_w = 5$, accompanied by an extended time horizon over which they can accurately reconstruct the given trajectory. It is important to note that these time horizons still depend on factors such as the input, initial conditions, and the chosen predictor.

Table 3.3: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 25$ for Van der Pol oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 25$)	34.269	32.38	7.8399	209.91
RBF basis				
E²DMD-DS ($n_w = 25$)	28.205	25.094	7.6633	253.95
E²DMD-MS ($n_w = 25$)	51.133	45.9	7.5924	169.36
E²DMD-HO ($n_w = 25$)	27.397	24.426	5.9043	276.44
Polynomial basis				
E²DMD-DS ($n_w = 25$)	31.534	27.513	15.6	201.29
E²DMD-MS ($n_w = 25$)	42.078	34.749	8.91364	673.03
E²DMD-HO ($n_w = 25$)	35.62	33.323	11.769	385.87

For $n_w = 25$ Deep-DMD performs well, outperforming most of the models, but it is surpassed by E²DMD-DS and E²DMD-HO when using an RBF basis, as well as E²DMD-DS when using a polynomial basis.

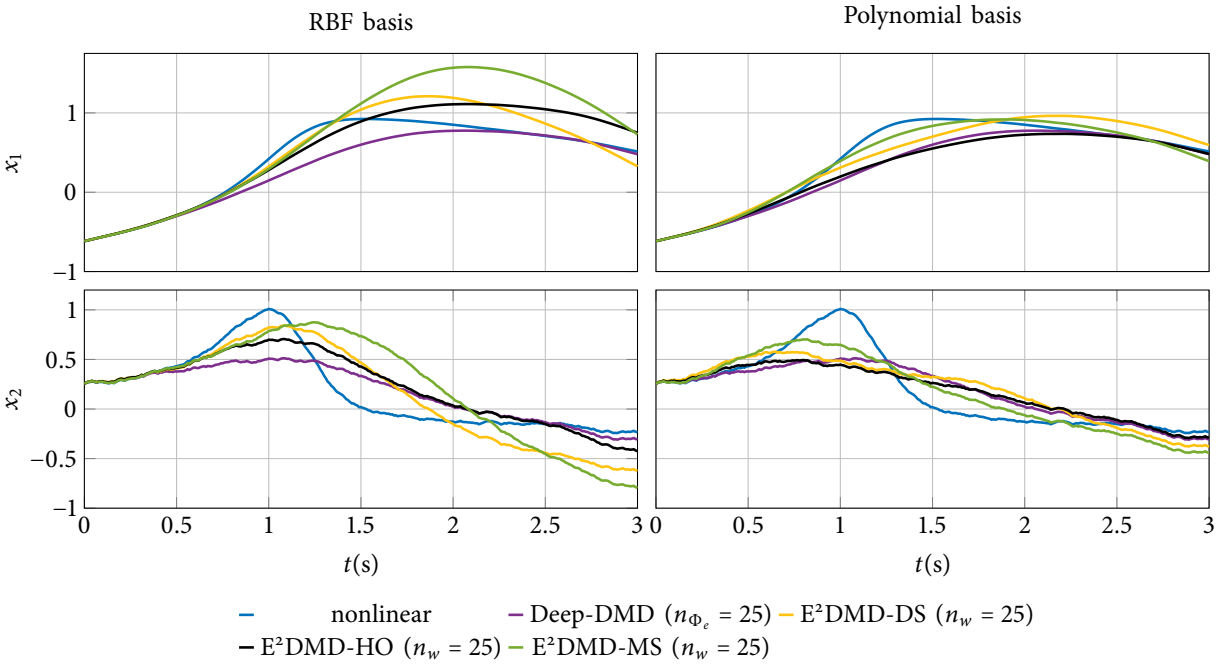

Figure 3.6: Comparison of different Koopman models with reduced state-space of size $n_w = 25$ for Van der Pol oscillator with $\mathbf{x}_0 = [-0.62\ 0.26]^T$ and random input signal.

Table3.4presents MNPE values for predictors with a state-space size of $n_w = 50$. Notably, predictors utilizing an RBF basis outperform those using a polynomial basis, with the top performers once again being E²DMD-DS and E²DMD-HO. However, in this case, Deep-DMD does not exhibit the same level of performance as seen in previous examples, as all the RBF-based predictors perform better, along with E²DMD-DS with a polynomial basis. Corresponding trajectory examples can be found in Figure3.7.

Table 3.4: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 50$ for Van der Pol oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 50$)	25.58	23.783	6.1033	124.81
RBF basis				
E²DMD-DS ($n_w = 50$)	18.721	16.402	4.543	139.93
E²DMD-MS ($n_w = 50$)	24.572	22.032	7.1997	179.76
E²DMD-HO ($n_w = 50$)	19.639	17.354	4.3959	208.32
Polynomial basis				
E²DMD-DS ($n_w = 50$)	24.22	21.554	4.5095	109.55
E²DMD-MS ($n_w = 50$)	33.645	28.196	8.2145	398.2
E²DMD-HO ($n_w = 50$)	26.931	23.854	5.1865	357.64

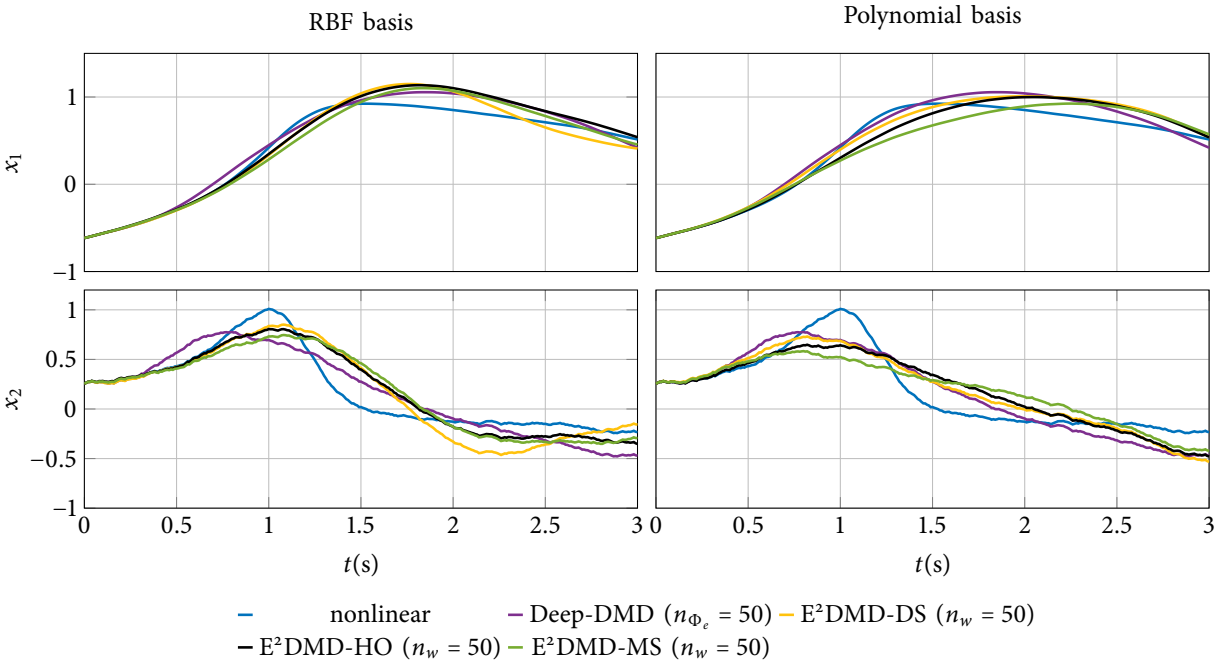

Figure 3.7: Comparison of different Koopman models with reduced state-space of size $n_w = 50$ for Van der Pol oscillator with $\mathbf{x}_0 = [-0.62\ 0.26]^T$ and random input signal.

3.5.3 *Damped Duffing oscillator simulation*

In this analysis of the damped Duffing oscillator, the same methodology as outlined in3.5.2was used. Two types of basis functions were utilized, i.e. thin plate spline radial basis functions with a lifted state-space dimension of $n_\Phi = 102$ and a polynomial basis of the 15th order with $n_\Phi = 136$.

The experiments compared various linearization methods mentioned earlier in the section. The initial conditions and the control inputs for the simulations were randomly selected the same way as for the Van der Pol oscillator, while the effectiveness of the predictors was again assessed using the metric3.46.

In Table3.5one can find the mean, median, minimum and maximum MNPE values. These values are calculated by averaging the errors from 5000 simulation cycles. Similar to the observations made with the Van der Pol oscillator, the Koopman models perform significantly better than the standard linearized models when applied to the Duffing oscillator. Deep-DMD again falls behind compared to the EDMD methods with RBF and polynomial basis, with a performance gap exceeding a factor of 3.

Table 3.5: MNPE comparison of different linear predictors for damped Duffing oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
EDMD-RBF ($n_\Phi = 102$)	8.81	6.0412	0.46685	463.46
EDMD-Poly ($n_\Phi = 136$)	8.3558	4.9683	0.3290	238.51
Deep-DMD ($n_{\Phi_e} = 100$)	25.402	19.156	0.83253	663.38
LIN 0	468.79	309.04	0.020536	2769.3
LIN x_0	121.57	113.92	0.042793	1566.8

The average outcomes find support in the performance of a single trajectory illustrated in Figure3.8. In this particular trajectory, the EDMD-RBF model has an error rate of 6.9%,the EDMD-Poly model of 2.55%, and the Deep-DMD model has a higher error of 13.95%.

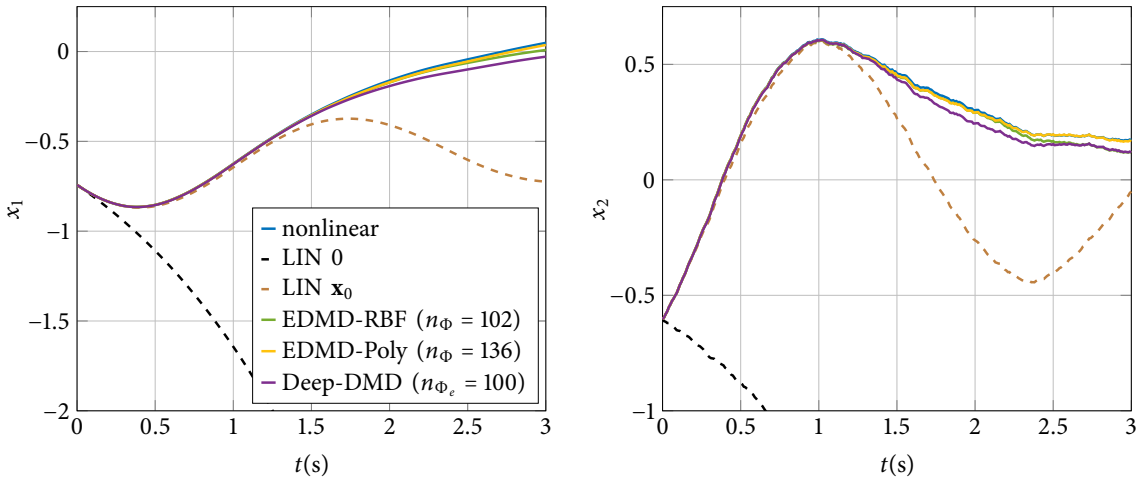

Figure 3.8: Predictor comparison for damped Duffing oscillator with $\mathbf{x}_0 = [-0.74 \ -0.6]^T$ and random input signal.

The Table3.6presents errors that reinforce the observation that, on average, predictors with a state-space dimension of $n_w = 5$ struggle to make accurate predictions of real trajectories. Models employing radial basis function reduction perform better than those utilizing polynomial basis reduction, with Deep-DMD falling somewhere in between. A specific example of a trajectory that illustrates these predictions can be seen in Figure3.9.

Table 3.6: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 5$ for damped Duffing oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 5$)	47.354	42.837	4.5246	425.49
RBF basis				
E²DMD-DS ($n_w = 5$)	42.445	36.672	0.75585	709.54
E²DMD-MS ($n_w = 5$)	49.165	41.091	1.9935	600.87
E²DMD-HO ($n_w = 5$)	37.147	32.764	0.95286	681.74
Polynomial basis				
E²DMD-DS ($n_w = 5$)	56.899	51.677	8.2602	374.57
E²DMD-MS ($n_w = 5$)	52.965	41.716	2.901	1450
E²DMD-HO ($n_w = 5$)	50.018	45.091	7.8628	407.93

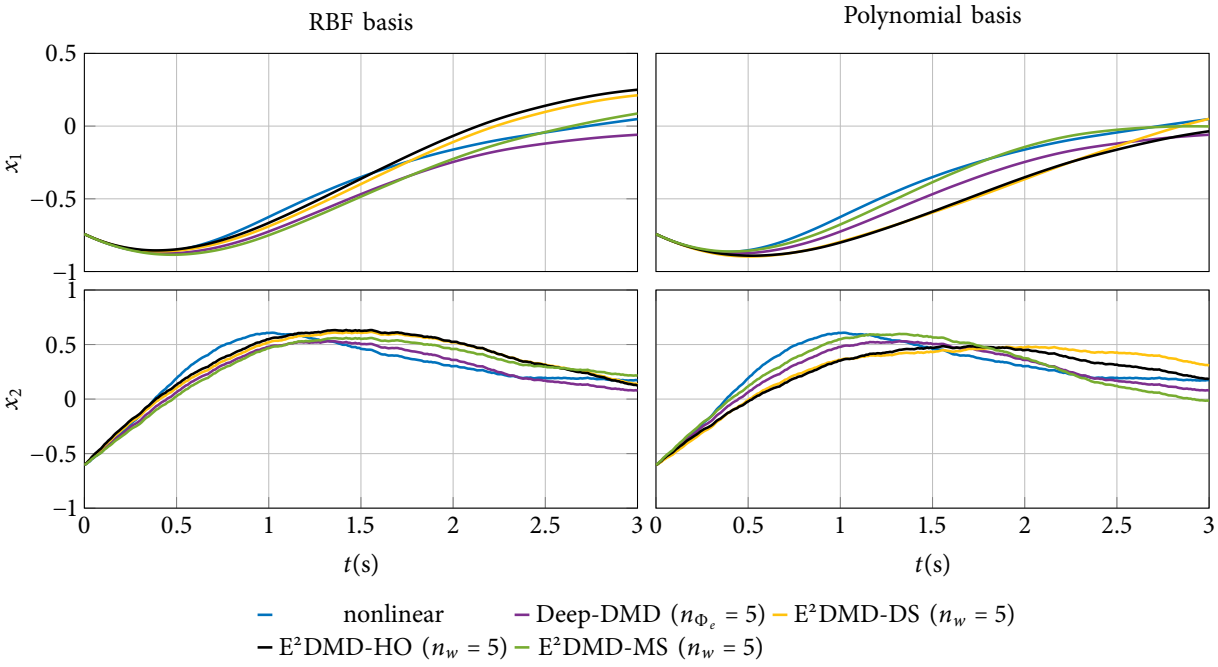

Figure 3.9: Comparison of different Koopman models with reduced state-space of size $n_w = 5$ for damped Duffing oscillator with $\mathbf{x}_0 = \begin{bmatrix} -0.74 & -0.6 \end{bmatrix}^T$ and random input signal.

For $n_w = 25$ the errors are smaller than for $n_w = 5$, but still quite significant. This information is presented in Table3.7and Figure3.10. When it comes to the model comparison, both the polynomial basis vector reduction and the RBF basis vector reduction as well as Deep-DMD perform similarly. In this particular case, however, the polynomial basis version of E²DMD-MS stands out as the worst model, but achieves a relatively small minimum MNPE value.

Table 3.7: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 25$ for damped Duffing oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 25$)	28.022	22.527	1.4965	458.82
RBF basis				
E²DMD-DS ($n_w = 25$)	19.369	14.51	1.135	444.89
E²DMD-MS ($n_w = 25$)	31.845	23.708	1.7862	829.02
E²DMD-HO ($n_w = 25$)	21.273	16.269	1.3793	567.01
Polynomial basis				
E²DMD-DS ($n_w = 25$)	20.621	16.37	1.4006	484.95
E²DMD-MS ($n_w = 25$)	49.758	37.009	1.2386	435.91
E²DMD-HO ($n_w = 25$)	31.003	26.677	1.7604	322.35

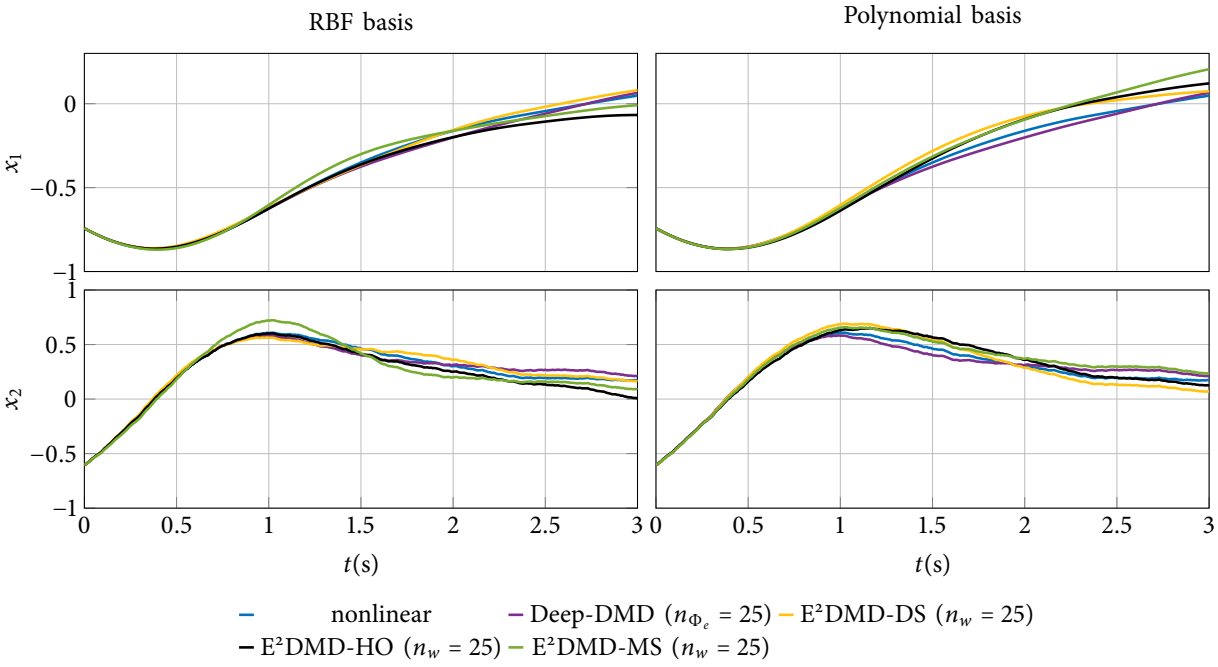

Figure 3.10: Comparison of different Koopman models with reduced state-space of size $n_w = 25$ for damped Duffing oscillator with $\mathbf{x}_0 = [-0.74 \ -0.6]^T$ and random input signal.

Table3.8presents the MNPE values for models with a state-space dimension of $n_w = 50$. These models exhibit lower errors compared to those with smaller dimensions. Notably, the models utilizing polynomial basis reduction perform less effectively than those employing RBF basis reduction. Furthermore, in this specific context, Deep-DMD demonstrates inferior performance compared to most models, except for the polynomial basis version of E²DMD-MS. The subpar performance of E²DMD-MS is also visually represented in Figure3.11. In contrast, the other models appear to provide relatively accurate predictions of the system's behaviour.

Table 3.8: MNPE comparison of different Koopman models with RBF basis and reduced state-space of size $n_w = 50$ for damped Duffing oscillator.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 50$)	20.758	15.198	0.51734	668.83
RBF basis				
E²DMD-DS ($n_w = 50$)	11.596	8.9162	0.74944	677.75
E²DMD-MS ($n_w = 50$)	16.137	12.564	1.3637	491.88
E²DMD-HO ($n_w = 50$)	11.613	8.5644	1.0996	433.61
Polynomial basis				
E²DMD-DS ($n_w = 50$)	17.169	12.897	0.88135	583.49
E²DMD-MS ($n_w = 50$)	31.501	21.74	0.67343	1042.9
E²DMD-HO ($n_w = 50$)	17.277	13.039	1.3875	438.29

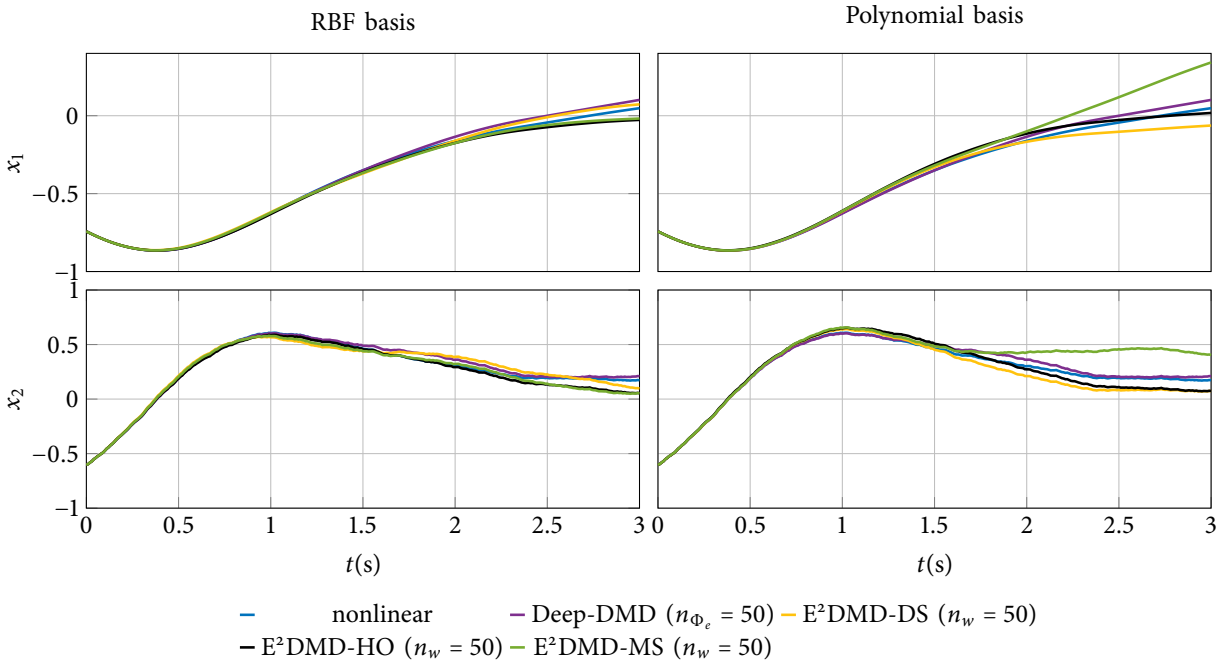

Figure 3.11: Comparison of different Koopman models with reduced state-space of size $n_w = 50$ for damped Duffing oscillator with $\mathbf{x}_0 = [-0.74 \ -0.6]^T$ and random input signal.

3.5.4 *Bilinear motor simulation*

Unlike systems previously discussed, the bilinear motor model does not satisfy a linear state-input relationship described by (3.8). Consequently, the state-space of the bilinear motor model (3.45), as elaborated in3.1.3, requres an expansion by incorporating an input as a new state in order to learn EDMD-based models. This additional state is defined as $x_3 = u$, and the input of the model is updated to the derivative of the original input signal, symbolized as $\tilde{u} = \dot{u}$. The modified model is expressed as follows:

$$\begin{aligned} \dot{x}_1 &= -\frac{R_a}{L_a}x_1 - \frac{k_m}{L_a}x_2x_3 - \frac{u_a}{L_a}, \\ \dot{x}_2 &= -\frac{B}{J}x_2 - \frac{k_m}{J}x_1x_3 - \frac{\tau_l}{J}, \\ \dot{x}_3 &= \tilde{u}. \end{aligned} \tag{3.47}$$

This extended nonlinear model's primary use was in learning Koopman models, whereas the simulations utilized the original bilinear motor model as referenced in equation (3.45).

Once again, two types of basis functions were used:

1. A set of 100 thin plate spline radial basis functions with centers sampled randomly across the unit box using a uniform distribution, and the original states were included in the expanded state-space vector. Lifted state-space has a dimension of $n_\Phi = 103$.

2. A polynomial basis of the order $d = 8$, which expanded the lifted state-space dimension to $n_\Phi = 165$.

The experiments evaluate the performance of the linearization techniques previously mentioned. Random selection of initial conditions and control inputs was employed, sampled from a

uniform distribution within the $[-1,1]^2$ box and the $[-1,1]$ unit interval, respectively. The simulations were conducted with a set sampling time of $T_s = 0.01$ seconds and lasted for $T_{sim} = 1$ second each. The output mapping was defined as $\mathbf{y} = [x_1\ x_2]^T$ and the efficiency was evaluated using3.46.

A bilinear motor model seems to be easy to identify with Koopman operator-based approaches, as can be seen from the comparison in Table3.9and Figure3.12, where various linear predictors were evaluated. Notably, both EDMD methods and the Deep-DMD method exhibit nearly identical performance and significantly outperform the traditional linearized models. This is particularly true for the LIN 0 model, as its responses show a rapid and substantial increase in (absolute) value.

Table 3.9: MNPE comparison of different linear predictors for bilinear motor.

Model	MNPE [%]			
	Mean	Median	Min	Max
EDMD-RBF ($n_\Phi = 103$)	12.776	11.871	3.4923	73.64
EDMD-Poly ($n_\Phi = 165$)	12.355	11.598	3.4339	91.867
Deep-DMD ($n_{\Phi_e} = 100$)	12.31	11.546	3.4302	87.151
LIN 0	12463	12333	7412	22101
LIN x_0	45.477	36.985	5.8264	319.11

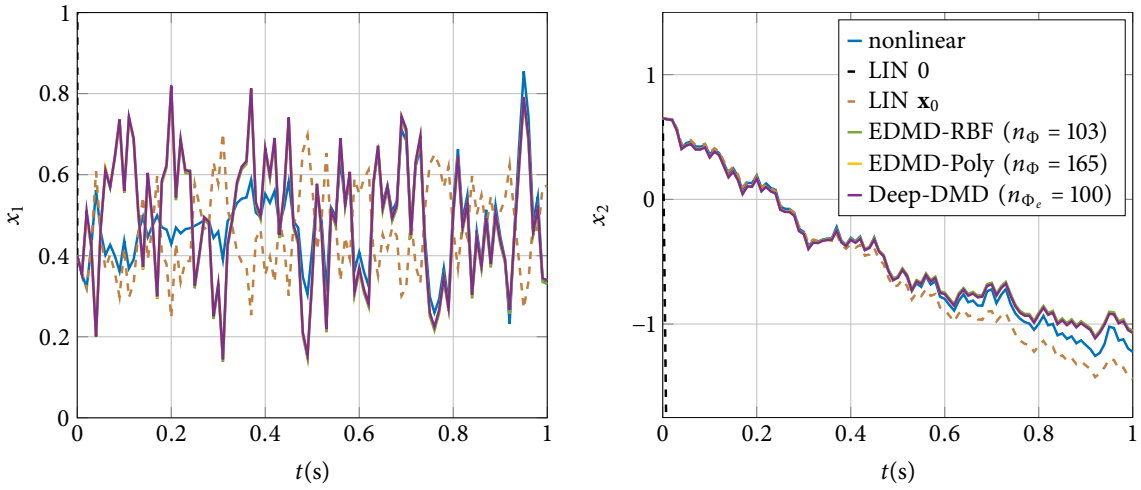

Figure 3.12: Predictor comparison for bilinear motor with $\mathbf{x}_0 = [0.4\ 0.65]^T$ and random input signal.

The claim that Koopman operator approaches can effectively capture the bilinear motor model is supported by the results presented in Table3.10. In this table, it is clear that even in cases where the dimensionality is significantly reduced ($n_w = 5$), the mean normalized prediction error remains small. However, it should be noted that this claim does not apply when considering the optimized discrete selection (E²DMD-DS) from the radial basis function (RBF) basis vectors. Furthermore, the claim does not apply to E²DMD-HO, for both the RBF and the polynomial basis reduction. An illustrative example of such a scenario is shown in Figure3.13.

Table 3.10: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 5$ for bilinear motor.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 5$)	15.067	14.06	3.6133	114.76
RBF basis				
E²DMD-DS ($n_w = 5$)	25.996	25.29	3.7378	106.33
E²DMD-MS ($n_w = 5$)	15.023	13.645	2.9147	51.694
E²DMD-HO ($n_w = 5$)	36.706	34.577	3.5545	100.24
Polynomial basis				
E²DMD-DS ($n_w = 5$)	13.104	12.164	3.4536	115.64
E²DMD-MS ($n_w = 5$)	14.328	13.346	3.3613	110.09
E²DMD-HO ($n_w = 5$)	92.227	85.7	4.5666	292.05

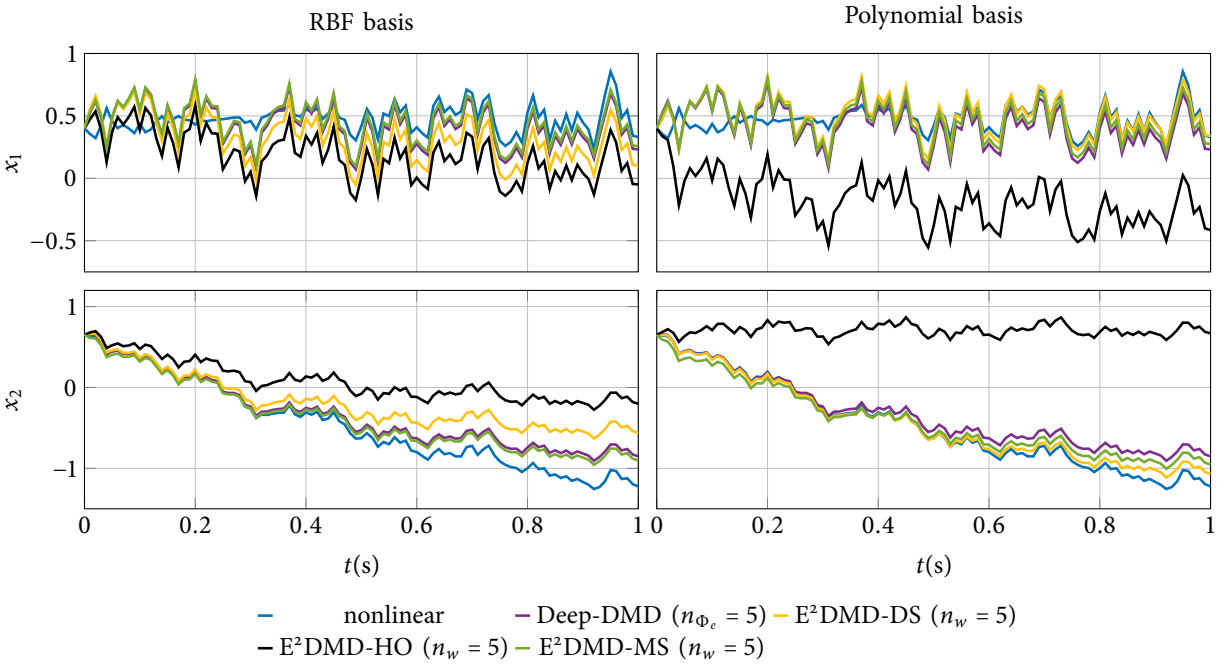

Figure 3.13: Comparison of different Koopman models with reduced state-space of size $n_w = 5$ for bilinear motor with $\mathbf{x}_0 = [0.4\ 0.65]^T$ and random input signal.

With an increase in the dimensionality of the lifted state-space to $n_w = 25$, the performance of the associated predictor also demonstrates improvement. In Table3.11, it becomes evident that nearly all mean normalized prediction error (MNPE) values approach those of the non-reduced predictors. The only exception is the E²DMD-HO based model with a polynomial basis. These findings are substantiated by the accompanying graphs in Figure3.14.

Table 3.11: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 25$ for bilinear motor.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 25$)	12.608	11.854	3.412	81.652
RBF basis				
E²DMD-DS ($n_w = 25$)	13.355	12.082	3.7011	60.88
E²DMD-MS ($n_w = 25$)	12.919	12.284	2.9244	53.7
E²DMD-HO ($n_w = 25$)	12.912	11.98	2.9262	47.567
Polynomial basis				
E²DMD-DS ($n_w = 25$)	12.29	11.521	3.438	91.674
E²DMD-MS ($n_w = 25$)	13.667	12.797	3.3835	68.662
E²DMD-HO ($n_w = 25$)	20.284	15.426	2.8414	186.65

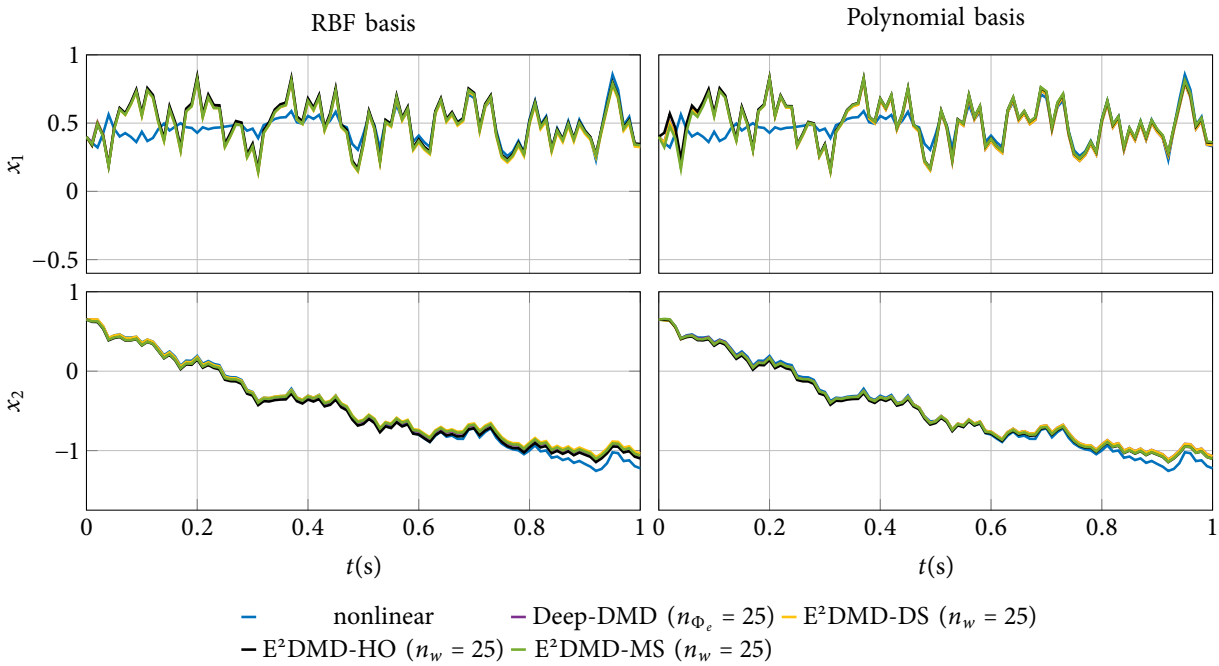

Figure 3.14: Comparison of different Koopman models with reduced state-space of size $n_w = 25$ for bilinear motor with $\mathbf{x}_0 = [0.4\ 0.65]^T$ and random input signal.

As the state-space dimension of the models keeps increasing and reaches $n_w = 50$, they come even closer to achieving accuracy levels similar to the original non-reduced models. The outcomes presented in Table3.12closely resemble those found in Table3.9, while the responses depicted in Figure3.15bear a resemblance to those in Figure3.12.

Table 3.12: MNPE comparison of different Koopman models with reduced state-space of size $n_w = 50$ for bilinear motor.

Model	MNPE [%]			
	Mean	Median	Min	Max
Neural network basis				
Deep-DMD ($n_{\Phi_e} = 50$)	12.365	11.613	3.4331	84.459
RBF basis				
E²DMD-DS ($n_w = 50$)	13.045	11.973	3.0681	54.123
E²DMD-MS ($n_w = 50$)	12.568	11.789	3.1137	47.747
E²DMD-HO ($n_w = 50$)	12.578	11.799	3.2082	51.125
Polynomial basis				
E²DMD-DS ($n_w = 50$)	12.418	11.665	3.439	101.37
E²DMD-MS ($n_w = 50$)	13.651	13.158	3.4005	73.023
E²DMD-HO ($n_w = 50$)	14.05	12.769	2.6828	121.49

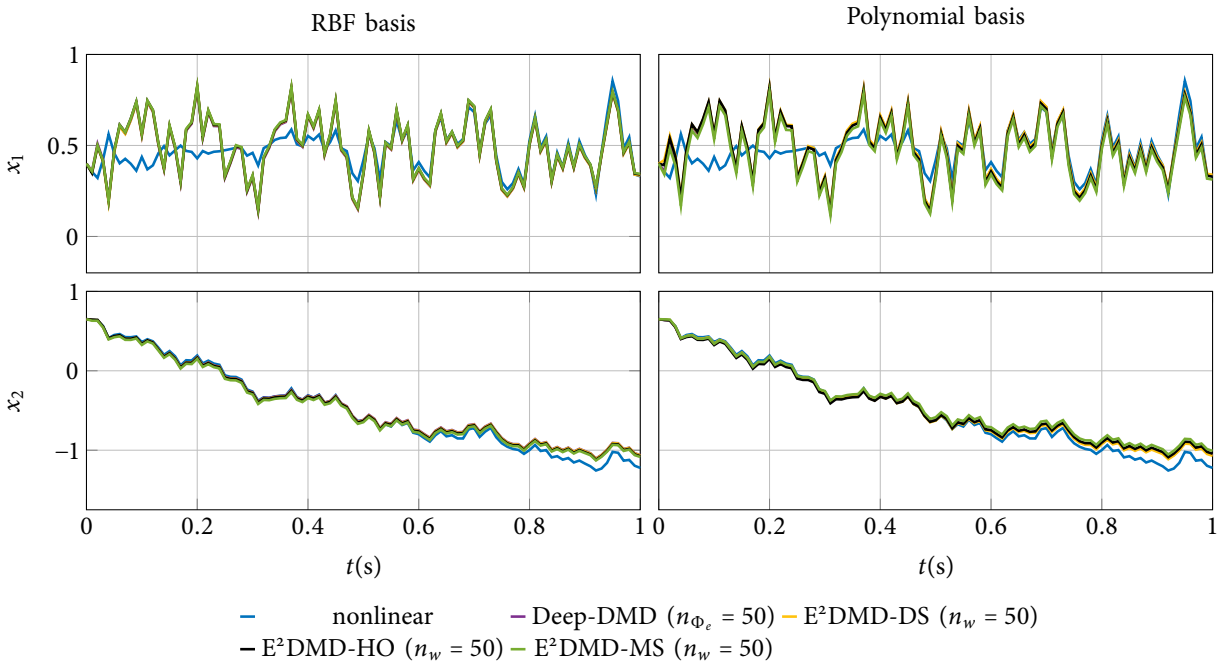

Figure 3.15: Comparison of different Koopman models with reduced state-space of size $n_w = 50$ for bilinear motor with $\mathbf{x}_0 = [0.4\ 0.65]^T$ and random input signal.

3.5.5 *Concluding remarks*

The different methods showed varying performance when applied to the different dimensions of the lifted state-spaces and the specific dynamical systems in question. Table3.13displays the combinations of dynamical systems and lifted state-space dimensions together with the specific E²DMD numerical method that performed best in terms of mean MNPE for each pair. E²DMD-DS performed best in most cases, while E²DMD-MS consistently performed worst. It is also worth noting that, contrary to expectations based on previous studies (such as [68] and [67]), Deep-DMD only outperformed the proposed predictors in certain cases.

Table 3.13: The combination of state-space dimension and dynamical system, along with the E²DMD method which performed the best in the specific scenario.

System	Van der Pol oscillator			Damped Duffing oscillator			Bilinear motor		
n_w	5	25	50	5	25	50	5	25	50
RBF basis	HO	HO	DS	HO	DS	DS	MS	HO	MS
Polynomial basis	DS	DS	DS	HO	DS	DS	DS	DS	DS

An important observation is that the optimization methods used for E²DMD rely on stochastic algorithms, leading to potential variations in performance upon repeated learning. Additionally, in the case of the bilinear motor, the mean errors exhibited similar values, particularly for $n_w = 25$ and $n_w = 50$. Several important findings emerge from this analysis:

- For $n_w = 5$, all predictors performed poorly, limiting the significance of their comparison.
- Increasing the lifted state-space dimensions generally led to improved model performance, which is consistent with previous research, e.g. [83].
- Surprisingly, the Deep-DMD algorithm deviated from expectations, possibly indicating problems with the network architecture, loss function or other design features, though these are beyond the scope of this thesis.
- E²DMD-HO and E²DMD-MS are likely to perform similarly well as they both use an affine transformation, whereas E²DMD-DS with its binary values and lack of a bias term is expected to perform less effectively. The fact that this investigation yielded different results suggests that the choice of training optimization method can have a significant and often unexpected impact on predictor performance.
- Given the similarity between E²DMD-MS and Deep-DMD algorithms, addressing Deep-DMD problems could also improve the performance of E²DMD-MS.
- E²DMD simplifies the reduction of lifted state-space compared to manual selection in EDMD and includes fewer hyperparameters than Deep-DMD, making it easier to fine-tune.
- Ultimately, the choice of numerical method depends on the specific dynamical system under investigation.

3.6 SUMMARY

In this chapter, various techniques for data-driven identification of Koopman models are presented. The first two sections introduce the well-recognized extended dynamic mode decomposition (EDMD) and deep dynamic mode decomposition (Deep-DMD), which have been extensively documented in the literature, serving as the foundation for subsequent discussions.

The core contribution of this thesis lies in the introduction of enhanced extended dynamic mode decomposition (E²DMD). This novel approach aims to employ an affine transformation to reduce the dimensionality of the lifted state vector used in EDMD, while striving to minimize any impact on model accuracy. It simplifies the reduction of lifted state-space compared to manual selection in EDMD and includes fewer hyperparameters than Deep-DMD, making it easier to fine-tune. Three distinct numerical methods for obtaining E²DMD models are presented and explained:

1. Basis function reduction by discrete selection (E²DMD-DS): This method systematically selects certain elements from the original basis function vector to reduce the basis function dimensionality and still efficiently capture essential system dynamics.

2. Multiple step prediction learning algorithm (E²DMD-MS): A learning algorithm which uses the same optimization method as Deep-DMD with the changed encoder structure.

3. Basis function reduction as a hyperparameter optimization problem (E²DMD-HO): Treating basis function reduction as a hyperparameter optimization problem allows for optimization problem separation and enables usage of existing hyperparameter optimization frameworks for this purpose.

In addition to E²DMD, other data-driven Koopman identification approaches are briefly mentioned to provide a broader perspective.

The chapter concludes with an evaluation of the linear predictors using three benchmark dynamical systems: the Van der Pol oscillator, the damped Duffing oscillator and the bilinear motor. The simulation results are compared and analyzed to evaluate the effectiveness of the different methods in capturing the system dynamics.

Key findings include a consistent improvement in predictor performance as the dimension of the state-space increases. Surprising variations in Deep-DMD performance were also found, which could indicate architectural or design issues. Solving Deep-DMD problems may also improve the performance of E²DMD-MS, as both are trained with similar algorithms. E²DMD-HO and E²DMD-MS are expected to perform similarly due to the use of an affine transformation, while E²DMD-DS is expected to be less effective due to the binary values and the absence of a bias term. The results of this study, from which different conclusions can be drawn, suggest that the optimization method used for training can have a significant and unexpected impact on the performance of the predictor.

Finally, the choice of numerical method depends on the specific dynamical system under investigation. This is also confirmed by the results in Chapter5, where the analysis leads to different conclusions regarding the proposed approaches.

4

Koopman-based vehicle control using tire slip

In this chapter the application of the Koopman operator for modelling and controlling vehicle dynamics using basic vehicle models and longitudinal tire slip is investigated, focusing on two different proof-of-concept approaches. The first section deals with the identification of a bicycle model, without a tire model, using the EDMD algorithm. The obtained model is then compared with the models created using Taylor expansion based linearization. Further validation is performed by incorporating this model into the MPC design and applying it to the nonlinear bicycle vehicle model. Both the model identification and the predictive controller design were validated using MATLAB and Simulink. The second section deals with the Koopman operator-based identification of a two-track vehicle model integrated with a torque (slip) vectoring algorithm via MPC. In this part, efforts to improve the stability of manually steered vehicles are emphasized, with a novel attempt to implement DYC with linear MPC based on the Koopman operator model. The EDMD method is again used for the model approximation, with the additional consideration of the requirement for a linear dependence of the system state propagation on the inputs mentioned in3.1.3. This linear dependency, which is generally not present in vehicles, is achieved by carefully selected nonlinear transformations. Once an accurate model is created, it is used to formulate a linear KMPC algorithm. This KMPC is then compared with a linear time-variant MPC for different prediction horizons.

4.1 Koopman operator-based control using bicycle model

This section introduces the research conducted in [84]. It involves the application of the EDMD identification method as outlined in [83] to the bicycle vehicle model equipped with front-wheel steering. Furthermore, the approach is expanded by incorporating linear MPC design. Unlike the approach described in [68], this method utilizes tire slip on both wheels, in addition to the steering angle, as control inputs, represented as $\mathbf{u} = [\delta_f \; s_{fx} \; s_{rx}]^T$.

4.1.1 *Three state bicycle model without tire model*

When the EDMD method is employed to approximate the model of a system, it requires a linear relationship between state derivatives and inputs, as dictated by the methodology outlined in3.1. To achieve this, the original bicycle model, detailed in2.1.1, is simplified by excluding equations

(2.4) to (2.6), resulting in the following expressions:

$$m\dot{v}_x = m\dot{\theta}_z v_y + F_{fx} + F_{rx} - \frac{1}{2}c_w \rho A_w v_x \sqrt{v_x^2 + v_y^2}, \tag{4.1}$$

$$m\dot{v}_y = -m\dot{\theta}_z v_x + F_{fy} + F_{ry} - \frac{1}{2}c_w \rho A_w v_y \sqrt{v_x^2 + v_y^2}, \tag{4.2}$$

$$J_z \ddot{\theta}_z = l_f F_{fy} - l_r F_{ry}. \tag{4.3}$$

In this reduced model, the state vector is represented as $\mathbf{x} = [v_x \; v_y \; \dot{\theta}_z]^T$, and the input vector is replaced by $\bar{\mathbf{u}} = [F_{fx} \; F_{fy} \; F_{rx}]^T$. This simplified model is employed for system identification in 4.1.2and for MPC system design in4.1.3. Meanwhile, the linear force model (2.39) is utilized to convert input forces into steering angles and tire slips in4.1.5. The parameter values of the model are given in Table4.1.

Table 4.1: Vehicle model parameters

Parameter	Description	Value	Unit
m	mass of the vehicle	1752	kg
l_f	front axle to CoG distance	1.435	m
l_r	rear axle to CoG distance	1.31	m
J_z	moment of inertia around yaw axis	2286	kg m²
c_w	drag coefficient	0.31	-
ρ	air density	1.2	kg / m ³
A_w	surface exposed to the air flow	2.2	m²
C_x	longitudinal tire stiffness	87712	N
C_y	lateral tire stiffness	51488	N/rad

4.1.2 *Koopman model identification*

To initiate the process of identifying a vehicle dynamics model, the first step involves selecting the learning dataset. All state variables are sampled uniformly from predefined intervals: $v_x \in [10,\ 50]$ m/s, $v_y \in [-30,\ 30]$ m/s, and $\dot{\theta}_z \in [-10,\ 10]$ rad/s. Simultaneously, forces are uniformly sampled within the interval $[-F_m,\ F_m]$, where $F_m \in \{5, 50, 500, 5000\}$ N. Each interval consists of $N_s = 20$ samples. When combined in all possible combinations, these sampled vectors result in a dataset containing a total of $4 \cdot 20^4$ values.

The subsequent step is the selection of basis functions denoted as $\phi(\cdot)$. In the absence of specific recommendations for the basis function vector, polynomial basis functions are opted for, as suggested in [83]. This polynomial basis consists of monomials derived from the elements of the state vector:

$$P_d = \{v_x^a \cdot v_y^b \cdot \dot{\theta}_z^c \mid a, b, c \in \mathbb{N} \cup \{0\},\ a + b + c \leq d\}, \tag{4.4}$$

where d represents the order of the basis P_d. The EDMD-based model is learned using bases of varying orders. Subsequently, simulations are conducted with the corresponding models, and the root mean square error (RMSE) is computed using the following formula:

$$RMSE = 100\frac{\sqrt{\sum_k \|\hat{\mathbf{x}}_k - \mathbf{x}_k\|_2^2}}{\sqrt{\sum_k \|\mathbf{x}_k\|_2^2}}. \tag{4.5}$$

Here, $\hat{\mathbf{x}}$ and $\mathbf{x}$ represent the state vectors obtained at time step k from the Koopman operator-based approximation and the actual nonlinear system, respectively. RMSE is assessed across 5000 trajectories, featuring the same initial condition distribution as the learning dataset, with a sampling time of $T_s = 0.01$ s and each trajectory being composed of $N_{traj} = 30$ samples. RMSE values for various basis orders are presented in Table4.2.

Table 4.2: Bicycle model prediction RMSE for polynomial basis functions of different orders.

d	1	2	3	4	5	6	7	8	9
$RMSE$ [%]	43.33	36.29	14.56	9.71	8.65	20.18	61.47	123.66	2509.78

Finally, polynomial basis function of order $d = 4$ is selected, resulting in an average RMSE of 9.7089 % and a basis cardinality of $n_\Phi = 35$. This choice is made as increasing the order to $d = 5$ leads to a more than 50 % increase in cardinality, while the RMSE improvement is less than one percent. An example simulation is depicted in Figure4.1. In this simulation, predictions are restarted every $\Delta t = 0.5$ s to emulate MPC execution. A comparison is made between the real nonlinear model, the linear model approximated by the Koopman operator with a polynomial basis of order $d = 4$, the model iteratively linearized at each restart of the prediction (every 0.5 s), and a model linearized around the origin. The initial state is randomly set as $\mathbf{x}_0 = [37.7777\ -8.2569\ -1.5693]^T$, and the input signal is randomly sampled from a uniform distribution within the interval $[-50000,\ 50000]$ N. It is evident that the Koopman operator results in a superior approximation compared to the other two linear approximations.

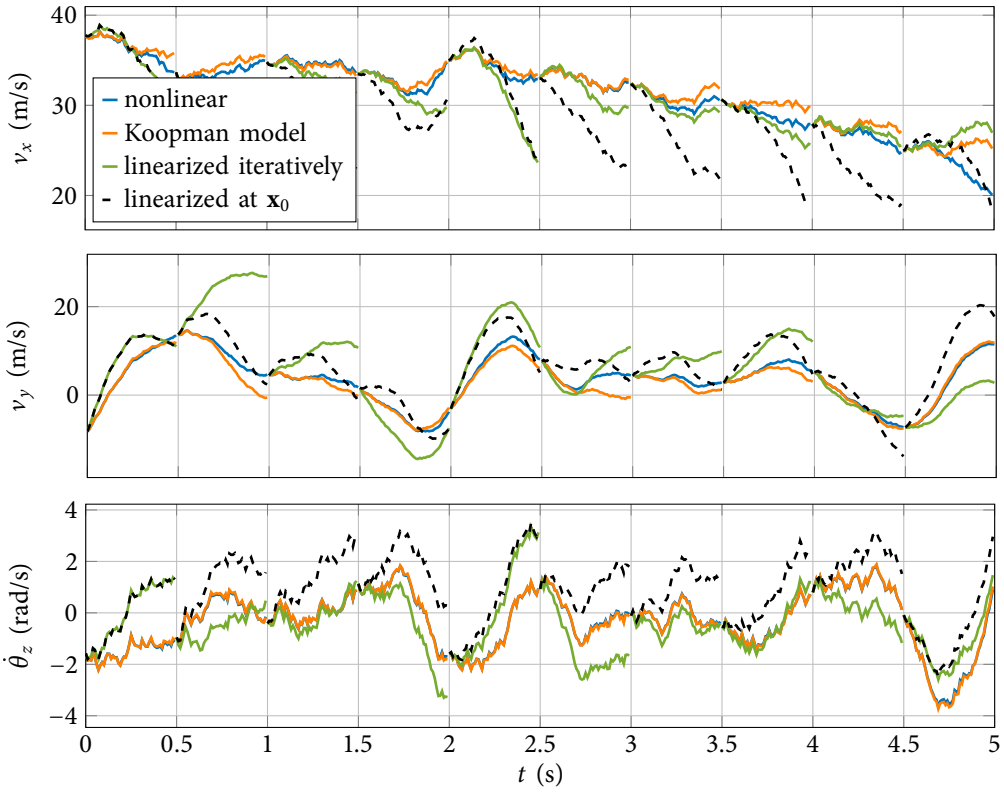

Figure 4.1: Comparison between Koopman model, iteratively linearized model and the model linearized at the origin. Approximations are done for the bicycle vehicle model.

4.1.3 *Koopman MPC design*

The initial step in the development of the KMPC system involves the identification of the linearized Koopman model, as illustrated in3.1. This identified model is then utilized to lift the current system state into a newly created higher-dimensional space. Subsequently, this linear model is employed for the propagation of the system state within the MPC algorithm. Once the optimization problem is resolved, an input transformation process is carried out to convert wheel forces into steering angles and tire slips, which are subsequently applied to the vehicle. The process diagram is depicted in Figure4.2, and the required steps are enumerated as follows:

1. Measure the current state $\mathbf{x}_t$.
2. Utilize the identified model to determine the elevated state $\mathbf{z}_t$ and forward it to the MPC.
3. Solve the MPC problem and obtain the optimal control input $\bar{\mathbf{u}}_t$ in the form of longitudinal and lateral tire forces.
4. Convert the optimal forces (vector $\bar{\mathbf{u}}_t$) into steering angles and tire slip (vector $\mathbf{u}_t$) and apply them to the vehicle.

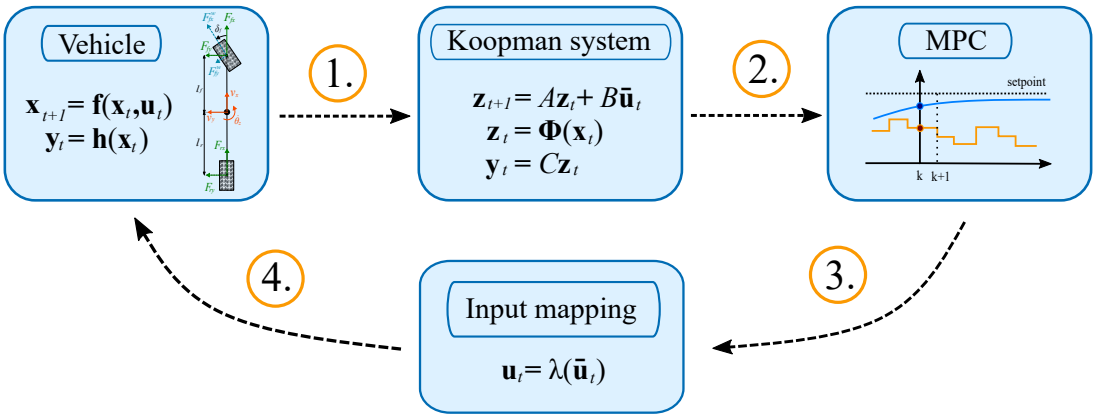

Figure 4.2: Diagram showing KMPC framework. All the steps are executed in a loop.

The formulation of the MPC problem can be described as follows:

$$\begin{aligned} \min_{\bar{\mathbf{U}}_t} \quad & J(\mathbf{z}_t, \bar{\mathbf{u}}_{t-1}, \mathbf{Y}_t^{ref}, \bar{\mathbf{U}}_t) \\ \text{s.t.} \quad & \mathbf{z}_{k+1} = A\mathbf{z}_k + B\bar{\mathbf{u}}_k, \quad k = t, \ldots, t+N-1, \\ & \mathbf{y}_k = C\mathbf{z}_k, \\ & E_k\bar{\mathbf{u}}_k \leq \mathbf{e}_k, \\ & \mathbf{z}_t = \Phi(\mathbf{x}(t)), \\ & \bar{\mathbf{u}}_{t-1} = \bar{\mathbf{u}}(t-1). \end{aligned} \tag{4.6}$$

Here, $\bar{\mathbf{U}}_t = [\bar{\mathbf{u}}_t, \bar{\mathbf{u}}_{t+1}, \ldots \bar{\mathbf{u}}_{t+N-1}]$ represents the input sequence, $\mathbf{Y}_t^{ref} = \left[\mathbf{y}_t^{ref}, \mathbf{y}_{t+1}^{ref}, \ldots \mathbf{y}_{t+N}^{ref}\right]$ represents the reference sequence, and $\mathbf{y}_k = [v_{xk}\ \dot{\theta}_{zk}]^T$ is the output vector. The matrix E_k and the vector $\mathbf{e}_k$ represent the input constraint. The cost function is given by:

$$J(\mathbf{z}_t, \bar{\mathbf{u}}_{t-1}, \mathbf{Y}_t^{ref}, \bar{\mathbf{U}}_t) = \sum_{k=t}^{t+N-1} \|\bar{\mathbf{u}}_k\|_S^2 + \|\Delta\bar{\mathbf{u}}_k\|_R^2 + \sum_{k=t}^{t+N} \|\mathbf{y}_k - \mathbf{y}_k^{ref}\|_Q^2, \tag{4.7}$$

where $Q = Q^T \succeq 0$, $R = R^T \succeq 0$ and $S = S^T \succeq 0$ are the weight matrices and $\Delta\tilde{\mathbf{u}}_k = \tilde{\mathbf{u}}_k - \tilde{\mathbf{u}}_{k-1}$ is the input rate.

4.1.4 *Reference generation*

The estimation of the yaw rate reference is performed using a kinematic bicycle model with zero slip angles on all wheels, as outlined in [9]:

$$\dot{\theta}_z^{ref} = \frac{v_x^{ref}}{l_f + l_r} \tan \delta_f. \tag{4.8}$$

However, it is worth noting that this model is particularly applicable for describing vehicle motion at low velocities. Therefore, at higher velocities, the controlled vehicle may not be able to reach the generated references. The reference vector comprises the longitudinal velocity and yaw rate, denoted as $\mathbf{y}^{ref} = [v_x^{ref} \ \dot{\theta}_z^{ref}]^T$.

4.1.5 *Input mapping*

The tire forces, represented as $\tilde{\mathbf{u}}_t = [F_{fx} \ F_{fy} \ F_{rx}]^T$, derived from the MPC algorithm, need to be converted into steering angle and tire slips, denoted as $\mathbf{u}_t = [\delta_f \ s_{fx} \ s_{rx}]^T$, before they can be applied to the system. In other words, the transformation $\mathbf{u}_t = \lambda(\tilde{\mathbf{u}}_t)$ must be determined. To achieve this transformation, the equations from (2.10) to (2.13) are employed to establish an expression describing the front tire slip angle $\alpha_f = \alpha_f(\delta_f)$ based on the measured state vector. Subsequently, by inverting (2.7) and combining it with (2.39), a set of nonlinear equations is obtained:

$$\begin{aligned} F_{fx} &= C_x s_{fx} \cos \delta_f + C_y \alpha_f(\delta_f) \sin \delta_f, \\ F_{fy} &= C_x s_{fx} \sin \delta_f - C_y \alpha_f(\delta_f) \cos \delta_f, \\ F_{rx} &= C_x s_{rx}. \end{aligned} \tag{4.9}$$

These equations are then solved for $\mathbf{u}_t$ using a trust-region algorithm [85], with the previously applied input vector $\mathbf{u}_{t-1}$ serving as the starting point.

4.1.6 *Simulation results*

The simulation is performed with MATLAB, and the MPC optimization problem is formulated with the YALMIP toolbox [86] and then solved with the Gurobi solver [15]. The controller is tested using a sine with dwell manoeuvre [87], involving a variable longitudinal velocity reference.

Sample time equals $T = 0.01$ s, prediction horizon $N = 10$ and the rest of the parameters are:

- input constraint: $\tilde{\mathbf{u}}_{max} = -\tilde{\mathbf{u}}_{min} = [F_{fx}^c \ F_{fy}^c \ F_{rx}^c]^T$,

$$F_{fx}^c = F_{rx}^c = \frac{3.5}{100} C_x, \quad F_{fy}^c = \frac{5^\circ \pi}{180^\circ} C_y,$$

- input rate constraints: $\Delta \ \tilde{\mathbf{u}}_{max} = -\Delta\tilde{\mathbf{u}}_{min} = [\Delta F_{fx}^c \ \Delta F_{fy}^c \ \Delta F_{rx}^c]^T$,

$$\Delta F_{fx}^c = \Delta F_{rx}^c = \frac{0.75}{100} C_x, \quad \Delta F_{fy}^c = \frac{0.5^\circ \pi}{180^\circ} C_y,$$

•input weight matrix:

$$S = \text{diag}\left(\frac{1}{{F_{fx}^c}^2}, \frac{1}{{F_{fy}^c}^2}, \frac{1}{{F_{rx}^c}^2}\right),$$

•input rate weight matrix:

$$R = \text{diag}\left(\frac{1}{\Delta{F_{fx}^c}^2}, \frac{1}{\Delta{F_{fy}^c}^2}, \frac{1}{\Delta{F_{rx}^c}^2}\right),$$

•tracking error weight matrix:

$$Q = \text{diag}\left(10^5, 10^5\right).$$

Input constraints and input rate constraints are established based on the associated tire slip and slip angle constraints, and these considerations hold true when dealing with small steering angles.

Figure4.3illustrates that the MPC effectively regulates the longitudinal velocity and yaw rate of the vehicle. However, there is a significant increase in the yaw rate tracking error between $t_1 = 4$ s and $t_2 = 5$ s. The optimal tire forces as well as the steering angle and tire slip are shown in Figure 4.4. It is worth noting that the tire forces always remain within the predefined limits due to the imposition of hard constraints. Furthermore, the tire slip values also remain within their limits. Although this is true in this particular case, it may not be the case in a general scenario. This observation is due to the fulfillment of the small steering angle assumption as described in4.1.3. The example shows that the proposed approach is suitable for controlling vehicle dynamics, even if the system does not guarantee perfect tracking of the reference trajectory.

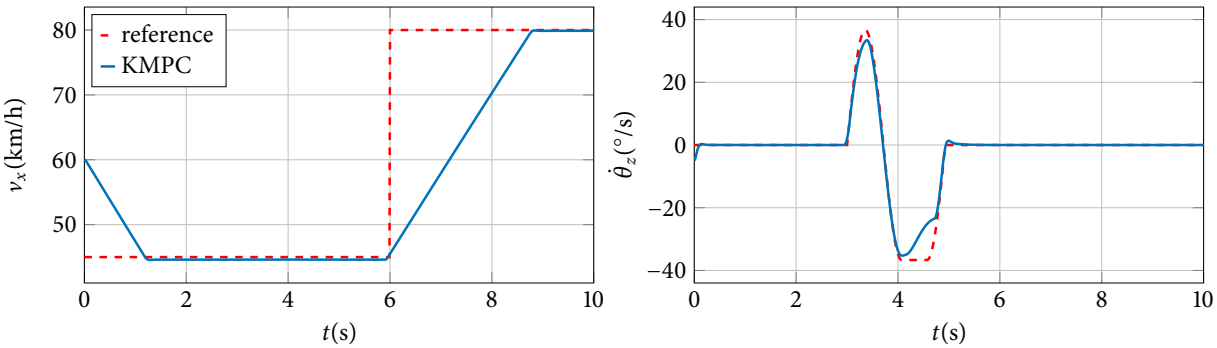

Figure 4.3: Longitudinal velocity v_x and yaw rate $\dot{\theta}_z$ response for the bicycle vehicle model.

4.2 KOOPMAN OPERATOR-BASED CONTROL USING TWO-TRACK MODEL

This section presents the research carried out in [88]. The work described here is based on the bicycle model control described in4.1, but aims to control the two-track vehicle model with front wheel steering. In this method, in addition to the steering angle, the tire slip on all four wheels is also used as a control input, represented as $\mathbf{u} = [\delta_f \; s_{flx} \; s_{frx} \; s_{rlx} \; s_{rrx}]^T$.

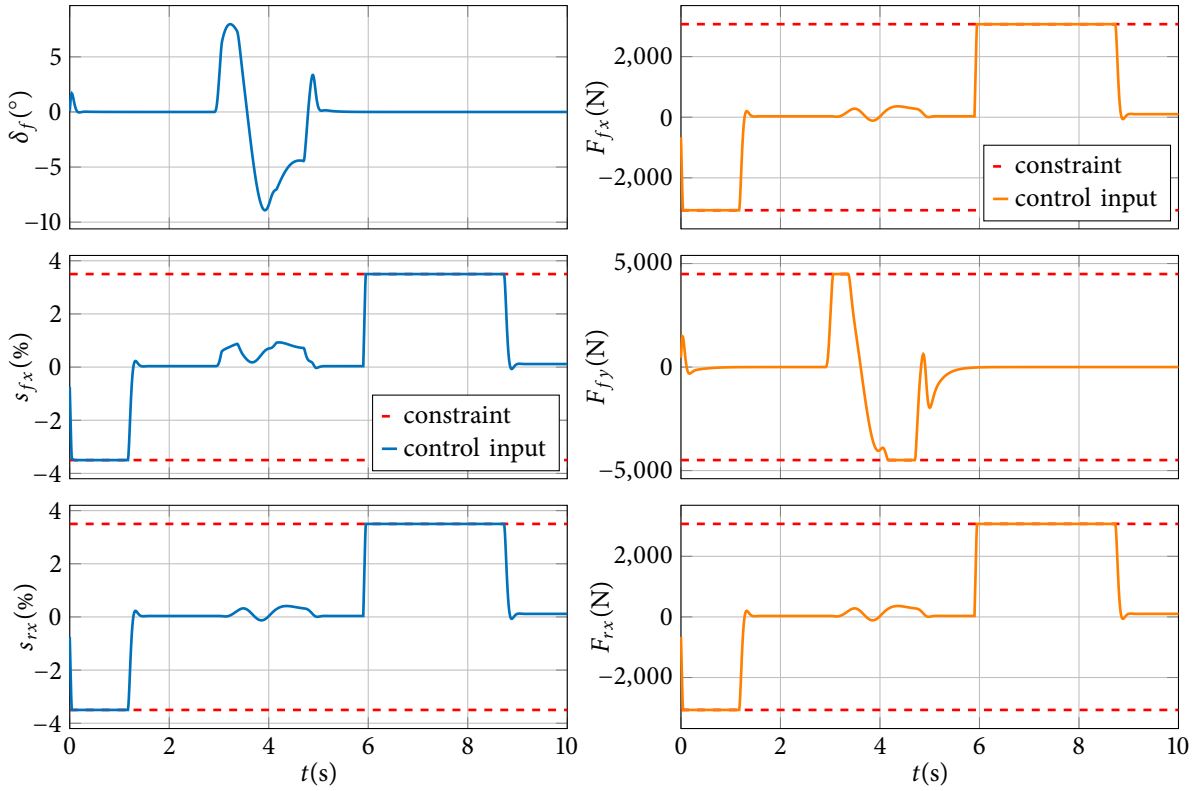

Figure 4.4: Optimal forces obtained from the MPC (right) together with their hard constraints and final control inputs (left) calculated from forces using input mapping described in4.1.5.

4.2.1 *Two-track model without tire model*

Similarly as done in4.1.1for the bicycle model, here the model introduced in2.1.2can be simplified by omitting (2.19) - (2.21):

$$m\dot{v}_x = m\dot{\theta}_z v_y + F_{flx} + F_{frx} + F_{rlx} + F_{rrx} - \frac{1}{2} c_w \rho A_w v_x \sqrt{v_x^2 + v_y^2}, \tag{4.10}$$

$$m\dot{v}_y = -m\dot{\theta}_z v_x + F_{fly} + F_{fry} + F_{rly} + F_{rry} - \frac{1}{2} c_w \rho A_w v_y \sqrt{v_x^2 + v_y^2}, \tag{4.11}$$

$$J_z \ddot{\theta}_z = l_f \left(F_{fly} + F_{fry}\right) - l_r \left(F_{rly} + F_{rry}\right) + w \left(-F_{flx} + F_{frx} - F_{rlx} + F_{rrx}\right). \tag{4.12}$$

In that case, the input vector becomes $\tilde{\mathbf{u}} = [F_{flx}\ F_{fly}\ F_{frx}\ F_{fry}\ F_{rlx}\ F_{rrx}]^T$, making the derivatives linearly dependent on the input. Nevertheless, implementing this model requires a sophisticated low-level force controller. This model is used for vehicle dynamics identification and controller design in sections4.2.2and4.2.4. The model parameters remain identical to those specified for the bicycle model, as detailed in Table4.1.

4.2.2 *Koopman model identification*

The first phase in the development of a vehicle dynamics model involves the selection of a suitable dataset. This dataset is generated by uniformly sampling the system state vector $\mathbf{x}_t = [v_x\ v_y\ \dot{\theta}_z]^T$ within specified ranges: $v_x \in [10,\ 50]$ m/s, $v_y \in [-30,\ 30]$ m/s, $\dot{\theta}_z \in [-10,\ 10]$ rad/s. In addition, the input forces are sampled from the range $[-F_m,\ F_m]$, where F_m is part of the set $\{10, 50, 500, 1000, 5000, 10000\}$ N. Each range comprises $N_s = 25$ sample points, resulting in a

total number of $6 \cdot 25^4$ unique data points in the dataset. To predict the next state $\mathbf{x}_{t+1}$, the nonlinear system is simulated with a sampling time of $T_s = 0.01$ s. It is important to note that varying the number or distribution of samples N_s can affect the prediction accuracy of the resulting model. However, this aspect of the study is not investigated in this context.

Once the dataset is generated, the next step is to determine the specific nonlinear state combinations to be included, i.e. selecting the basis functions, referred to as $\phi(\cdot)$. Similar to the identification of the bicycle model, polynomial basis functions, as shown in equation (4.4), are used. Additionally, this basis is extended to include the slip angle vector $\boldsymbol{\alpha} = [\alpha_{fl}(0°)\ \alpha_{fr}(0°)\ \alpha_{rl}\ \alpha_{rr}]^T$, which is calculated under the assumption that $\delta_f = 0°$. Consequently, the comprehensive basis is represented as follows:

$$B_d = \{P_d,\ \boldsymbol{\alpha}\}. \tag{4.13}$$

The slip angle vector $\boldsymbol{\alpha}$ is included in the basis to simplify the input handling, as described in Section4.2.4. To identify the degree d that provides optimal accuracy, an averaged RMSE, as defined in equation (4.5), is calculated over multiple trajectories. This evaluation includes 15000 trajectories, each comprising up to 50 samples with a sample time of $T_s = 0.01$ s. Both the initial conditions and the input sequences are uniformly sampled from the same distribution used for the training dataset. This analysis is restricted to samples that are within the specified ranges of the state and input vectors, as the accuracy of the RMSE would be compromised outside of these limits. The results are shown in Table4.3.

Table 4.3: Two-track model prediction RMSE for polynomial basis functions of different orders.

Order	1	2	3	4	5	6	7	8
RMSE [%]	26.50	12.24	4.44	2.37	1.50	1.25	1.14	1.11

A polynomial basis function of order $d = 5$ is chosen for the simulation, based on its average RMSE of 1.5006% and a basis cardinality of $n_\Phi = 60$. This decision is made because the RMSE for $d \geq 5$ shows negligible fluctuations. The simulation updates its predictions every $\Delta t = 0.5$ s, which is consistent with the timing of the MPC execution, supporting the use of this model for MPC design in Section4.2.4. Figure4.5shows a specific simulation scenario where each system received a randomly chosen input signal that is uniformly distributed in the range $[-10000,\ 10000]$ N, while the initial state is set randomly to $\mathbf{x}_0 = [32.5698\ \ -9.1526\ 2.5326]^T$. In this case, the RMSE is 0.3163% for the Koopman system, 58.0229% for the linearized system around $\mathbf{x}_0$ and 20.7010% for the system with iterative linearization. The Koopman operator shows superior approximation accuracy compared to the other methods tested, both the iteratively linearized model and the model linearized around $\mathbf{x}_0$.

4.2.3 *Linear time-variant model*

In this section, linear time-variant (LTV) model is used for comparison with Koopman based model. To define it, lets start by defining the nonlinear model (2.16) - (2.18) with linear tire model described in2.2.2and tire slips used as inputs, i.e. $\mathbf{u} = [\delta_f\ s_{flx}\ s_{frx}\ s_{rlx}\ s_{rrx}]^T$. This model can be written in compact form as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)). \tag{4.14}$$

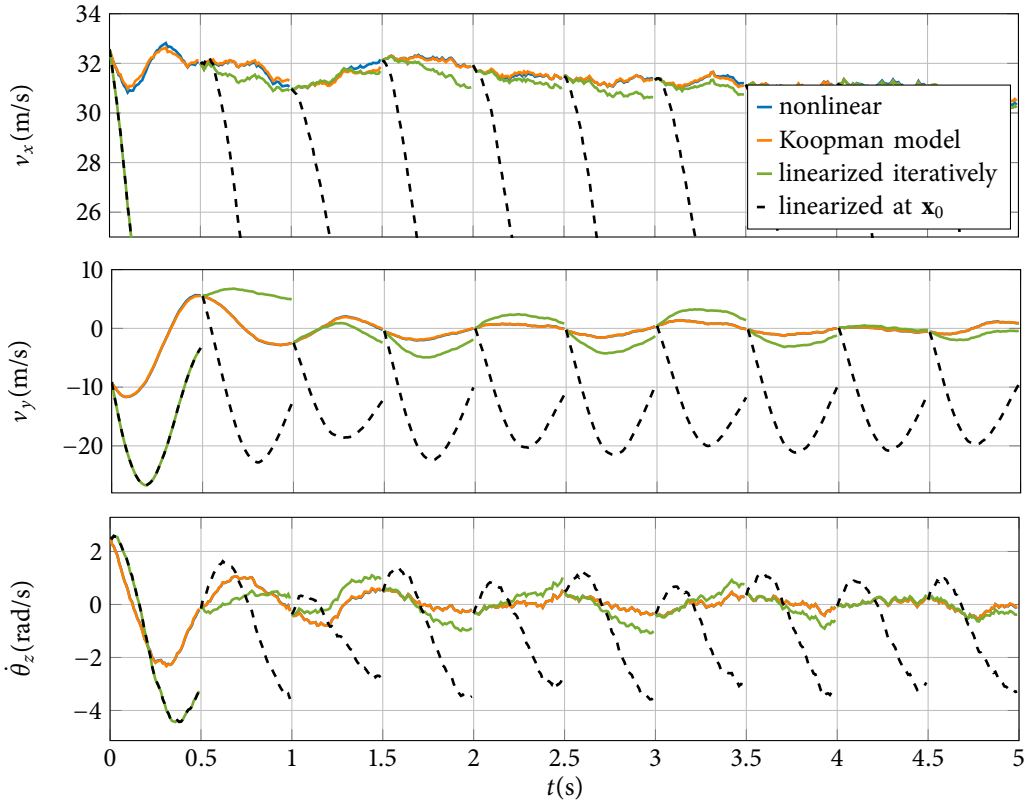

Figure 4.5: Comparison between Koopman model, iteratively linearized model and the model linearized at the origin. Approximations are done for the two-track vehicle model.

The discrete-time version of this model is given by:

$$\mathbf{x}_t = \mathbf{f}_d(\mathbf{x}_t, \mathbf{u}_t). \tag{4.15}$$

LTV model is then formulated similarly to the one proposed in [89]. The prediction of the state vector at time k is

$$\mathbf{x}_{k+1} = A_t\mathbf{x}_k + B_t\mathbf{u}_k + \mathbf{d}_k, \quad k = t, \dots, t+N-1. \tag{4.16}$$

In this representation, the discrete-time system defined by the matrices A_t and B_t is derived by linearizing the equation (4.14) around the current state $\mathbf{x}_t$ and the previous input $\mathbf{u}_{t-1}$, after which the resulting linear system is discretized using bilinear transform. Additionally, $\mathbf{d}_t$ represents the deviation of the nonlinear model's steady-state response from that of the LTV model:

$$\begin{aligned} \mathbf{d}_k &= \hat{\mathbf{x}}_{k+1} - A_t\hat{\mathbf{x}}_k - B_t\mathbf{u}_{t-1}, \\ \hat{\mathbf{x}}_{k+1} &= \mathbf{f}_d(\hat{\mathbf{x}}_k, \mathbf{u}_{t-1}), \quad \hat{\mathbf{x}}_t = \mathbf{x}_t. \end{aligned} \tag{4.17}$$

A similar method is described in more details in Chapter5.

4.2.4 *Koopman MPC design*

The KMPC problem is structured as follows:

$$\begin{aligned}
\min_{\bar{\mathbf{U}}_t, \mathcal{E}_t} \quad & J(\mathbf{z}_t, \bar{\mathbf{u}}_{t-1}, \mathbf{Y}_t^{ref}, \bar{\mathbf{U}}_t, \mathcal{E}_t) \\
\text{s.t.} \quad & \mathbf{z}_{k+1} = A\mathbf{z}_k + B\bar{\mathbf{u}}_k, \quad k = t, \dots, t+N-1, \\
& \mathbf{y}_k = C\mathbf{z}_k, \\
& E_k\bar{\mathbf{u}}_k \leq \mathbf{e}_k, \\
& F_k\mathbf{z}_{k+1} \leq \mathbf{f}_k + \varepsilon_k, \\
& \varepsilon_k \geq 0, \\
& \mathbf{z}_t = \Phi(\mathbf{x}(t)), \\
& \bar{\mathbf{u}}_{t-1} = \bar{\mathbf{u}}(t-1).
\end{aligned} \tag{4.18}$$

Here, the notation aligns with that in (4.6) with $\mathcal{E}_t = [\varepsilon_t, \varepsilon_{t+1}, \dots \varepsilon_{t+N-1}]$ being the slack vector sequence. Cost function equals

$$J(\mathbf{z}_t, \bar{\mathbf{u}}_{t-1}, \mathbf{Y}_t^{ref}, \bar{\mathbf{U}}_t, \mathcal{E}_t) = \sum_{k=t}^{t+N-1} \|\bar{\mathbf{u}}_k\|_S^2 + \|\Delta\bar{\mathbf{u}}_k\|_R^2 + \sum_{k=t}^{t+N} \|\mathbf{y}_k - \mathbf{y}_k^{ref}\|_Q^2 + \sum_{k=t}^{t+N-1} p\|\varepsilon_k\|_I^2. \tag{4.19}$$

In this context, weight matrices are as defined in (4.7), with the addition of $p \geq 0$ representing the slack weight and I indicating the identity matrix of the corresponding size. The term $\Delta\bar{\mathbf{u}}_k$ denotes the input rate.

The text below details two critical observations that play a key role in shaping the state and input constraints for the given design:

1. *Force transformation*: To ensure that the state propagation depends linearly on the inputs, the input vector is chosen as $\bar{\mathbf{u}} = [F_{flx}\ F_{fly}\ F_{frx}\ F_{fry}\ F_{rlx}\ F_{rrx}]^T$. With a precise prediction of the steering angle, this input vector and its rate can be transformed from the coordinate system of the vehicle to that of each individual wheel. The transformed input vector $\mathbf{u}^w = [F_{flx}^w\ F_{fly}^w\ F_{frx}^w\ F_{fry}^w\ F_{rlx}^w\ F_{rrx}^w]^T$ results from the following equation:

$$\begin{aligned}
\mathbf{u}_k^w &= T(\delta_k)\bar{\mathbf{u}}_k, \\
\Delta\mathbf{u}_k^w &= T(\delta_k)\Delta\bar{\mathbf{u}}_k + \Delta\delta_k \Delta T(\delta_k)\bar{\mathbf{u}}_k,
\end{aligned} \tag{4.20}$$

   where $T(\delta_k) = \mathrm{diag}(T_0(\delta_k), T_0(\delta_k), I_2)$ and $\Delta T(\delta_k) = \mathrm{diag}(\Delta T_0(\delta_k), \Delta T_0(\delta_k), I_2)$ are block diagonal matrices composed of rotation matrices

$$T_0(\delta_k) = \begin{bmatrix} \cos\delta_k & \sin\delta_k \\ -\sin\delta_k & \cos\delta_k \end{bmatrix}, \quad \Delta T_0(\delta_k) = \begin{bmatrix} -\sin\delta_k & \cos\delta_k \\ -\cos\delta_k & -\sin\delta_k \end{bmatrix} \tag{4.21}$$

   and I_2 is the 2×2 identity matrix.

2. *Slip angle identity*: According to [90], the slip angles of the front wheel tires, as shown in (2.25), can be expressed in the following manner:

$$\alpha_{f\star}(\delta_f) = \arctan\left(\frac{v_{f\star y}}{v_{f\star x}}\right) - \delta_f = \alpha_{f\star}(0^\circ) - \delta_f. \tag{4.22}$$

   Note that the slip angles $\alpha_{f\star}(0^\circ)$ are already included in the basis outlined in (4.13) and can therefore be easily extracted from the vector $\mathbf{z}_t$.

The matrices E_k and F_k as well as the vectors $\mathbf{e}_k$ and $\mathbf{f}_k$ can be uniquely reconstructed using the following constraints. The constraints for the input and its rate of change are

$$\begin{aligned} \bar{\mathbf{u}}_{min} \leq T(\delta_k)\bar{\mathbf{u}}_k \leq \bar{\mathbf{u}}_{max}, \\ \Delta\bar{\mathbf{u}}_{min} \leq T(\delta_k)\Delta\bar{\mathbf{u}}_k + \Delta\delta_k \Delta T(\delta_k)\bar{\mathbf{u}}_k \leq \Delta\bar{\mathbf{u}}_{max}, \end{aligned} \tag{4.23}$$

and the constraints for the state are:

$$\boldsymbol{\alpha}_{min} - \boldsymbol{\varepsilon}_k^{min} \leq \boldsymbol{\alpha}_k - \boldsymbol{\Delta}_k \leq \boldsymbol{\alpha}_{max} + \boldsymbol{\varepsilon}_k^{max}, \quad \boldsymbol{\varepsilon}_k^{min} \geq 0 \quad \boldsymbol{\varepsilon}_k^{max} \geq 0, \tag{4.24}$$

where the vector $\boldsymbol{\Delta}k = \begin{bmatrix}\delta_k & \delta_k & 0 & 0\end{bmatrix}^T$ is defined. In addition, two equality constraints are introduced to ensure the feasibility of the lateral forces at any given time:

$$F_{fly}^w = -C_y\left(\alpha_{fl}(0^\circ) - \delta_f\right), \quad F_{fry}^w = -C_y\left(\alpha_{fr}(0^\circ) - \delta_f\right). \tag{4.25}$$

This linear force model is applicable because the constraints in (4.24) keep the tire forces within their linear range.

After determining the optimal solution $\bar{\mathbf{u}}^*$ from (4.18), the control input is derived as $\mathbf{u}^* = \begin{bmatrix}\delta_f & s_{flx}^* & s_{frx}^* & s_{rlx}^* & s_{rrx}^*\end{bmatrix}^T$. Here, each optimal slip value is calculated as $s_{\bullet\star x}^* = F_{\bullet\star x}^{w*}/C_x$.

4.2.5 *Simulation results*

The performance of the proposed controller is evaluated and compared with that of the LTV-MPC, using identical parameters for comparison. Given that the input space for the LTV-MPC is different from that of the KMPC, the input weight matrices are adjusted to $S = 0$ and $R = 0$. The chosen sampling interval is $T_s = 0.01$ s, and the other parameters are set as follows:

- longitudinal force inequality constraints: $F_{x,max}^w = -F_{x,min}^w = 0.05C_x$,
- longitudinal force rate constraints: $\Delta\ F_{x,max}^w = -\Delta F_{x,min}^w = 0.0075C_x$,
- slip angle constraints: $\alpha_{max} = -\alpha_{min} = 5^\circ$,
- tracking error weight matrix and slack weight: $Q = \mathrm{diag}\,(1, 100),\ p = 10^8$.

Force inequality constraints implicitly define slip and slip rate constraints as $s_{max} = -s_{min} = 5$ % and $\Delta s_{max} = -\Delta s_{min} = 0.75$ %. Lateral force equality constraints are set as (4.25).

The simulations are run in MATLAB, where the MPC optimization problems are structured in a dense format and solved with the OSQP solver [16]. By converting to a dense format, the performance of the MPC algorithm becomes independent of the size of the state vector, ensuring that the KMPC maintains consistent execution times regardless of the lifted state-space dimensions [32]. These algorithms are evaluated using the sine with dwell manoeuvre [87] assuming accurate steering angle prediction. Sine with dwell is often used to test active stability systems, so it is expected that the reference yaw rate trajectory will not be reached. Instead, it is important that the vehicle remains stable throughout the manoeuvre, and the performance of the controllers is compared in terms of execution time and closed-loop cost.

Figure4.6shows a comparative analysis of the responses of LTV-MPC and KMPC for a prediction horizon of $N = 5$. From this comparison, it can be seen that KMPC provides better

tracking of the longitudinal velocity reference, while the yaw rate responses of both controllers appear almost identical. KMPC achieves these responses with reduced slip, as depicted in Figure 4.7. This reduction in slip is associated with lower energy consumption. As can be seen in Figure 4.8, although violations of the slip angle limit occur with both controllers, the peak values of these violations are lower with KMPC.

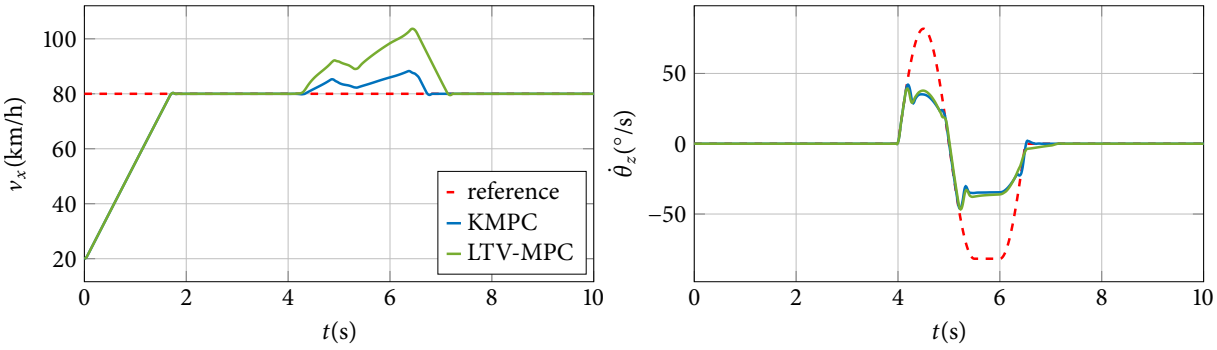

Figure 4.6: Longitudinal velocity v_x and yaw rate $\dot{\theta}_z$ response for $N = 5$ and two-track vehicle model.

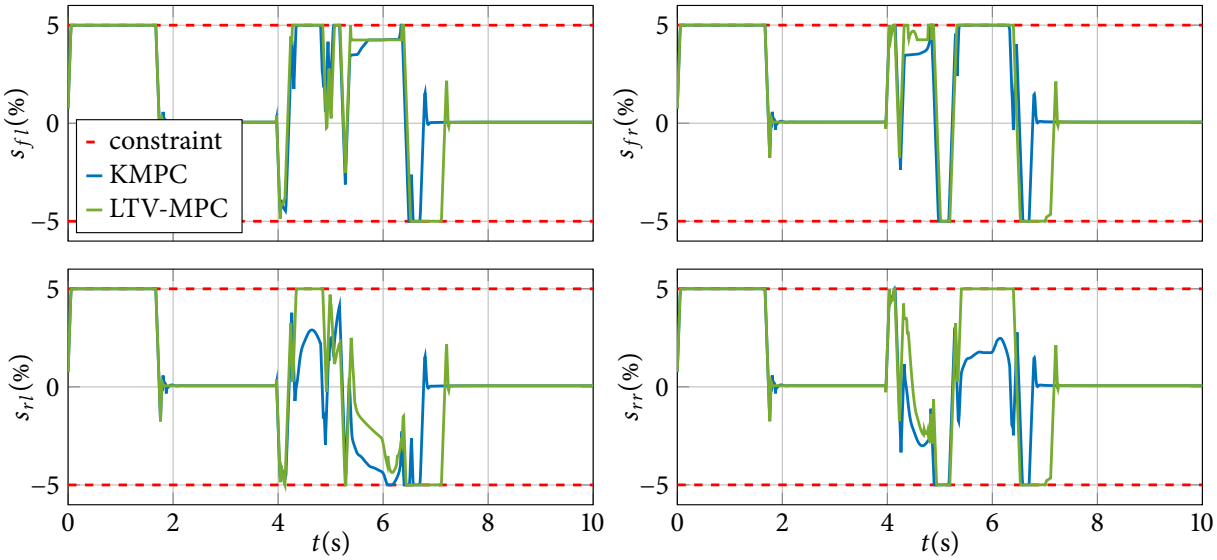

Figure 4.7: Optimal slip ratio for $N = 5$ and two-track vehicle model.

Figure4.9compares the responses for a prediction horizon of $N = 20$. Here, the longitudinal velocity response is similar to that observed for $N = 5$, with the KMPC again performing better, whereas the yaw rate response shows better results with the LTV-MPC, although at the cost of violating soft constraints related to slip angle. The optimal slip ratio is illustrated in Figure4.10. These graphs reveals that, unlike the LTV-MPC, the KMPC rarely reaches slip ratio saturation, a behaviour consistent with the results for $N = 5$. Nevertheless, after the manoeuvre, at $t_1 = 6.5$ s, both controllers continue to demand non-zero slip on the left front and rear wheels, even though the vehicle maintains a constant speed. This results in the left front wheel (s_{fl}) trying to accelerate while the left rear wheel (s_{rl}) tries to brake, a situation that is neither efficient nor safe, highlighting a potential flaw in the design of the control algorithm. The slip angle response depicted in Figure 4.11shows that the KMPC keeps the slip angle within the limits, while the LTV-MPC still violates these constraints, though to a lesser extent compared to the scenario with $N = 5$.

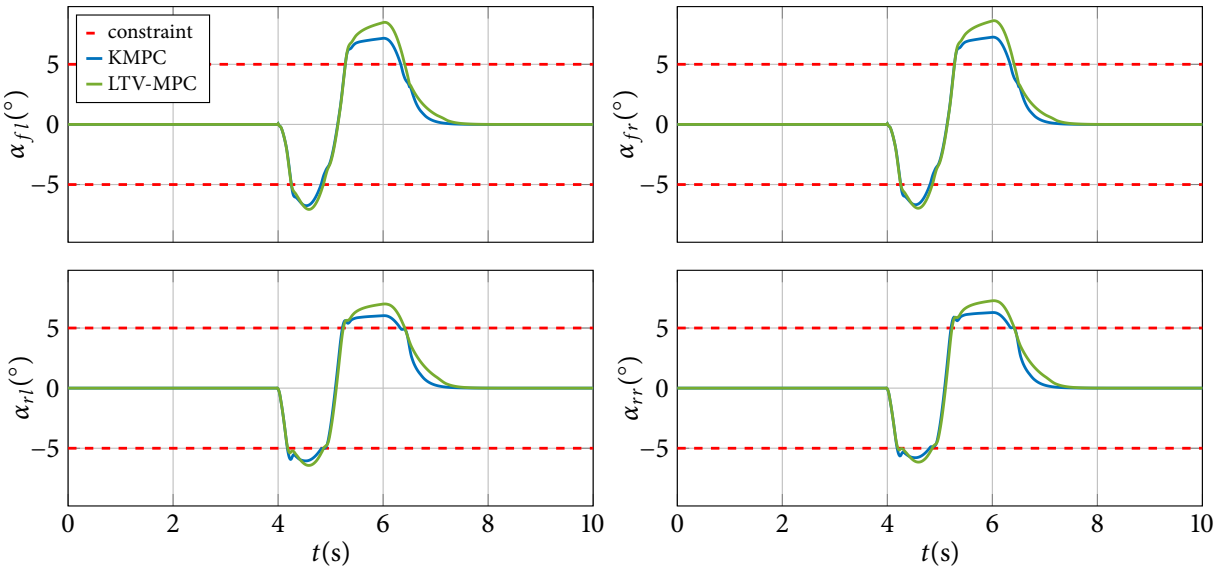

Figure 4.8: Slip angle response for $N = 5$ and two-track vehicle model.

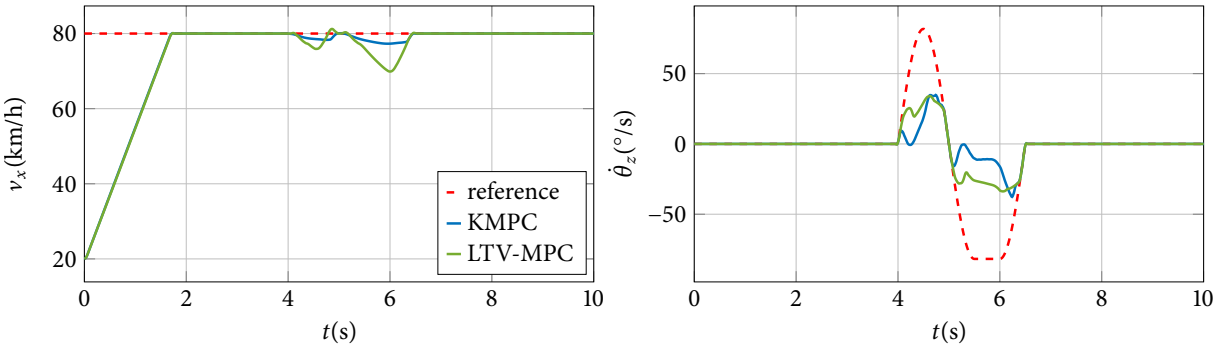

Figure 4.9: Longitudinal velocity v_x and yaw rate $\dot{\theta}_z$ response for $N = 20$ and two-track vehicle model.

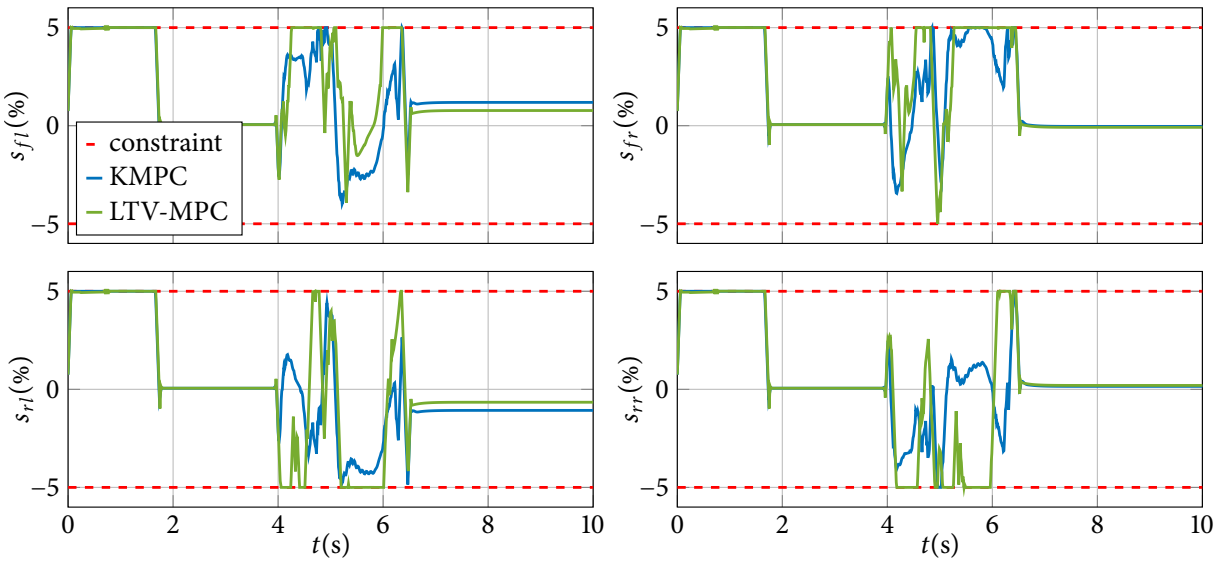

Figure 4.10: Optimal slip ratio for $N = 20$ and two-track vehicle model.

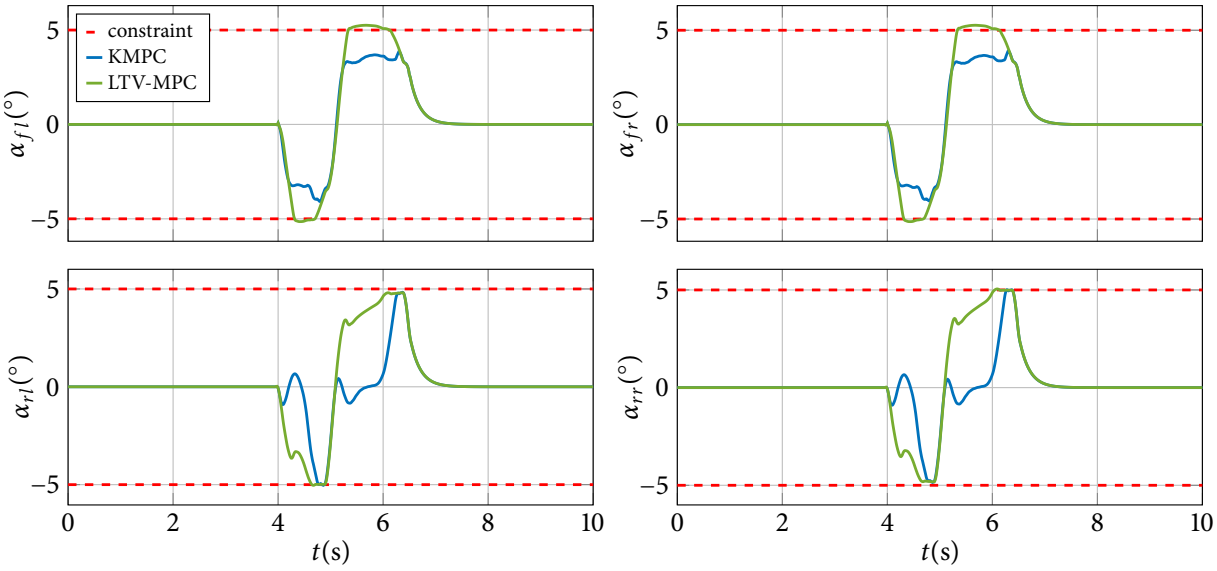

Figure 4.11: Slip angle response for $N = 20$ and two-track vehicle model.

Table4.4displays the normalized closed-loop costs for each of the four test scenarios. In these comparisons, the closed-loop costs associated with the KMPC are consistently lower than those of the LTV-MPC across both prediction horizons. In addition, a significant decrease in costs can be observed with the extension of the prediction horizon. This reduction is anticipated, as it reflects the controller's improved prediction of future events.

Table 4.4: Normalized closed-loop cost

Prediction horizon	LTV-MPC	KMPC
$N = 5$	1	0.9004
$N = 20$	0.2506	0.0151

Table4.5presents the execution times recorded for each algorithm at every time step of the simulation, totaling 10,000 samples[1]. As indicated in the table, the KMPC not only achieves a lower cost but also demonstrates smaller average execution times.

Table 4.5: Algorithm execution times (ms)

Controller	$N = 5$				$N = 20$			
	Mean	Median	Min	Max	Mean	Median	Min	Max
KMPC	7.2	5.8	4.3	655.4	63.2	49.5	46.2	1243.2
LTV-MPC	28.1	15.4	13.5	560.6	93.8	50.9	48.1	1415.9

All the results presented in this section demonstrate the applicability of KMPC for vehicle dynamics control and state its advantages over standard methods such as LTV-MPC. Nevertheless, this method still uses overly simplified vehicle model, which makes it less suitable for real-world applications. This approach is generalized in Chapter5, where a different version of the KMPC algorithm capable of controlling more complex vehicle models is presented.

[1] Experiments were conducted on a computer equipped with an Intel Core i5-7600K CPU at 3.8 GHz, 8 GB of RAM, and running Windows 10.

4.3 SUMMARY

This chapter presents two different applications of the Koopman operator, in particular the EDMD, for modelling and controlling vehicle dynamics using simple vehicle models and longitudinal tire slip as input. Both are simple proof-of-concept approaches proposed in [84] and [88].

The first section shows how the Koopman operator is used for identifying vehicle dynamics. A simplified bicycle model, excluding a tire model, is used to generate the dataset for the EDMD algorithm. This algorithm then approximates a linear model in a higher dimensional state-space and shows promising results that outperform traditional Taylor expansion based linearization models in predicting system trajectories. For further validation, this model is integrated into an MPC design and tested with the original vehicle model, confirming the effectiveness and potential of the method for vehicle motion control.

In the second section, the focus shifts to the identification of a two-track vehicle model using the Koopman operator, integrated with an MPC algorithm. The EDMD method is again used for approximation, and its accuracy is validated against a nonlinear vehicle model. Once sufficient model accuracy is achieved, it is used to develop a linear Koopman MPC algorithm. This algorithm is then compared to the LTV-MPC for prediction horizons of $N = 5$ and $N = 20$, using the sine with dwell algorithm. The Koopman MPC algorithm performs better in all test scenarios in terms of both closed-loop cost and execution times.

These results underline the effectiveness of Koopman MPC in controlling vehicle dynamics and highlight its advantages over conventional approaches such as LTV-MPC. However, it is noteworthy that this method is still based on overly simple vehicle models, which limits its practical applicability in real-world scenarios.

5

Koopman-based predictive torque vectoring

This chapter presents the implementation of the Koopman operator-based model predictive torque vectoring algorithm and is an extension of the work done in [70]. It begins with a literature review that includes a thorough investigation of existing vehicle dynamics control systems. Conventional control strategies such as ABS, ESC and cruise control, which are based on heuristic algorithms or basic mathematical models, are compared with more advanced nonlinear models and control laws. This section highlights the limitations of current methods and introduces the Koopman operator as a potential solution to improve vehicle control under nonlinear conditions. The following section focuses on the application of the Koopman operator for modelling of vehicle dynamics. First, a nonlinear model parameter identification is presented, then the methods for data acquisition and Koopman model identification are introduced. The novelty of the proposed methods is their applicability to more general contexts, as they are not based on specific assumptions, as is the case with the approaches presented in Chapter4. The performance of the obtained models (predictors) is evaluated on both training and test datasets and some of these predictors are selected for the design of control system. In the third section, the linear time-variant MPC, the Koopman MPC and the nonlinear MPC for torque vectoring are explained in detail and their differences and possible problems are mentioned. Although a version of the LTV-MPC is already described in Section4.2.3, here a slightly different and more detailed derivation is presented. After the mathematical description, the given controllers are tested in a high-fidelity simulation environment using four different test cases. The results are documented and discussed, critically analyzing the performance and practicality of Koopman-based predictive torque vectoring systems in comparison to other strategies.

5.1 Existing work

Section2.5provides a brief overview of the vehicle dynamics control systems. In contrast to conventional systems such as ABS, ESC or cruise control, which are usually based on heuristic algorithms or basic mathematical models [9], current algorithms often use nonlinear models and sophisticated control laws [53]. Given that a vehicle is a complex, nonlinear dynamical system with coupled lateral and longitudinal dynamics and nonlinear tire behaviour, accurately modelling its behaviour under various conditions is a challenge. In [91], a comparative analysis of three common modelling techniques is presented: 1. nonlinear physical models, 2. linear physical models and 3. data-driven models. The results show that data-driven models predict vehicle behaviour under standard driving conditions better than models based on physical principles. James and

Anderson's research [92] on longitudinal dynamics models demonstrates that the precision of linear data-driven models can be superior to that of nonlinear physical models under normal driving conditions, while being simpler and more suitable for control system design.

Different control structures, employing both linear and nonlinear vehicle models, can be utilized to implement TV control algorithms. These often include PI controllers [93], PID-based and sliding mode algorithms [94, 95], H_∞ controllers [96] and fuzzy control TV [97, 98]. However, using such controllers can make it difficult to apply constraints to control inputs and states. MPC can help mitigate this problem. Since vehicle dynamics models are generally nonlinear, NMPC provides more accurate predictions of future behaviour [99, 100, 101], but the integration of nonlinear models in MPC can lead to non-convex optimization problems, often posing difficulties due to multiple local optimal points. On the other hand, linear models result in simpler optimization problems that can be solved more efficiently. For this purpose, linear time-invariant (LTI) or linear time-variant (LTV) models are typically used [102, 89], although these may lead to significant prediction inaccuracies, especially over longer prediction horizons [18]. To use a nonlinear model and manage its computational complexity, one approach is to simplify the model, for example by applying the model reduction techniques described in [103]. Alternatively, the Koopman operator can be used to develop a linear model with increased accuracy.

In vehicle dynamics identification, the Koopman operator was first applied to a simple single-track vehicle model in [83]. The study employed two methods for selecting basis functions: EDMD and the eigenfunction approach. While EDMD yielded good results for a model excluding tire nonlinearity, the eigenfunction approach excelled in modelling tire nonlinearities, albeit without considering inputs. This method, enhanced to include inputs, was later adopted in [104] for linear MPC design, demonstrating interesting outcomes. It successfully stabilized a vehicle in a 90-degree drift, a scenario dominated by nonlinearities, yet struggled with basic steering manoeuvres. Further exploration of the EDMD method using a simple bicycle model was done in [84] and was expanded to a two-track model in [88] as detailed in Chapter4. Both applications surpassed LTV models in trajectory prediction. Another application of the Koopman operator for vehicle dynamics control is illustrated in [68], where the Deep-DMD method was applied. Using this technique, a KMPC system was developed that includes the steering angle and the engine throttle as control inputs. The effectiveness of this controller was demonstrated by testing it with an autonomous vehicle model in a sophisticated simulation environment, where it achieved notable tracking performance. This approach was further refined in [66], where a deep learning method called Deep Direct Koopman (DDK) was introduced. In DDK, the Koopman eigenvalues and the input matrix are learned directly, resulting in a model that is represented in an LTI form with a diagonal state transition matrix. This format makes it suitable for systems where the inputs need to be taken into account. Using the DDK model, a linear MPC was developed and deployed to control a vehicle in a high-fidelity simulator. This implementation confirmed the method's ability to efficiently track point-to-point trajectories in real time. DMD was employed for vehicle dynamics identification and lane-keeping control in [105, 106], while [107] investigated the vertical stabilization of off-road vehicles. In [108], an MPC formulation with a vehicle model based on a bilinear Koopman operator for real-time trajectory planning for autonomous driving is presented. Yu et al. [109] used DMD and EDMD methods to build KMPC and use it to control a vehicle in various test scenarios, and concluded that this approach reduces the computational cost compared to NMPC. The eco-driving problem was formulated as a KMPC in [110, 111] to reduce the computational effort

and enable real-time implementation. The KMPC was integrated into a high-fidelity simulator, tested using a selected route scenario and demonstrated the efficacy considering nonlinearities compared to a linear approach. Sassella and associates[112] implemented ABS based on Koopman MPC and compared it with NMPC and an LTV-MPC (referred to as a linear constant-speed model in the paper). Simulation tests showed that the Koopman-based solution is feasible in practise as it leads to a trade-off between tracking performance and computation time. Guo et al. [113] opted for a different approach and used the Koopman operator to capture the intrinsic characteristics of the driver-vehicle system dynamics and develop a shared controller. They discovered that an online update mechanism for the Koopman model is of great importance to capture the adaptive behaviour of the driver in the course of the driver-automation interaction. The work by Chen et al. [114] proposes a safety command governor for autonomous vehicles using a Deep-DMD. This method, which has been validated through extensive testing, outperforms conventional and data-driven models in accurately handling nonlinear vehicle dynamics. It ensures vehicle safety and stability through the integration of control barrier functions (CBFs) and a QP optimization process that significantly improves lateral stability and computational efficiency.

As mentioned in Section2.4.4, since this is an active field of research, new applications of the Koopman operator appear quite frequently. The same applies to applications in the field of vehicle dynamics. However, most of the work referenced in this section mainly focuses on two key concepts: 1) Koopman operator-based models offer a trade-off in between classical linear and nonlinear models when it comes to model complexity and prediction performance, and 2) the KMPC approach reduces computational cost compared to NMPC. This is also the main idea behind the work presented in the rest of the chapter.

5.2 KOOPMAN MODEL IDENTIFICATION

To identify the model, a learning dataset must first be created. Two different approaches can be used for this purpose:

1. simulate several trajectories directly in a high-fidelity simulation environment (this corresponds to conducting experiments when a real vehicle is available);
2. create a nonlinear model based on the data from a high-fidelity simulation environment or a real vehicle and simulate different scenarios with this model.

In this thesis, the second approach is used, representing one of its contributions. A more detailed analysis of the approach is provided in Chapter6.

5.2.1 *Nonlinear vehicle model parameter identification*

In this chapter, the high-fidelity model is used to generate data and calibrate the nonlinear model, which is then treated as a ground-truth model to generate new data for training the linear Koopman model. Therefore, it is necessary to validate the accuracy of the nonlinear model. The high-fidelity simulation software of choice is CarMaker by IPG Automotive. An example vehicle during a manoeuvre in CarMaker is shown in Figure5.1.

The structure of the nonlinear vehicle dynamics model is as described in2.1.2, with piecewise linear tire model and tire force coupling, but without alternative slip formulation and wheel

Figure 5.1: Vehicle in the Car Maker simulation software.

viscous friction. For the sake of clarity, lets write this particular system compactly as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)). \tag{5.1}$$

State and input vectors are $\mathbf{x} = [v_x\ v_y\ \dot{\theta}_z\ \omega_{fl}\ \omega_{fr}\ \omega_{rl}\ \omega_{rr}]^T$ and $\mathbf{u} = [\delta_{sw}\ T_{fl}\ T_{fr}\ T_{rl}\ T_{rr}]^T$, and the parameters of the model are given in Table5.1. In this input vector formulation, δ_{sw} is the steering wheel angle, which is equal to

$$\delta_{sw} = i_{sw}\delta_f. \tag{5.2}$$

Most of the parameters in the table are taken directly from CarMaker, with the exception of the longitudinal tire stiffness C_{xf} and C_{xr} on the front and rear axle, the lateral tire stiffness C_{yf} and C_{yr} on the front and rear axle, the rolling resistance f_f and f_r on the front and rear wheels and the steering wheel ratio i_{sw}, which are determined using the MATLAB System Identification Toolbox on the basis of numerous experiments recorded from CarMaker. [1] The comparison between the CarMaker data and the derived nonlinear model is shown in Figure5.2.

5.2.2 *Data collection*

The dataset is created by randomly sampling the initial state vector and the input vector sequence of p steps from a uniform distribution, after which the system (5.1) is simulated starting from the sampled initial states and excited by the sampled input sequences. The set of initial states $\{v_{x0},\ v_{y0},\ \dot{\theta}_{z0}\}$ is sampled from the given intervals: $v_{x0} \in [20,\ 150]$ km/h, $v_{y0} \in [-45,\ 45]$ km/h, $\dot{\theta}_{z0} \in [-45,\ 45]$ °/s, while the wheels are assumed to be free rolling, i.e. $\omega_{f\star 0} = v_{x0}/R_{wf}$ and $\omega_{r\star 0} = v_{x0}/R_{wr}$. The input sequence is sampled from intervals $T_{\bullet\star} \in [-500,\ 500]$ Nm and $\delta_{sw} \in [-20i_{sw},\ 20i_{sw}]$ °. In addition, the steering rates are limited to $\Delta\delta_{sw} \in [-4i_{sw},\ 4i_{sw}]$ °. In this way, both the continuity and the smoothness of the sample input trajectory are taken into account. Constraints can be imposed on the torque rate $\Delta T_{\bullet\star}$, but realistic constraints would be greater than the torque limits $T_{\bullet\star}$, as it is assumed that we have an electric powertrain that can reach maximum torque in a few milliseconds. This means that the torque rate limits are always met and are therefore redundant.

The dataset has a sample time of $T_s = 0.05$ s and consists of $N_s = 200000$ sample trajectories that are $p = 15$ steps long, resulting in $N_{total} = 3{\cdot}10^6$ sample points. Of these points, $N_{nl} = 2464528$,

[1] Although some of these parameters can be found in CarMaker for default tire models, they are not correct for the custom piecewise linear model used in this chapter.

Table 5.1: Vehicle model parameters

Parameter	Description	Value	Unit
m	mass of the vehicle	1599.98	kg
l_f	front axle to CoG distance	1.311	m
l_r	rear axle to CoG distance	1.311	m
w	half of the wheel track	0.8035	m
J_z	moment of inertia around yaw axis	2393.665	kg · m²
c_w	drag coefficient	0.37	-
ρ	air density	1.2	kg/m³
A_w	surface exposed to the air flow	2.156	m²
μ	road coefficient of friction	1	-
C_{xf}	front axle longitudinal tire stiffness	$9.0903 \cdot 10^4$	N
C_{yf}	front axle lateral tire stiffness	$3.0419 \cdot 10^4$	N/rad
R_{wf}	front axle effective tire radius	0.336705	m
J_{wf}	front wheel moment of inertia	2.084	kg · m²
f_f	front wheel rolling resistance	0.001	-
C_{xr}	rear axle longitudinal tire stiffness	$1.8831 \cdot 10^5$	N
C_{yr}	rear axle lateral tire stiffness	$2.4165 \cdot 10^5$	N/rad
R_{wr}	rear axle effective tire radius	0.33601	m
J_{wr}	rear wheel moment of inertia	1.985	kg · m²
f_r	rear wheel rolling resistance	0.0143	-
i_{sw}	steering wheel ratio	13.4684	-
$b_{w\bullet}$	front/rear axle viscous friction	0	Nm/(rad/s)

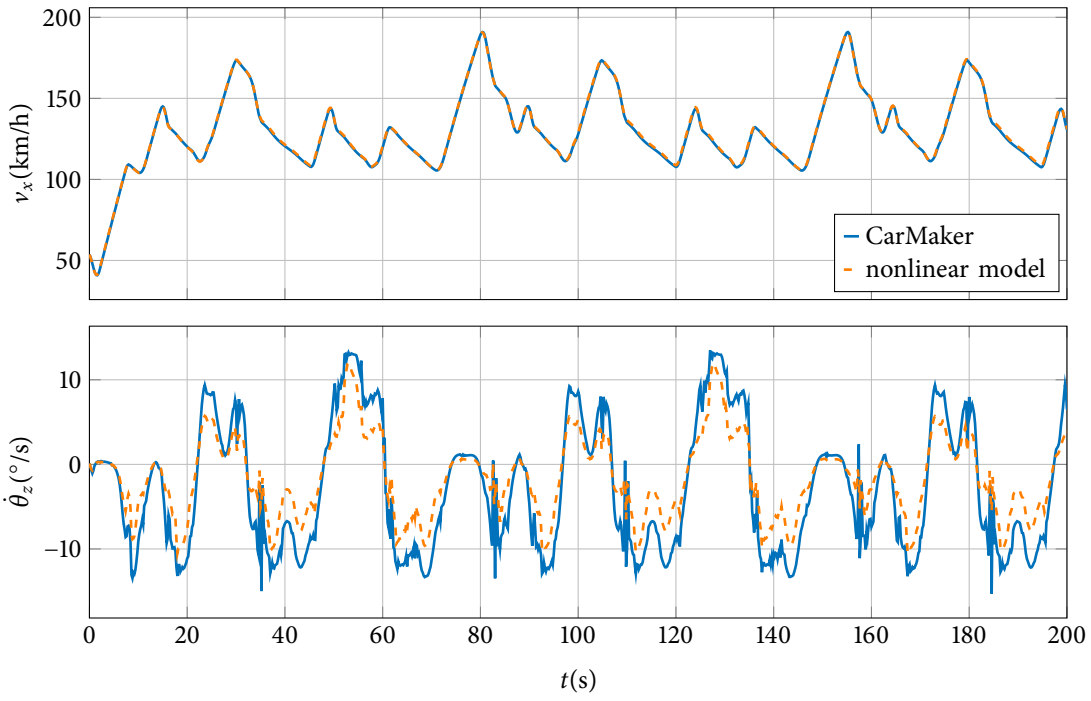

Figure 5.2: CarMaker and nonlinear model comparison.

i.e. about 82.15 % covers nonlinear regions of the tire slip angles. Nonlinear regions refer to the areas outside the linear force range, which means that the forces in the model (2.41a) are saturated.

Figure5.3shows ten different example trajectories (each marked with a different color).

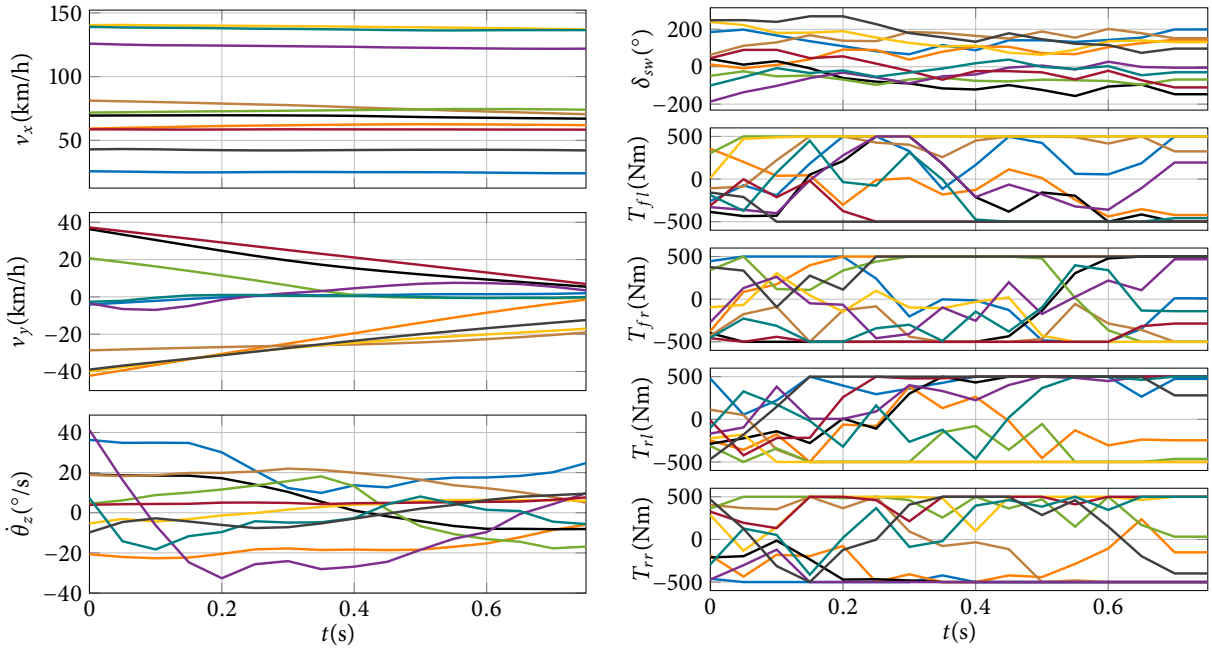

Figure 5.3: Example state (on the left) and input (on the right) trajectories.

5.2.3 *Learning Koopman model*

After the dataset has been created, the basis functions $\phi(\cdot)$ must be selected. As suggested in Chapter4, the polynomial basis P_d was chosen and extended with slip angle vector $\boldsymbol{\alpha} = [\alpha_{fl}\ \alpha_{fr}\ \alpha_{rl}\ \alpha_{rr}]^T$, so that the final basis is as follows:

$$B_d = \{P_d,\ \boldsymbol{\alpha}\}. \tag{5.3}$$

To ensure a linear state-input relationship, two different approaches are proposed in Chapter 4, both assuming a small slip angle, i.e. a linear tire model. Here, a more general approach is proposed. The key idea is to add the steering wheel angle to the state-space of the nonlinear system (5.1), while the steering wheel rate becomes a new input. There is no need to add torques to the modified state-space, as it can be seen from (2.19) that the state propagation already depends linearly on torques. The modified state and input vectors are

$$\begin{aligned} \tilde{\mathbf{x}} &= [v_x\ v_y\ \dot{\theta}_z\ \omega_{fl}\ \omega_{fr}\ \omega_{rl}\ \omega_{rr}\ \delta_{sw}]^T, \\ \tilde{\mathbf{u}} &= [\Delta\delta_{sw}\ T_{fl}\ T_{fr}\ T_{rl}\ T_{rr}]^T. \end{aligned} \tag{5.4}$$

The output is defined as:

$$\tilde{\mathbf{y}} = [v_x\ \dot{\theta}_z\ \delta_{sw}\ \alpha_{fl}\ \alpha_{fr}\ \alpha_{rl}\ \alpha_{rr}]^T. \tag{5.5}$$

To improve the numerical stability of the learning algorithm, the collected data is normalized to the range $[-1,\ 1]$ according to the following rule:

$$\mathbf{x}_{norm} = 2\frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} - 1. \tag{5.6}$$

The learning dataset is divided into a training set of size $N_{train} = 140000$ and a validation set of size $N_{valid} = N_{learn} - N_{train} = 30000$. The test set contains $N_{test} = N_s - N_{learn} = 30000$ samples.

It is used to evaluate the performance of the identified model and contains data samples that are different from those used for learning.

The methods used for learning different Koopman predictors are those described in Chapter 3, with the addition of multiple step prediction minimization EDMD (denoted as EDMD-MS). EDMD-MS works in the same way as the E²DMD-MS method, but does not reduce the state-space, i.e. the reduction equation (3.29) becomes $\mathbf{w}_t = \mathbf{z}_t$.

The methods have the following parameters:

- **EDMD**: trained exclusively using the training data (no consideration of validation data, as this is not part of the standard algorithm),
- **Deep-DMD**: encoder with a total of $L = 5$ layers, where $n_1 = 32$, $n_2 = 64$, $n_3 = 128$, $n_4 = 64$ and $n_5 = n_{\Phi_e} - n_x$; activation function for the layers n_1 to n_4 is the rectified linear unit (ReLU) and for n_5 the hyperbolic tangent; weights $\alpha_1 = 1$, $\alpha_2 = 1$, $\alpha_3 = 0.3$, $\alpha_4 = 10^{-9}$ and $\alpha_5 = 10^{-9}$, maximum number of training epochs $e_{max} = 100000$, number of prediction steps $p = 15$, violation patience $v_p = 25$, number of evaluation epochs $n_e = 5$, batch size $b_s = 256$ and learning rate $l_r = 10^{-4}$; state included in the lifted state vector;
- **E²DMD-DS**: number of training epochs $e_{max} = 10000$, violation patience $v_p = 100$, randomization epoch number $n_e = 100$, initial temperature $T_{init} = 1000$, temperature reduction coefficient $\alpha_{temp} = 0.99$, and a number of prediction steps $p = 15$; state included in the lifted state vector;
- **E²DMD-MS**: uses the same parameters as Deep-DMD (those that are applicable); state included in the lifted state vector;
- **EDMD-MS**: uses the same parameters as Deep-DMD and E²DMD-MS (those that are applicable);
- **E²DMD-HO**: number of trials $n_t = 250$ and prediction horizon $p = 15$, encoder weight and bias values limited to $[-1,\ 1]$; state included in the lifted state vector.

Since the states are included in the lifted state vectors, an additional implementation detail used in the learning algorithms for Deep-DMD, E²DMD-MS and EDMD-MS is a partially hard-coded output matrix. In other words, the matrix C used to extract the output (5.5) from the lifted state-space vector is defined as follows:

$$C = \left[C_{const}^T,\ C_\alpha^T\right]^T, \tag{5.7}$$

where

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \dots \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \dots \end{bmatrix} \tag{5.8}$$

extracts the states v_x, $\dot{\theta}_z$ and δ_{sw}, while the matrix C_α extracts the slip angle vector α and is discovered by the learning algorithm. This trick embeds existing system knowledge into the learning algorithms and helps to learn predictors with higher prediction performance.

5.2.4 *Predictor comparison*

The performance of the learned models is evaluated on learning (training + validation) and test sets using the MNPE (3.46), averaged over all learning/test set trajectories. The orders of the polynomial basis in (5.3) are set to $d = 2$, $d = 3$ and $d = 4$, resulting in basis function vectors of size $n_\Phi = 49$, $n_\Phi = 169$ and $n_\Phi = 499$, respectively. The reduced basis functions are set to the dimension $n_w = 50$. The resulting errors are shown in Table5.2. For the sake of simplicity, all errors are normalized by the EDMD error ($n_\Phi = 499$), which equals $\text{MNPE}_{\text{EDMD}} = 1.5615$ %.

Table 5.2: Learning and test set normalized error

Model	n_Φ	Learning set error	Test set error
EDMD	49	1.2404	1.231
	169	1.0538	1.0461
	499	1	0.9919
Deep-DMD	-	0.7555	0.7552
EDMD-MS	49	1.1814	1.1737
	169	1.052	1.0435
	499	0.9402	0.9362
E²DMD-DS	169	1.8644	1.8442
	499	1.8249	1.8096
E²DMD-HO	169	1.0574	1.0498
	499	1.2034	1.1949
E²DMD-MS	169	1.0011	0.9965
	499	0.9175	0.913

Based on the information provided, it can be concluded that the EDMD models perform better as the size of the basis function vector increases. This trend supports the hypothesis that larger basis function vectors can capture the system dynamics more accurately, which emphasises the importance of model complexity for prediction accuracy. Furthermore, the EDMD-MS results suggest that multiple step minimization has the potential to improve model performance compared to single step minimization in EDMD.

The Deep-DMD model outperforms all variants of EDMD models, including standard EDMD, EDMD-MS, and variants of E²DMD, on both learning and test sets. This superior performance of Deep-DMD indicates that it is able to leverage deeper representations to capture the system dynamics more effectively than the polynomial basis functions used by the EDMD variants.

Despite the reduction of the basis functions to a dimension of $n_w = 50$, some of the E²DMD models maintain relatively good performance. This reduction likely limits the model complexity and thus possibly prevents overfitting. However, the E²DMD variants show mixed results. E²DMD-DS performs significantly worse compared to the other models, indicating that it is not as effective in capturing system dynamics of vehicle models. Conversely, the variants E²DMD-MS and E²DMD-HO show competitive or improved performance compared to standard EDMD, especially at higher dimensions of the basis function vectors.

In essence, the data indicates a complex interaction between the dimensionality of the basis function vector, model simplification strategies, and the specific configurations of the EDMD framework in accurately modelling the system dynamics. The outstanding performance of Deep-

DMD highlights the potential benefits of integrating deep learning approaches with dynamic mode decomposition, while the different results of E²DMD variants emphasize the crucial role of selecting the appropriate learning algorithm. Moreover, the results presented here do not match those obtained in Section3.5, where simpler models were identified. The conclusion could simply be that not all methods are equally efficient or suitable for identifying all systems. However, this investigation falls outside the scope of this thesis.

Figure5.4shows the comparison of the responses of EDMD, E ²DMD-MS with $n_\Phi = 499$ and Deep-DMD for one of the well-predicted trajectories from the test set. In this example, E²DMD outperforms EDMD, with Deep-DMD performing best, confirming the results presented previously. It is interesting to note that Deep-DMD is the only method able to successfully capture yaw rate changes, while the other methods assumed a nearly linear behaviour. These three models are used for MPC development in the remainder of the chapter.

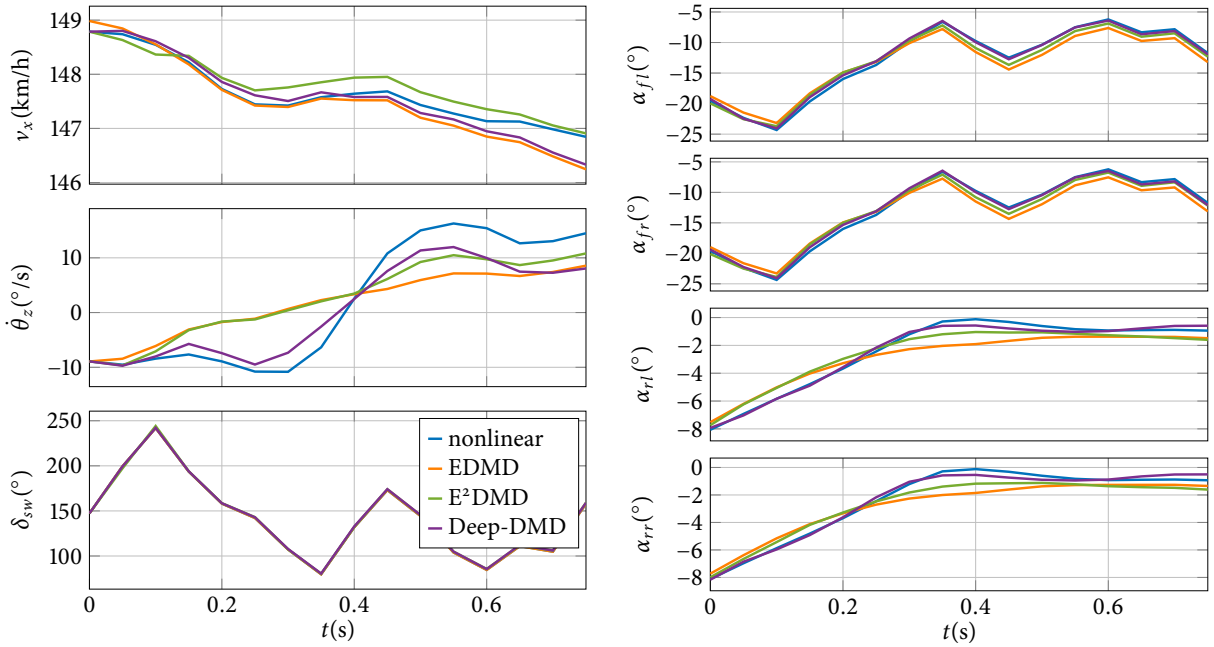

Figure 5.4: Open loop predictions of different Koopman models. Errors of the given models are $\text{MNPE}_{\text{EDMD}}$ = 0.3284 %, $\text{MNPE}_{\text{E²DMD}}$ = 0.2641 % and $\text{MNPE}_{\text{Deep-DMD}}$ = 0.1771 %.

5.3 MPC DESIGN

As mentioned in Section2.3, an MPC uses a model of a system to predict its future behaviour over a finite horizon, choosing the optimal control input sequence to minimize the desired cost function. Since the model (5.1) is nonlinear and we want to have a convex optimization problem as part of predictive controller, a model approximation must be used. In this section, two such MPC versions for torque vectoring applications are derived: LTV-MPC and MPC based on the Koopman operator. Additionally, the NMPC using a full nonlinear vehicle model is described and later used as a benchmark.

5.3.1 *Linear time-variant MPC*

Let us define the discrete-time model of the system (5.1) in the following form:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{f}_d(\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{y}_t &= H\mathbf{x}_t. \end{aligned} \tag{5.9}$$

To formulate the corresponding LTV-MPC optimization problem, there are two possible approaches [18]:

1.create the discrete-time model (5.9) and linearize it;

2.linearize the continuous-time model (5.1) and then discretize the resulting linear model.

The second approach is used in this chapter.

The LTV model is obtained by linearizing (5.1) and is similar to the model in4.2.3. Assume the prediction of the state vector at time $k \in \{t, t+1, \ldots, t+N-1\}$. Then the LTV model can be written as follows:

$$\mathbf{x}_{k+1} = A_{k,t}\mathbf{x}_k + B_{k,t}\mathbf{u}_k + \mathbf{d}_{k,t}, \tag{5.10}$$

where $A_{k,t}$ and $B_{k,t}$ represent discrete-time system matrices. By linearizing the model (5.1) around $\hat{\mathbf{x}}_t$ and $\mathbf{u}_{t-1}$ we obtain continuous-time system matrices [2]

$$A^c_{k,t} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_t, \mathbf{u}_{t-1}}, \quad B^c_{k,t} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\hat{\mathbf{x}}_t, \mathbf{u}_{t-1}}. \tag{5.11}$$

These matrices are transformed using the bilinear transform to obtain the discrete-time system matrices $A_{k,t}$ and $B_{k,t}$ as follows:

$$A_{k,t} = \left(I_{n_x} - \frac{T_s}{2}A^c_{k,t}\right)^{-1}\left(I_{n_x} + \frac{T_s}{2}A^c_{k,t}\right), \quad B_{k,t} = T_s\left(I_{n_x} - \frac{T_s}{2}A^c_{k,t}\right)^{-1}B^c_{k,t}, \tag{5.12}$$

where T_s is the sample time and $I_{n_x} \in \mathbb{R}^{n_x \times n_x}$ identity matrix.

The signal $\mathbf{d}_{k,t}$ represents the deviation of the steady-state response of the LTV model from the nonlinear model:

$$\mathbf{d}_{k,t} = \hat{\mathbf{x}}_{k+1} - A_{k,t}\hat{\mathbf{x}}_k - B_{k,t}\mathbf{u}_{t-1}. \tag{5.13}$$

The prediction of the state trajectory $\hat{\mathbf{x}}_k$ is calculated using the system model, the current state and the previous input value:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{f}_d(\hat{\mathbf{x}}_k, \mathbf{u}_{t-1}), \quad \hat{\mathbf{x}}_t = \mathbf{x}_t. \tag{5.14}$$

The difference to the LTV model (4.16) is the iterative linearization around the predicted state trajectory and not just around the current state. This can lead to a more accurate linear model (if the actual trajectory does not deviate far from the predicted trajectory) than linearization around the current state.

[2] Although not explicitly mentioned, all models in the thesis assume some kind of redefinition of non-differentiable functions to support linearization as proposed in [115].

The model (5.10) enables us to formulate the MPC as a quadratic program given by:

$$\min_{\mathbf{U}_t, \mathcal{E}_t} \quad J(\mathbf{x}_t, \mathbf{u}_{t-1}, \mathbf{Y}_t^{ref}, \mathbf{U}_t, \mathcal{E}_t) \tag{5.15a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = A_{k,t}\mathbf{x}_k + B_{k,t}\mathbf{u}_k + \mathbf{d}_{k,t}, \quad k = t, \dots, t+N-1, \tag{5.15b}$$

$$\mathbf{y}_k = H\mathbf{x}_k, \tag{5.15c}$$

$$\mathbf{u}_{min} \le \mathbf{u}_k \le \mathbf{u}_{max}, \tag{5.15d}$$

$$\Delta\mathbf{u}_{min} \le \Delta\mathbf{u}_k \le \Delta\mathbf{u}_{max}, \tag{5.15e}$$

$$\alpha_{min} - \boldsymbol{\varepsilon}_k^{min} \le \boldsymbol{\alpha}_{k+1} \le \alpha_{max} + \boldsymbol{\varepsilon}_k^{max}, \tag{5.15f}$$

$$\boldsymbol{\varepsilon}_k \ge 0, \tag{5.15g}$$

$$\mathbf{x}_t = \mathbf{x}(t), \tag{5.15h}$$

$$\mathbf{u}_{t-1} = \mathbf{u}(t-1). \tag{5.15i}$$

In the described problem formulation, the variables are as follows: $\mathbf{y}_k = [v_{xk}\ \dot{\theta}_{zk}]^T$ stands for the output vector, the input rate is given by $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$, the slip angle vector is $\boldsymbol{\alpha}_k = [\alpha_{fl,k}\ \alpha_{fr,k}\ \alpha_{rl,k}\ \alpha_{rr,k}]^T$, $\boldsymbol{\varepsilon}_k = \left[\boldsymbol{\varepsilon}_k^{min}\ \boldsymbol{\varepsilon}_k^{max}\right]^T$ is the slack variable vector, $\mathbf{U}_t = [\mathbf{u}_t, \mathbf{u}_{t+1}, \dots \mathbf{u}_{t+N-1}]$ control input sequence, $\mathbf{Y}_t^{ref} = \left[\mathbf{y}_t^{ref}, \mathbf{y}_{t+1}^{ref}, \dots \mathbf{y}_{t+N}^{ref}\right]$ reference sequence and $\mathcal{E}_t = [\boldsymbol{\varepsilon}_t, \boldsymbol{\varepsilon}_{t+1}, \dots \boldsymbol{\varepsilon}_{t+N-1}]$ slack vector sequence.

The cost function (5.15a) can be written as

$$J(\cdot) = J_{ref} + J_{\mathbf{u}} + J_{\Delta\mathbf{u}} + J_{\mathbf{T}} + J_{\boldsymbol{\varepsilon}} \tag{5.16}$$

and consists of five parts:

- reference tracking cost

$$J_{ref} = \sum_{k=t}^{t+N} \|\mathbf{y}_k - \mathbf{y}_k^{ref}\|_Q^2, \quad Q = Q^T \ge 0,$$

- input cost

$$J_{\mathbf{u}} = \sum_{k=t}^{t+N-1} \|\mathbf{u}_k\|_R^2, \quad R = R^T \ge 0,$$

- input rate cost

$$J_{\Delta\mathbf{u}} = \sum_{k=t}^{t+N-1} \|\Delta\mathbf{u}_k\|_{R_\Delta}^2, \quad R_\Delta = R_\Delta^T \ge 0,$$

- front to rear torque difference cost

$$J_{\mathbf{T}} = \sum_{k=t}^{t+N-1} \|\Delta T_k\|_S^2, \quad S \ge 0,$$

- slack variable cost

$$J_{\boldsymbol{\varepsilon}} = \sum_{k=t}^{t+N-1} p\|\varepsilon_k\|_I^2, \quad p \ge 0,$$

with Q, R, R_Δ, which are weight matrices, S torque difference weight, p slack weight and I the identity matrix of corresponding size.

Two remarks are worth mentioning:

1) Torque difference cost: The torque difference is given by $\Delta T_k = T_{fl,k} + T_{fr,k} - T_{rl,k} - T_{rr,k}$. Including the torque difference between the front and rear axles in the cost function prevents the controller from exhibiting behaviour such as maintaining zero torque sum by using $T_{fl} + T_{fr} = -(T_{rl} + T_{rr})$, which was noticed before. Although such torque distribution is feasible, it is also very inefficient and impractical and should therefore be avoided.

2) Slip angle constraints: For implementation purposes, slip angle constraints (5.15f) are defined using the state vector, i.e. as a polytopic constraint

$$E\mathbf{x}_k \leq \boldsymbol{\varepsilon}_k, \tag{5.17}$$

where

$$E = \begin{bmatrix} -\mathbf{g}_{fl,k} + \mathbf{f}_{fl,k}\tan\alpha_{min} \\ -\mathbf{g}_{fr,k} + \mathbf{f}_{fr,k}\tan\alpha_{min} \\ -\mathbf{g}_{rl,k} + \mathbf{f}_{rl,k}\tan\alpha_{min} \\ -\mathbf{g}_{rr,k} + \mathbf{f}_{rr,k}\tan\alpha_{min} \\ \mathbf{g}_{fl,k} - \mathbf{f}_{fl,k}\tan\alpha_{max} \\ \mathbf{g}_{fr,k} - \mathbf{f}_{fr,k}\tan\alpha_{max} \\ \mathbf{g}_{rl,k} - \mathbf{f}_{rl,k}\tan\alpha_{max} \\ \mathbf{g}_{rr,k} - \mathbf{f}_{rr,k}\tan\alpha_{max} \end{bmatrix},$$

$$\mathbf{g}_{fl,k} = \left[-\sin\hat{\delta}_{f,k},\ \cos\hat{\delta}_{f,k},\ l_f\cos\hat{\delta}_{f,k} + w\sin\hat{\delta}_{f,k},\ \mathbf{0}_{1\times4}\right],$$

$$\mathbf{g}_{fr,k} = \left[-\sin\hat{\delta}_{f,k},\ \cos\hat{\delta}_{f,k},\ l_f\cos\hat{\delta}_{f,k} - w\sin\hat{\delta}_{f,k},\ \mathbf{0}_{1\times4}\right],$$

$$\mathbf{g}_{rl,k} = \mathbf{g}_{rr,k} = \left[0,\ 1,\ -l_r,\ \mathbf{0}_{1\times4}\right],$$

$$\mathbf{f}_{fl,k} = \left[\cos\hat{\delta}_{f,k},\ \sin\hat{\delta}_{f,k},\ l_f\sin\hat{\delta}_{f,k} - w\cos\hat{\delta}_{f,k},\ \mathbf{0}_{1\times4}\right],$$

$$\mathbf{f}_{fr,k} = \left[\cos\hat{\delta}_{f,k},\ \sin\hat{\delta}_{f,k},\ l_f\sin\hat{\delta}_{f,k} + w\cos\hat{\delta}_{f,k},\ \mathbf{0}_{1\times4}\right],$$

$$\mathbf{f}_{rl,k} = \left[1,\ 0,\ -w,\ \mathbf{0}_{1\times4}\right],$$

$$\mathbf{f}_{rr,k} = \left[1,\ 0,\ w,\ \mathbf{0}_{1\times4}\right].$$

The matrix E results from the system equations (2.25) and (2.28)-(2.33). The estimate of the front wheel steering angle $\hat{\delta}_{f,k}$ are optimal values that were calculated in the previous optimization cycle. Otherwise, the constraint (5.17) would not be linear.

5.3.2 *Koopman operator-based MPC*

In Chapter3, a discrete-time model approximated by EDMD or Deep-DMD is referred to as (3.4), while the model approximated by one of the E²DMD variants is referred to as (3.26). The reason for this is to emphasise the difference between the basis vector $\mathbf{z}$ and the reduced basis vector $\mathbf{w}$. In this chapter, the dynamics of all Koopman-based models is written as (3.4) for simplicity, regardless of the method.

With that taken into consideration, the KMPC is formulated as a quadratic problem of the following form:

$$\begin{aligned}
\min_{\tilde{\mathbf{U}}_t, \mathcal{E}_t} \quad & J(\mathbf{x}_t, \tilde{\mathbf{u}}_{t-1}, \mathbf{Y}_t^{ref}, \tilde{\mathbf{U}}_t, \mathcal{E}_t) && \text{(5.18a)}\\
\text{s.t.} \quad & \mathbf{z}_{k+1} = A\mathbf{z}_k + B\mathbf{u}_k, \qquad k = t, \dots, t+N-1, && \text{(5.18b)}\\
& \tilde{\mathbf{y}}_k = C\mathbf{z}_k, && \text{(5.18c)}\\
& \mathbf{y}_k = H_{ref}\tilde{\mathbf{y}}_k, && \text{(5.18d)}\\
& \tilde{\mathbf{u}}_{min} \leq \tilde{\mathbf{u}}_k \leq \tilde{\mathbf{u}}_{max}, && \text{(5.18e)}\\
& \Delta\tilde{\mathbf{u}}_{min} \leq \Delta\tilde{\mathbf{u}}_k \leq \Delta\tilde{\mathbf{u}}_{max}, && \text{(5.18f)}\\
& \delta_{min} \leq \delta_{sw,k+1} \leq \delta_{max}, && \text{(5.18g)}\\
& \alpha_{min} - \boldsymbol{\varepsilon}_k^{min} \leq \boldsymbol{\alpha}_{k+1} \leq \alpha_{max} + \boldsymbol{\varepsilon}_k^{max}, && \text{(5.18h)}\\
& \boldsymbol{\varepsilon}_k \geq 0, && \text{(5.18i)}\\
& \mathbf{z}_t = \Phi\left(\tilde{\mathbf{x}}(t)\right), && \text{(5.18j)}\\
& \tilde{\mathbf{u}}_{t-1} = \tilde{\mathbf{u}}(t-1). && \text{(5.18k)}
\end{aligned}$$

The variables in the KMPC are as in the problem (5.15), with the difference that here the extended state vector $\tilde{\mathbf{x}}$, the input vector $\tilde{\mathbf{u}}$ and the output vector $\tilde{\mathbf{y}}$ are defined in (5.4) - (5.5). Since the output vector also contains slip angle, the equations (5.18d), (5.18h) and (5.18g) represent linear constraints without further changes:

$$\begin{aligned}
\mathbf{y}_k &= H_{ref}\tilde{\mathbf{y}}_k = \left[\mathbf{I}_2,\ \mathbf{0}_{2\times 5}\right]\tilde{\mathbf{y}}_k,\\
\alpha_k &= H_{\alpha}\tilde{\mathbf{y}}_k = \left[\mathbf{0}_{4\times 3},\ \mathbf{I}_4\right]\tilde{\mathbf{y}}_k,\\
\delta_{sw,k} &= H_{\delta}\tilde{\mathbf{y}}_k = \left[\mathbf{0}_{1\times 2},\ \mathbf{1},\ \mathbf{0}_{1\times 4}\right]\tilde{\mathbf{y}}_k.
\end{aligned} \tag{5.19}$$

Comparing the formulations (5.17) and (5.19), it becomes clear that the Koopman operator can be used to create a convex optimization problem in a simple and straightforward manner.

5.3.3 *Nonlinear MPC*

To formulate the corresponding optimization problem as a nonlinear program, the equation (5.1) must be discretized. There are two main approaches to achieve this [116]:

1.direct discretization of the nonlinear system;

2. utilization of direct single or multiple shooting methods, often in conjunction with a numerical integration technique like Runge-Kutta or similar methods.

By applying one of these techniques, a discrete-time, nonlinear model is obtained, which is denoted as follows:

$$\begin{aligned}
\mathbf{x}_{t+1} &= \mathbf{f}_{nd}(\mathbf{x}_t, \mathbf{u}_t)\\
\mathbf{y}_t &= H\mathbf{x}_t.
\end{aligned} \tag{5.20}$$

Note that this model does not necessarily have to correspond to (5.9), which depends on the method and/or numerical solver used.

The resulting optimization problem can be expressed as a nonlinear program, given by:

$$\min_{\mathbf{U}_t, \mathcal{E}_t} \quad J(\mathbf{x}_t, \mathbf{u}_{t-1}, \mathbf{Y}_t^{ref}, \mathbf{U}_t, \mathcal{E}_t) \tag{5.21a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}_{nd}(\mathbf{x}_k, \mathbf{u}_k), \quad k = t, \dots, t+N-1, \tag{5.21b}$$

$$\mathbf{y}_k = H\mathbf{x}_k, \tag{5.21c}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max}, \tag{5.21d}$$

$$\Delta\mathbf{u}_{min} \leq \Delta\mathbf{u}_k \leq \Delta\mathbf{u}_{max}, \tag{5.21e}$$

$$\alpha_{min} - \boldsymbol{\varepsilon}_k^{min} \leq \boldsymbol{\alpha}_{k+1} \leq \alpha_{max} + \boldsymbol{\varepsilon}_k^{max}, \tag{5.21f}$$

$$\boldsymbol{\varepsilon}_k \geq 0, \tag{5.21g}$$

$$\mathbf{x}_t = \mathbf{x}(t), \tag{5.21h}$$

$$\mathbf{u}_{t-1} = \mathbf{u}(t-1). \tag{5.21i}$$

The notations are the same as in (5.15). Nonlinear programs like these are commonly solved using interior-point methods or sequential quadratic programming, as mentioned in Section2.3.4.

5.4 SIMULATION RESULTS

In this section, the controllers based on E²DMD-MS with $n_\Phi = 499$ (E²DMD-MPC) and Deep-DMD (Deep-DMD-MPC) from Section5.2.3are compared with the LTV-MPC, the EDMD-MPC proposed by Korda and Mezić in [32] and the NMPC.

The experiments are carried out using MATLAB Simulink and CarMaker simulation software on a computer with an Intel Core i9-10900K CPU running at 3.7 GHz, with 64 GB of RAM and running Windows 10. The LTV-MPC, EDMD-MPC, E²DMD-MPC and Deep-DMD-MPC are solved with the OSQP [16] and NMPC with the FORCESPRO solver [117, 118]. All controllers are tested with the same parameters and their performance was compared. The sample time is $T_s = 0.05$ s, while the remaining parameters are:

- reference tracking weight matrix: $Q = \text{diag}\,(2 \cdot 10^4,\ 10^4)$,
- input weight matrix: $R_u = \text{diag}\,(0, 0.01, 0.01, 0.01, 0.01)$,
- input rate weight matrix: $R_{du} = \text{diag}\,(0, 0.01, 0.01, 0.01, 0.01)$,
- front to rear torque difference weight and slack weight: $S = 1$, $p = 10^8$,
- slip angle constraints: $\alpha_{max} = 3\ ^\circ$, $\alpha_{min} = -3\ ^\circ$,
- torque constraints: $T_{max} = 500$ Nm, $T_{min} = -500$ Nm,
- torque rate constraints: $\Delta\ T_{max} = 500$ Nm, $\Delta T_{min} = -500$ Nm.

In all simulations, the steering angle δ_{sw} was set by the corresponding manoeuvre reference, i.e. the performance of TV for manually steered vehicle is evaluated. This is the reason why steering angle and steering angle rate constraints are omitted. It can be stated that torque rate constraints are equal to torque constraints, effectively eliminating rate constraints. This is not necessarily the case, but led to good results in the following experiments. In a real environment, the dynamics of

the electric motor and the driver should be taken into account, which may require a different set of parameters.

The experiments were conducted with three different tests, each with two different prediction horizons $N = 5$ and $N = 15$. The test cases are:

- batch of experiments using random initial conditions and (semi-)random references (tested using a nonlinear model in Simulink),
- Nürburgring racetrack experiment (tested in CarMaker),
- Nürburgring racetrack low speed experiment(tested in CarMaker),
- Hockenheimring racetrack experiment (tested in CarMaker).

5.4.1 *Batch of randomized test runs*

In this section, the proposed controllers are tested on a set of random trajectories. This is important because the simulation results corresponding to a particular initial condition and manoeuvre are not sufficient to show that the proposed KMPC is indeed effective for a range of operating conditions. The effectiveness of the proposed approach can be better demonstrated by testing with random initial conditions and manoeuvres. However, it is not always realistic to start the test with an arbitrary initial condition and apply a random reference. This is due to the fact that the torque vectoring system is usually activated when driving straight ahead above a certain longitudinal velocity before the driver performs a manoeuvre. That being said, initializing the vehicle with a random lateral velocity and yaw rate (i.e. a random side-slip angle) would require the introduction of an additional controller acting as a driver, and in this case does not show real applicability of the approach. Therefore, the experiments carried out here are based on the three different manoeuvres:

1.step steer

$$\delta_{sw,ref}(t) = \delta_{sw,max}\left(1 - \exp\left(\frac{t-10}{0.1}\right)\right) S(t-10)$$

$S(t)$ stands for a step function. In this way, the equation represents a step steer signal that is filtered by a first-order function to reduce the jerk. The steering is 0 until $t_{start} = 10$ s.

2.sine with dwell (SWD)

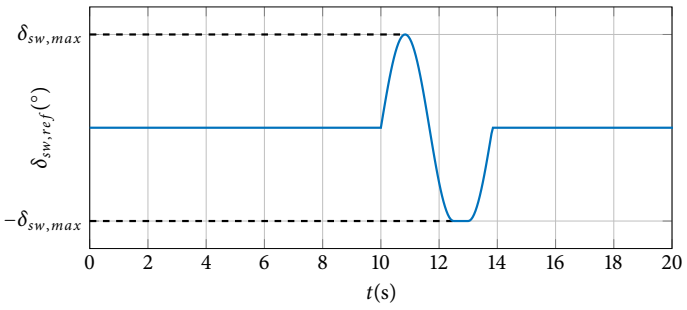

Figure 5.5: Sine with dwell steering signal.

3.sine steer

$$\delta_{sw,ref}(t) = \delta_{sw,max} \sin\left(2\pi f_{ref} t\right)$$

The initial longitudinal velocity was sampled from $v_{x0} \in [20,\ 150]$ km/h, while the lateral velocity and yaw rate were set to $v_{y0} = 0$ km/h, $\dot{\theta}_{z0} = 0\ ^\circ/s$ and the wheels were assumed to be rolling freely, i.e. $\omega_{f\star 0} = v_{x0}/R_{wf}$ and $\omega_{r\star 0} = v_{x0}/R_{wr}$. The longitudinal velocity reference was sampled from $v_{ref} \in [40,\ 150]$ km/h, the predetermined steering wheel angle amplitude from $\delta_{sw,max} \in [-10 i_{sw},\ 10 i_{sw}]\ ^\circ$ and the sine steer frequency from $f_{ref} \in [0.05,\ 1]$ Hz. The yaw rate reference is defined by a kinematic model of the vehicle [55]:

$$\dot{\theta}_{z,ref} = \frac{v_{x,ref}}{l_f + l_r + K_u v_{x,ref}^2} \tan\left(\frac{\delta_{sw,ref}}{i_{sw}}\right), \tag{5.22}$$

where K_u is the understeer gradient

$$K_u = \frac{m\left(l_r C_{yr} - l_f C_{yf}\right)}{\left(l_f + l_r\right) C_{yf} C_{yr}}. \tag{5.23}$$

The simulation time was $T_{sim} = 20$ s and a total of $N_{exp} = 300$ experiments are performed, 100 for each previously mentioned manoeuvre. The Table5.3compares the average normalized closed-loop costs [3], while the Table5.4shows the mean, median, minimum and maximum execution times averaged over N_{exp} experiments. The comparison between the two tables shows revealing contrasts and performance metrics.

Looking first at the closed-loop cost, it is clear that the NMPC outperforms the other methods across both prediction horizons ($N = 5$ and $N = 15$), as it offers the lowest cost and thus has a higher efficiency in handling manoeuvres. Interestingly, although E²DMD-MPC, Deep-DMD-MPC and EDMD-MPC show relatively similar performance in terms of cost terms, the efficiency of LTV-MPC decreases as the prediction horizon increases, as evidenced by the significant cost increase at $N = 15$.

When it comes to execution times, the situation changes. Although NMPC has better cost efficiency, it requires significantly longer execution times. This is particularly evident when the prediction horizon extends to $N = 15$, where its execution time dramatically surpasses that of the other controllers. In contrast, E²DMD-MPC and Deep-DMD-MPC not only offer competitive cost efficiency, but also benefit from significantly lower execution times, indicating a balance between efficiency and computational demand. While EDMD-MPC and LTV-MPC are generally slower than the former two methods, they show a different increase in execution time with the prediction horizon. This applies in particular to the LTV-MPC, which may reflect the underlying inefficiencies under certain conditions.

This analysis highlights the trade-offs between computational efficiency and execution speed of different MPC methods, with NMPC excelling in manoeuvre handling at the cost of computational effort, whereas E²DMD-MPC and Deep-DMD-MPC represent a balanced compromise between the two metrics.

The vehicle responses for one of the manoeuvres are shown in Figure5.6for horizon $N = 5$ and Figure5.7for $N = 15$. The graphs depict what is reported in the Table5.3and discussed in the

[3] The normalized costs are first calculated for each run and then averaged over all experiments.

Table 5.3: Random manoeuvres average normalized closed-loop cost

Prediction horizon	E²DMD-MPC	Deep-DMD-MPC	EDMD-MPC	LTV-MPC	NMPC
$N = 5$	1	0.9940	1.0345	1.5337	0.9484
$N = 15$	1.1325	1.0772	1.1314	4.0914	0.9295

Table 5.4: Random manoeuvres average execution times (ms)

Controller	$N = 5$				$N = 15$			
	Mean	Median	Min	Max	Mean	Median	Min	Max
E²DMD-MPC	3.419	3.232	1.871	10.289	22.616	21.010	18.020	44.334
Deep-DMD-MPC	2.900	2.720	1.880	7.725	21.901	20.560	17.972	40.995
EDMD-MPC	6.829	6.121	2.554	27.113	29.160	26.461	20.186	73.273
LTV-MPC	6.312	5.934	5.632	15.709	34.709	31.396	29.703	152.988
NMPC	34.793	35.199	20.187	43.198	94.549	98.697	50.127	123.504

previous paragraph. NMPC shows superior behaviour for both prediction horizons, while LTV-MPC undoubtedly performs the worst. Koopman-based controllers all exhibit similar behaviour as well as performance degradation with increasing prediction horizon.

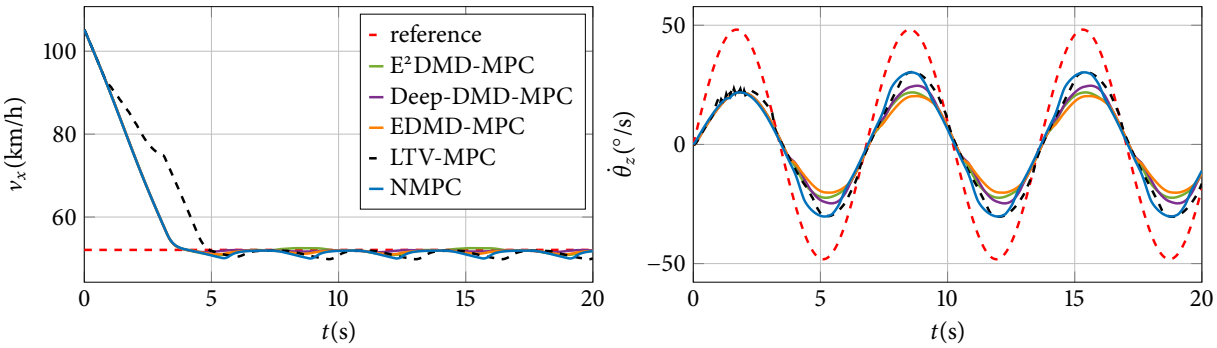

Figure 5.6: Output tracking during one of the sine steer manoeuvres and $N = 5$.

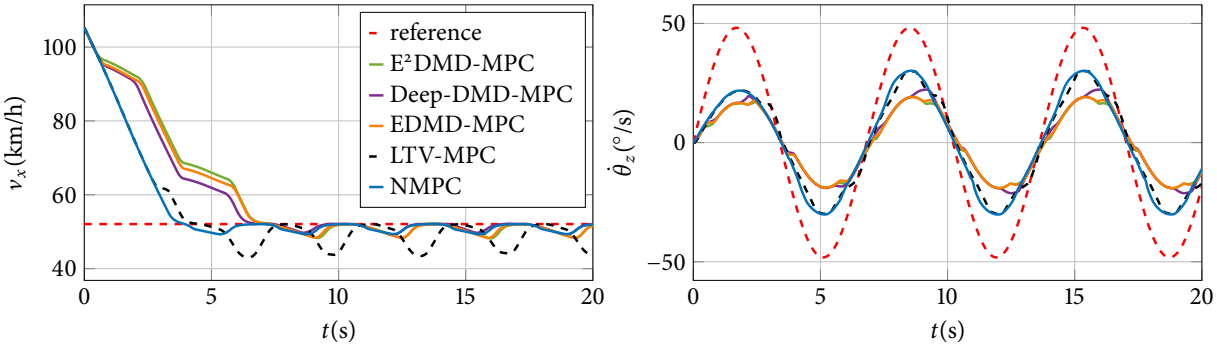

Figure 5.7: Output tracking during one of the sine steer manoeuvres and $N = 15$.

5.4.2 *Nürburgring racetrack experiment*

Studying long-term performance comparisons between the different approaches is useful because it represents a more relevant mode of operation for the (racing) vehicle than a simple short-term manoeuvre. In this section, a reference profile based on the Nürburgring racetrack is used to compare the aforementioned controllers. The total duration of the driving cycle is T_{sim} = 344.75 s, i.e. slightly more than 5.7 minutes. The experiments are performed using CarMaker to demonstrate the applicability of the approach to high-fidelity models which include unmodeled dynamics. Table5.5shows the comparison of the normalised closed-loop costs, with the nominal cost for E²DMD-MPC and $N = 5$ is equal to $2.4759 \cdot 10^8$. The Table5.6shows the execution times.

NMPC stands out for its efficiency in manoeuvre handling, as shown by the lowest normalized closed-loop cost for both prediction horizons. This superior performance is especially visible for $N = 15$. However, this comes at a significant computational cost as execution times increase significantly with the prediction horizon. This trade-off highlights the potential limitations of NMPC in real-time applications, despite its good control capabilities. LTV-MPC, on the other hand, is consistently the least efficient in terms of normalized closed-loop cost, with its performance deteriorating even further as the prediction horizon increases. Combined with a considerable range in execution times, especially at $N = 15$, it is therefore unsuitable for real-time applications.

As in the previous section, the Koopman-based controllers show similar control performance for $N = 5$. When the prediction horizon increases to $N = 15$, the cost of E²DMD-MPC and EDMD-MPC increases significantly, indicating their limitations for longer predictions. On the other hand, Deep-DMD-MPC shows better adaptation to longer horizons, indicated by a reduction in cost, making it a viable option for scenarios requiring longer predictions. Interestingly, this is in contrast to the results obtained for random manoeuvres in the Table5.3.

In terms of computational efficiency, E²DMD-MPC and Deep-DMD-MPC are almost twice as fast as EDMD-MPC and even faster than other controllers due to the smaller size of the state-space. This in turn makes them a balanced compromise among the tested methods, E²DMD-MPC for $N = 5$ and Deep-DMD-MPC for $N = 15$.

Table 5.5: Nürburgring experiment normalized closed-loop cost

Prediction horizon	E²DMD-MPC	Deep-DMD-MPC	EDMD-MPC	LTV-MPC	NMPC
$N = 5$	1	1.0477	1.0059	6.3028	0.8737
$N = 15$	1.7107	0.8336	1.6083	6.6278	0.5981

Table 5.6: Nürburgring experiment execution times (ms)

Controller	$N = 5$				$N = 15$			
	Mean	Median	Min	Max	Mean	Median	Min	Max
E²DMD-MPC	2.647	2.069	1.940	13.118	21.463	19.929	18.414	47.264
Deep-DMD-MPC	2.466	2.075	1.963	21.209	21.237	19.659	18.285	40.304
EDMD-MPC	4.555	2.976	2.672	34.182	23.919	21.951	19.981	71.386
LTV-MPC	7.222	6.701	6.229	82.599	33.972	32.109	30.458	204.227
NMPC	36.256	35.952	4.641	47.087	105.570	104.433	12.519	137.344

Figure5.8and Figure5.11show the system outputs during the entire manoeuvre for both prediction horizons. Since the experiment is slightly longer and it is difficult to visualize all signals at once, shorter time windows are shown. Figures5.9and5.10show the output signals and slip angle responses for $N = 5$ and Figure5.12and Figure5.13for $N = 15$.

These enlarged graphs support the results given in the Table5.5and show that the performance of E²DMD-MPC, EDMD-MPC and LTV-MPC decreases with increasing prediction horizon. This becomes especially clear when looking at the behaviour of the longitudinal velocity in the time interval $t_1 \in [160,\ 175]$ s. E²DMD-MPC and EDMD-MPC exhibit a visible deterioration in performance with increasing prediction horizon, while LTV-MPC performs poorly in both cases. In contrast, Deep-DMD shows an improvement in performance when examining slip angle responses in the time inteval $t_2 \in [155,\ 170]$ s. For $N = 5$, a constraint violation can be observed, which arguably increases the cost value. These violations occur when the controller aims to brake and reduce the longitudinal velocity. As the prediction horizon increases, constraints are no longer violated. This happens several times during the manoeuvre, and at $N = 15$ the violation of constraints is either reduced or completely eliminated.

The performance of NMPC is good for both prediction horizon lengths, and it is difficult to see differences in the graphs.

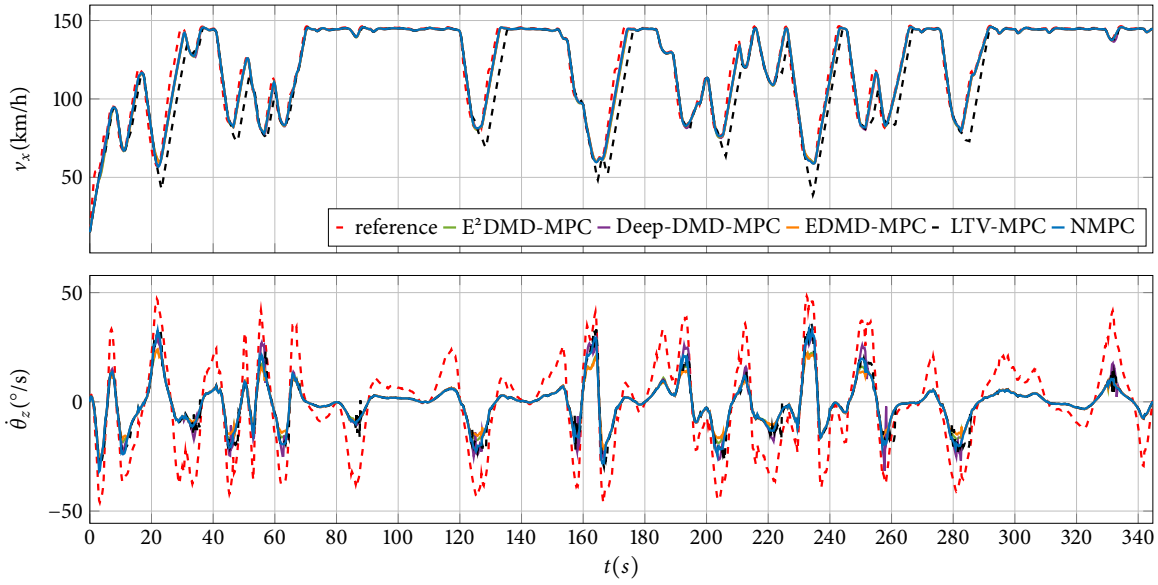

Figure 5.8: Output tracking during Nürburgring experiment with $N = 5$.

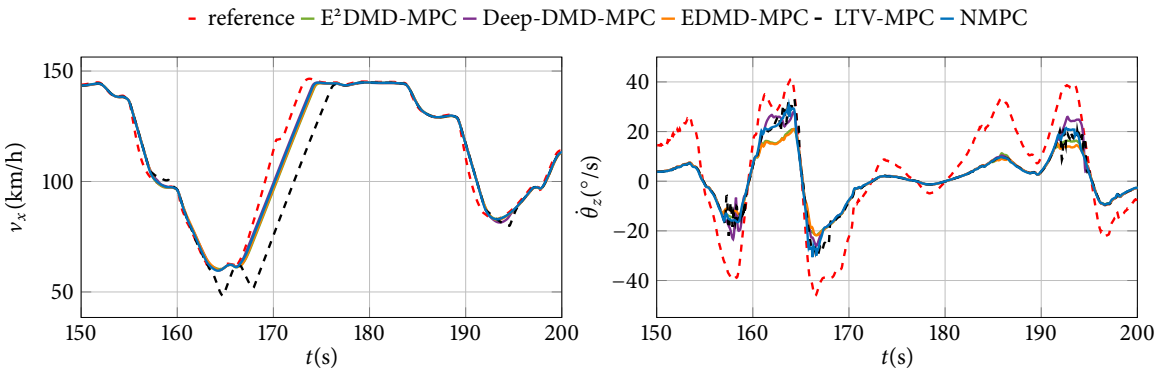

Figure 5.9: Output tracking during Nürburgring experiment with $N = 5$ (shorter time window).

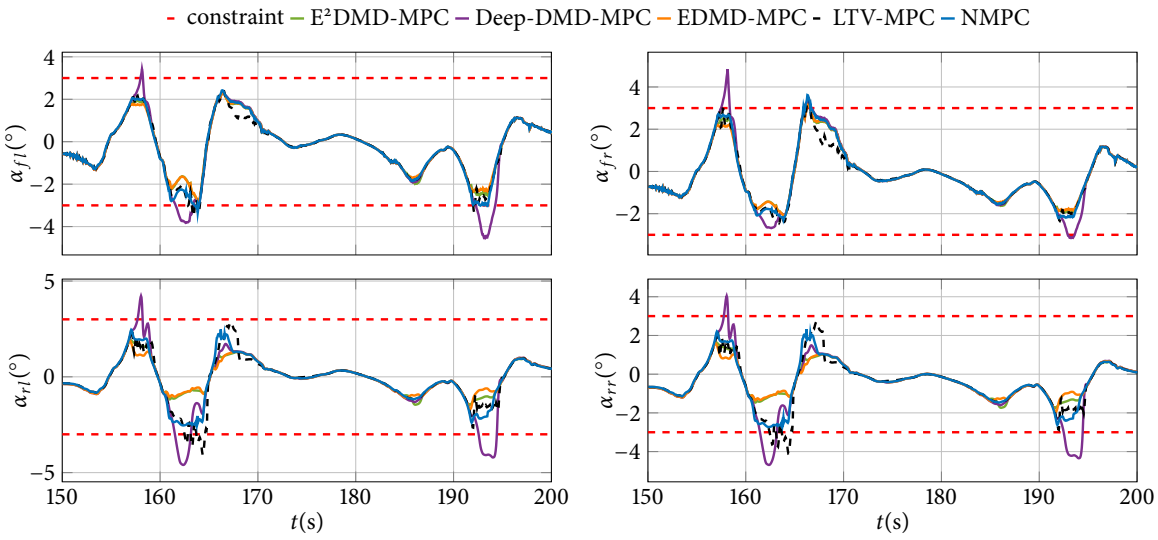

Figure 5.10: Slip angles during Nürburgring experiment with $N = 5$ (shorter time window).

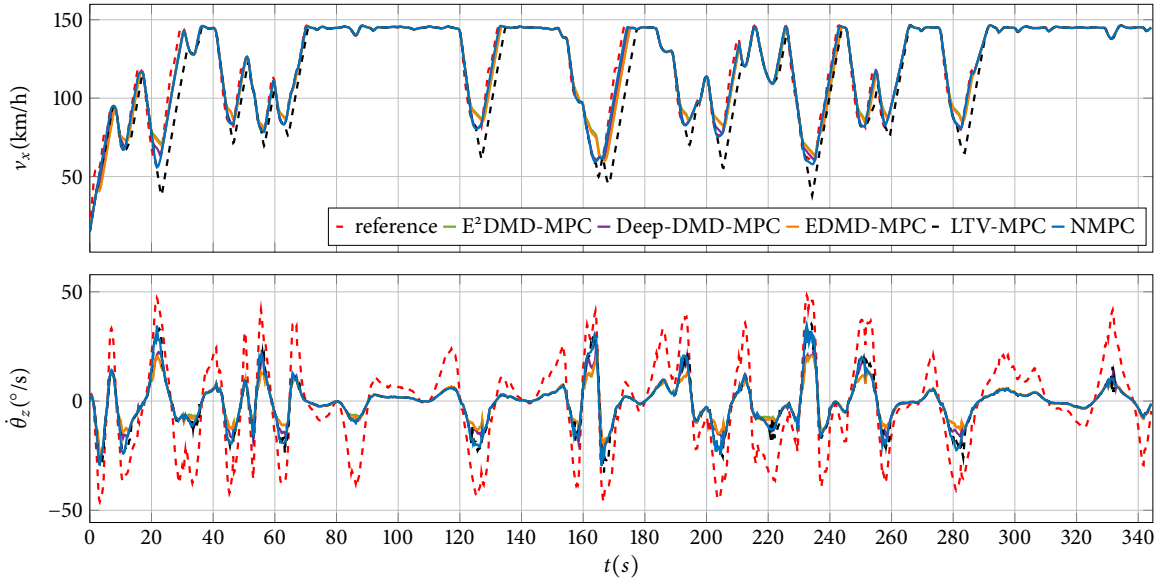

Figure 5.11: Output tracking during Nürburgring experiment with $N = 15$.

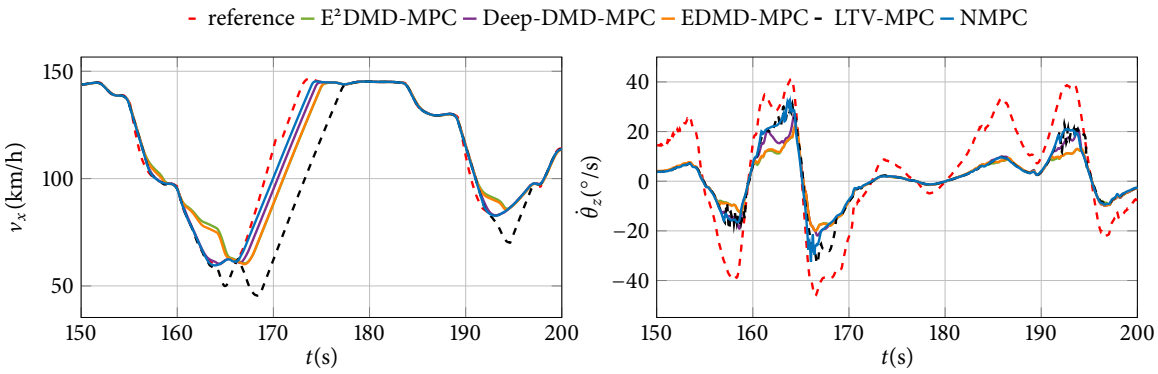

Figure 5.12: Output tracking during Nürburgring experiment with $N = 15$ (shorter time window).

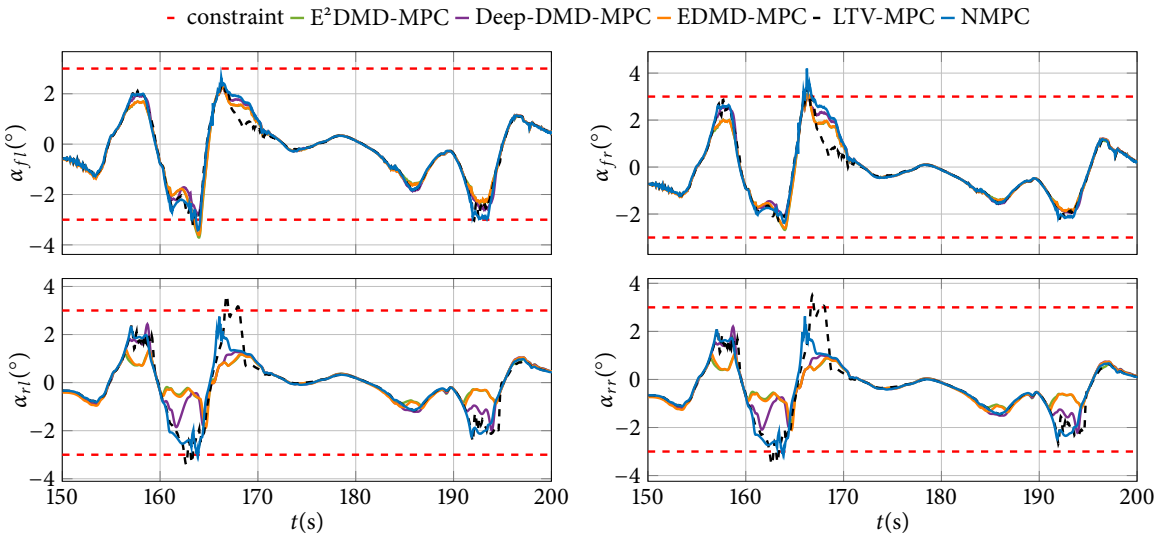

Figure 5.13: Slip angles during Nürburgring experiment with $N = 15$ (shorter time window).

5.4.3 *Nürburgring racetrack low speed experiment*

This section presents a comparison of the already introduced controllers during a low-speed run on a Nürburgring racetrack in CarMaker. Once again, controller prefromance and execution times are evaluated for the prediction horizons $N = 5$ and $N = 15$ and are given in Table5.7. The duration of the experiment is $T_{sim} = 250$ s and the absolute value of the E²DMD-MPC cost is $2.291 \cdot 10^6$.

At the shorter prediction horizon of $N = 5$, the performance of the strategies varies, with E²DMD-MPC and EDMD-MPC having normalized costs close to or slightly above 1, indicating less efficient optimization for short-term predictions. In contrast, Deep-DMD-MPC, LTV-MPC and NMPC exhibit better efficiency and achieve normalized costs below 1, indicating that they are more capable of minimizing costs in the short term. Among these, LTV-MPC stands out as the most efficient strategy for this time horizon. When the prediction horizon extends to $N = 15$, the dynamics between the strategies shifts. The performance of E²DMD-MPC deteriorates slightly, indicating a potential decline in efficiency for longer-term predictions. On the other hand, the performance of Deep-DMD-MPC sees a significant improvement, with the normalized cost decreasing substantially, highlighting its potential for effective long-term prediction. EDMD-MPC shows a slight decrease in normalized cost compared to its performance at $N = 5$, but does not

outperform the other strategies.

Remarkably, both LTV-MPC and NMPC demonstrate a significant improvement in their ability to minimize costs over longer prediction horizon, with NMPC showing the best overall performance. The results for NMPC are consistent with those of the previous two sections, while LTV-MPC performs significantly better at lower speeds and even outperforms the other controllers for $N = 5$. This makes sense as in this case the nonlinear effects are not present and this representation of the model is very similar to the nonlinear one. The differences between LTV-MPC and NMPC probably result from the fact that different optimizers are used for solving them. Deep-DMD-MPC is again the only Koopman-based MPC where the cost decreases with increasing prediction horizon.

The execution times from the Table5.8are similar to those from previous experiments, with NMPC being the slowest and E²DMD-MPC and Deep-DMD-MPC being the fastest control methods.

Table 5.7: Slow Nürburgring experiment normalized closed-loop cost

Prediction horizon	E²DMD-MPC	Deep-DMD-MPC	EDMD-MPC	LTV-MPC	NMPC
$N = 5$	1	0.9847	1.0855	0.9438	0.9534
$N = 15$	1.0696	0.6745	1.0294	0.6358	0.6339

Table 5.8: Slow Nürburgring experiment execution times (ms)

Controller	$N = 5$				$N = 15$			
	Mean	Median	Min	Max	Mean	Median	Min	Max
E²DMD-MPC	2.146	2.063	1.934	5.912	20.718	19.503	18.372	41.225
Deep-DMD-MPC	2.081	2.031	1.904	6.719	20.814	19.434	18.374	36.096
EDMD-MPC	3.233	2.989	2.702	14.445	22.879	21.436	20.031	50.516
LTV-MPC	7.197	6.777	6.243	87.500	32.868	32.620	31.088	104.173
NMPC	33.270	33.097	31.466	40.737	95.730	94.967	89.622	120.256

Figure5.14and Figure5.15depict output responses and are intended to show what the manoeuvre looks like, but don't provide any additional information about the performance.

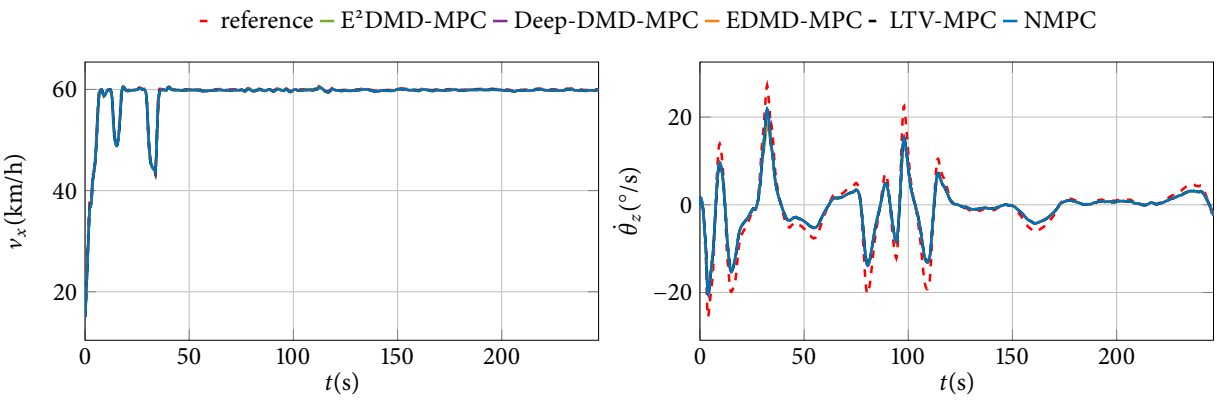

Figure 5.14: Output tracking during slow Nürburgring experiment with $N = 5$.

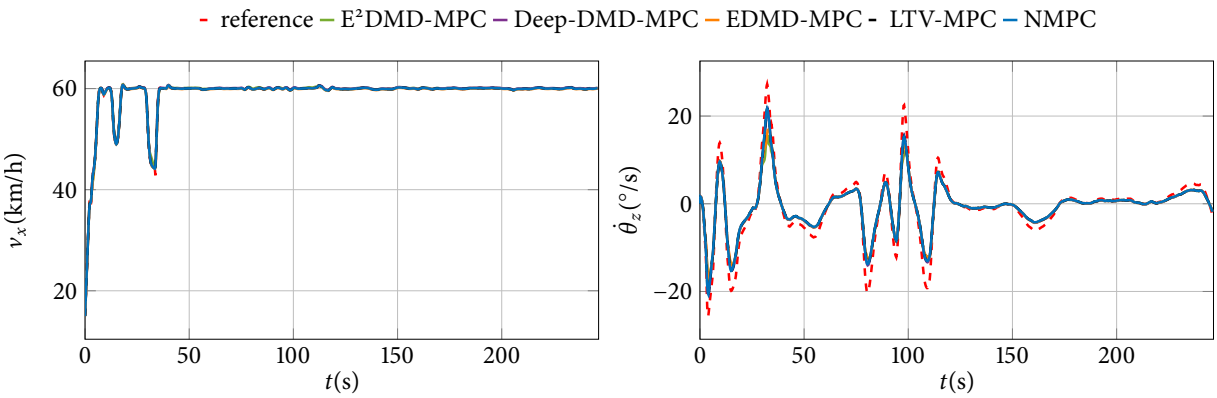

Figure 5.15: Output tracking during slow Nürburgring experiment with $N = 15$.

5.4.4 *Hockenheimring racetrack experiment*

This section presents the analysis of the Hockenheimring racetrack in CarMaker. It is another experiment conducted under higher speed conditions and provides several insightful conclusions. The normalized closed-loop costs for $N = 5$ and $N = 15$ are listed in Table5.9, while the nominal value of the E²DMD-MPC cost is $3.6393 \cdot 10^8$. The duration of the experiment is $T_{sim} = 297.7$ s.

Firstly, NMPC is again characterized by its superior tracking performance (lowest closed-loop cost) over both evaluated prediction horizons, indicating its robustness under different conditions. LTV-MPC consistently shows lower performance at both prediction horizons, characterizing it as the least effective MPC strategy in this experiment. E²DMD-MPC and EDMD-MPC both maintain relatively consistent and moderate performance across different conditions. Although they do not outperform NMPC, their reliability shows that they are a viable option in certain circumstances, although not always the most efficient.

The performance of Deep-DMD-MPC for this manoeuvre is particularly noteworthy as it performs very poorly for $N = 5$ (high cost) and improves significantly when the prediction horizon is extended to $N = 15$. Although it underperforms for the short horizon, its efficiency increases for longer prediction periods, indicating its potential advantages in more complex or extended scenarios.

The computational costs are documented in the Table5.10in terms of execution times. The results are very similar to those mentioned in the previous three sections. E²DMD-MPC and Deep-DMD-MPC have the lowest execution times, while NMPC has the highest.

Table 5.9: Hockenheimring experiment normalized closed-loop cost

Prediction horizon	E²DMD-MPC	Deep-DMD-MPC	EDMD-MPC	LTV-MPC	NMPC
$N = 5$	1	4.1689	1.0037	3.9029	0.9270
$N = 15$	1.6154	0.7444	1.4675	3.0408	0.6284

Figure5.16contains the comparison of the output signals for $N = 5$ for the entire run, while the shorter time window is shown in Figure5.17. Corresponding side slip angles are shown in Figure5.18. Output responses for $N = 15$ are shown in the same way in Figure5.19and Figure5.20, while the side slip angles can be seen in Figure5.21. Based on the longitudinal velocity responses,

Table 5.10: Hockenheimring experiment execution times (ms)

Controller	$N = 5$				$N = 15$			
	Mean	Median	Min	Max	Mean	Median	Min	Max
E²DMD-MPC	2.784	2.071	1.947	17.546	19.941	18.742	18.430	38.409
Deep-DMD-MPC	2.986	2.037	1.917	31.302	20.285	18.786	18.305	42.679
EDMD-MPC	5.023	2.918	2.569	39.329	22.580	21.254	19.952	60.549
LTV-MPC	7.143	6.601	6.005	87.116	33.193	31.671	30.805	282.542
NMPC	37.182	37.132	5.007	46.473	109.909	111.328	13.047	133.229

it is easy to see that LTV-MPC does not follow the reference for neither prediction horizon. If we also consider the time intervals $t_1 \in [110,\ 120]$ s and $t_2 \in [130,\ 140]$ s, a deterioration of the reference tracking for E²DMD-MPC and EDMD becomes clear in the case of a longer prediction horizon.

The poor performance of Deep-DMD-MPC for $N = 5$ causes yaw rate spikes on several occasions, some of which occur at about $t_{s1} = 20$ s, $t_{s2} = 100$ s, $t_{s3} = 190$ s, and $t_{s4} = 280$ s. The same can be seen in Figure5.16, where it is clear that other methods do not exhibit the same behaviour. These spikes are caused by side slip angle contraint violations shown in Figure5.18, which are caused by badly allocated wheel torques. Looking at the vehicle responses for $N = 15$, it is obvious that these spikes do not occur. Similar results are reported for Nürburgring experiment in Section 5.4.2, but to a much lesser extent. In other words, the slip angle constraint violations are not as extreme, so the corresponding cost value degradation is smaller.

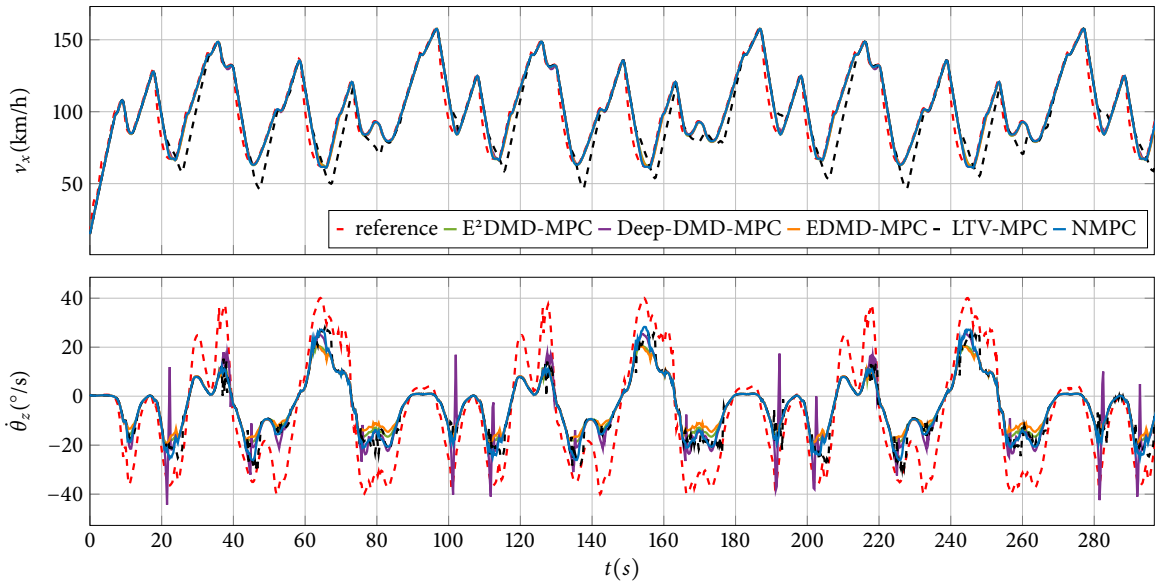

Figure 5.16: Output tracking during Hockenheimring experiment with $N = 5$.

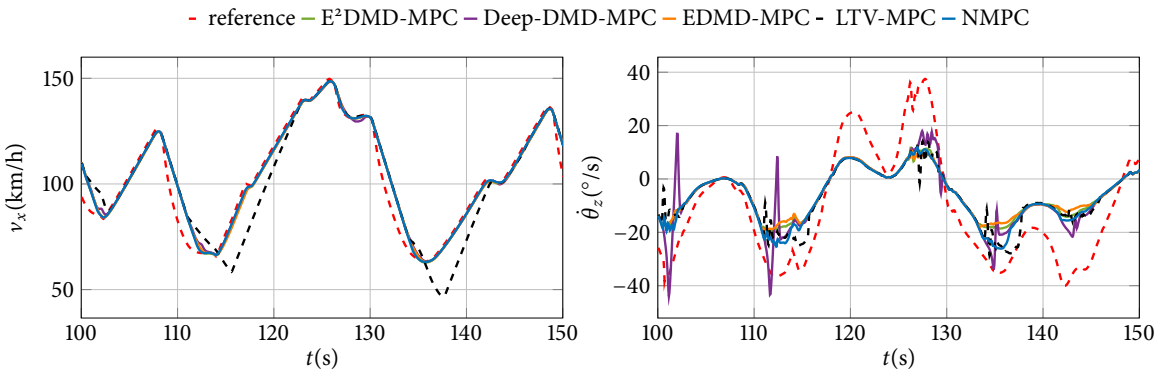

Figure 5.17: Output tracking during Hockenheimring experiment with $N = 5$ (shorter time window).

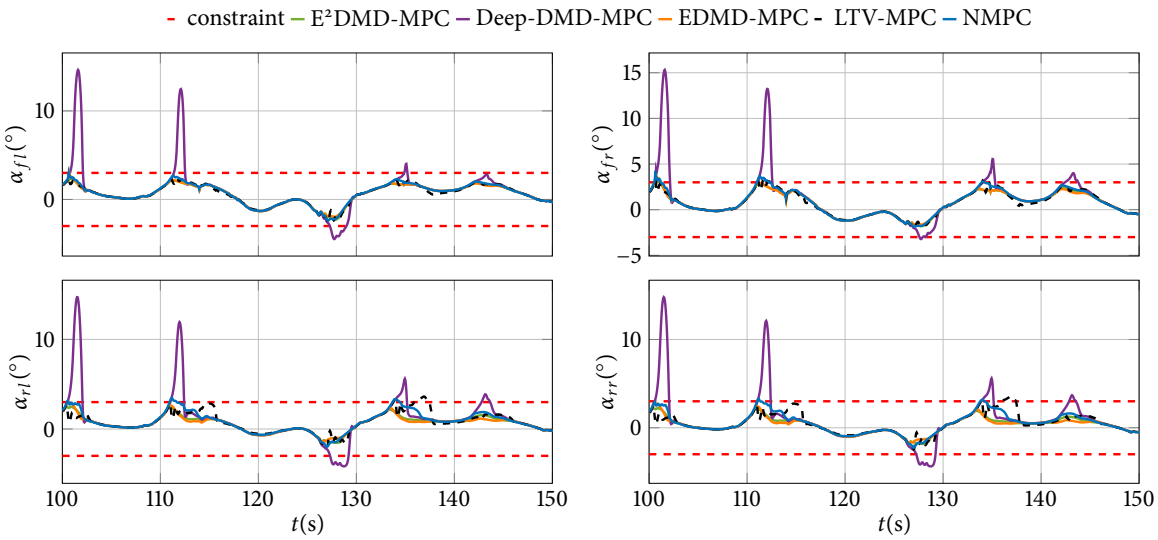

Figure 5.18: Slip angles during Hockenheimring experiment with $N = 5$ (shorter time window).

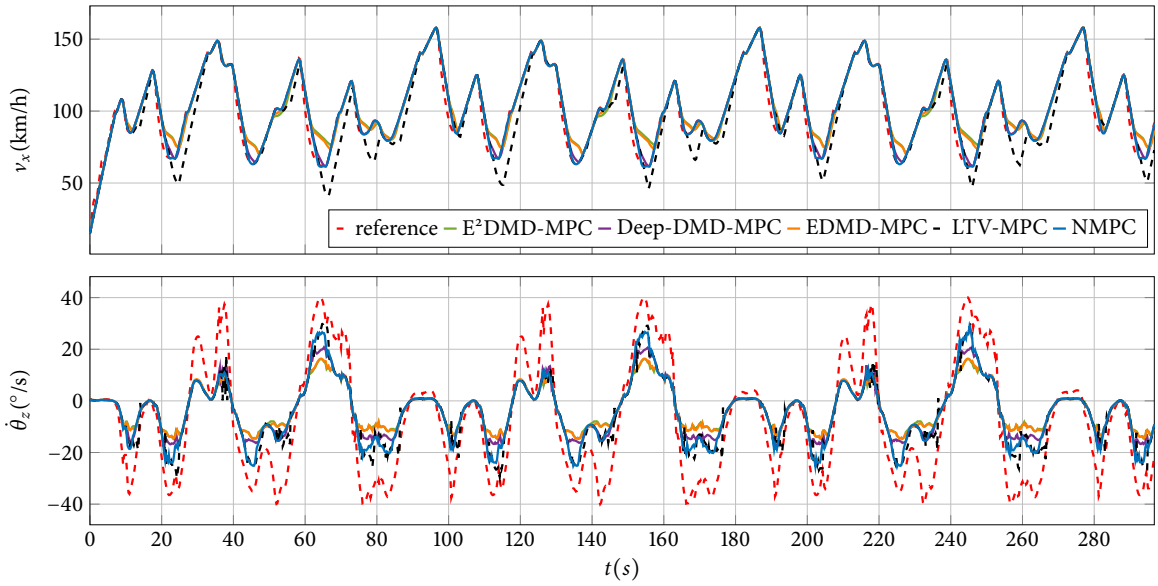

Figure 5.19: Output tracking during Hockenheimring experiment with $N = 15$.

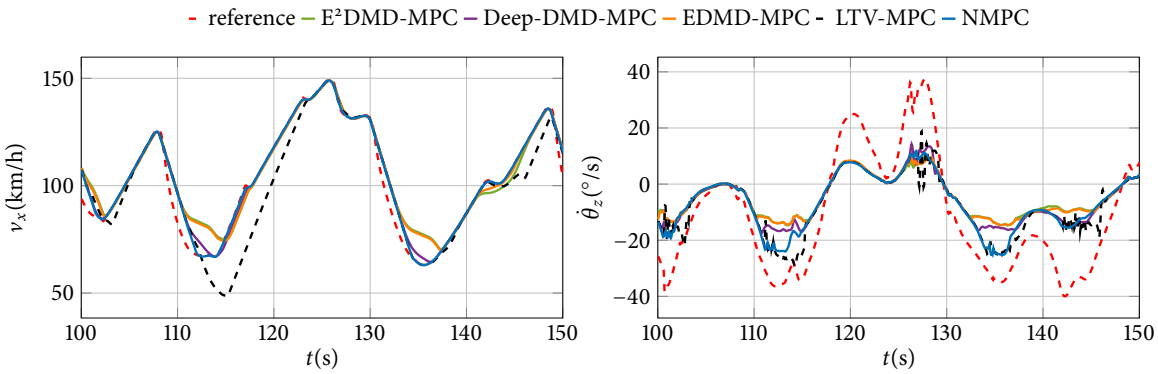

Figure 5.20: Output tracking during Hockenheimring experiment with $N = 15$ (shorter time window).

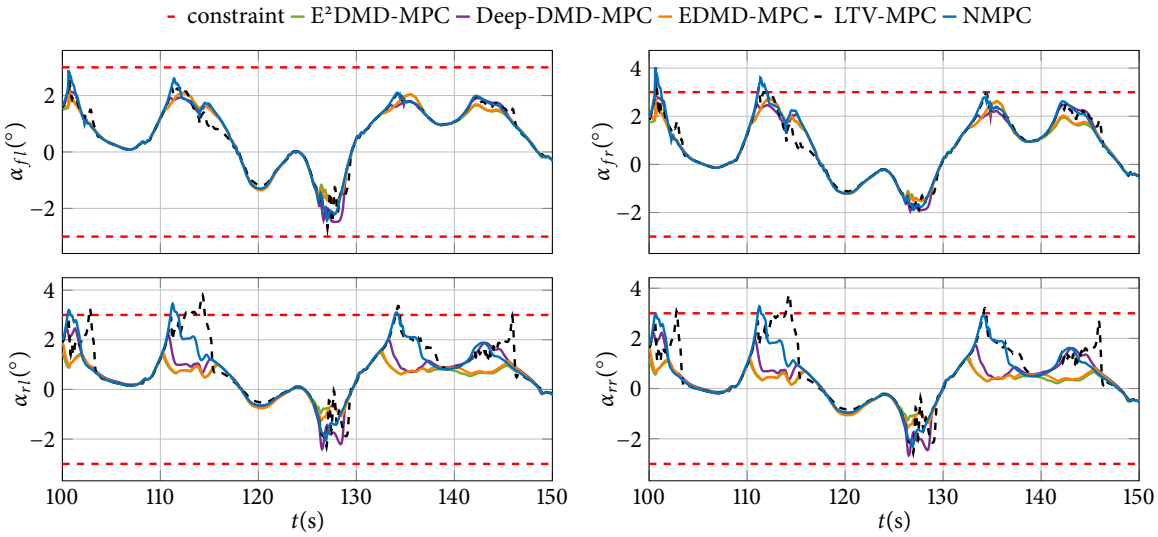

Figure 5.21: Slip angles during Hockenheimring experiment with $N = 15$ (shorter time window).

5.4.5 *Concluding remarks*

The comprehensive simulations and analyzes presented in this study evaluate the performance of various MPC strategies, including E²DMD-MPC, Deep-DMD-MPC, EDMD-MPC, LTV-MPC, and NMPC, in a spectrum of dynamic vehicular manoeuvres and racetrack simulations. Using simulation environments such as MATLAB Simulink and CarMaker and utilizing the computing capacity of a high-performance computer setup, these controllers are systematically compared with each other under uniform test parameters. The efficiency of the controllers is benchmarked through random manoeuvres as well as high and low speed experiments on the Nürburgring and Hockenheimring racetracks.

The results explain the trade-offs between computational efficiency and manoeuvre handling performance. NMPC consistently shows superior manoeuvreing capabilities, especially in high-speed racetrack experiments, as evidenced by the consistently lower normalized closed-loop costs. However, this comes at the cost of higher computational demands, which is reflected in longer execution times, especially when the prediction horizon is extended. This highlights the critical trade-off between control performance and computational feasibility, particularly in real-time applications where execution speed is very important.

On the other hand, Koopman-based controllers, specifically E²DMD-MPC and Deep-DMD-

MPC, offer a balanced compromise between computational efficiency and control performance. Their ability to achieve competitive efficiency at significantly lower execution times makes them a viable alternative, especially for scenarios where computational resources are limited or real-time responses are crucial. Deep-DMD-MPC in particular shows a remarkable improvement in efficiency as the prediction horizon gets longer, emphasising its potential in scenarios that require larger prediction windows. However, it also shows poor performance in some situations with small prediction horizon, indicating unpredictability and tuning complexity likely due to the neural network background.

LTV-MPC, despite its theoretical simplicity and lower computational cost than NMPC, underperforms in terms of control efficiency in most scenarios. Its performance is significantly better in the low-speed Nürburgring experiment, suggesting that it may be better suited for applications where the nonlinearities are less pronounced or more predictable.

In summary, this study not only provides valuable insights into the comparative performance of different MPC strategies in complex vehicular control scenarios, but also emphasises the crucial importance of balancing control performance and computational efficiency. The results underline the potential of Koopman-based controllers as a promising middle ground that offers a viable compromise between the high control performance of NMPC and the computational efficiency required for real-time applications.

5.5 SUMMARY

This chapter deals with the comparative analysis of different model predictive control strategies for vehicle dynamics control, with a focus on torque vectoring applications. The work presented here is based on [70]. The chapter begins with a literature overview, highlighting the evolution from conventional control systems to more complex models that account for the nonlinear dynamics of vehicle behaviour. The discussion then transitions to the exploration of Koopman model identification, where the potential of the Koopman operator to linearize nonlinear dynamics is showcased. In this section, the process of model identification using the Koopman operator is outlined and a comprehensive approach to capturing vehicle dynamics through linear approximations is presented.

In the following section, the design of several MPC approaches is explained, including the linear time-variant MPC (LTV-MPC), the Koopman operator-based MPC (KMPC) and the nonlinear MPC (NMPC). Particular attention is paid to the formulation of these controllers and the integration of constraints, which are important for the effective implementation of torque vectoring control strategies.

Through simulations of randomized test runs and experiments on renowned racetracks like Nürburgring and Hockenheimring, the chapter evaluates the performance of E^{2}DMD-MPC, Deep-DMD-MPC, EDMD-MPC, LTV-MPC and NMPC. This evaluation sheds light on the trade-offs between model complexity, computational efficiency and control performance. NMPC proves to be the superior strategy in terms of manoeuvre handling, but incurs a higher computational cost. Conversely, Koopman-based controllers, in particular E^{2}DMD-MPC and Deep-DMD-MPC, offer a promising trade-off between computational complexity and control efficiency, although their effectiveness varies with the prediction horizon. E^{2}DMD-MPC proves to be more consistent in the experiments. On the other hand, although the performance of Deep-DMD-MPC is superior to other Koopman-based controllers at longer prediction horizons, it shows performance fluctuations and occasionally poor performance at shorter horizons.

The chapter concludes by emphasising the importance of selecting an appropriate control strategy based on the application, including real-time control capabilities and the degree of nonlinear dynamics involved. The comparative analysis presented here provides a foundation for understanding the problems of advanced Koopman operator-based control strategies in vehicle dynamics.

6

Experimental investigation

The sixth chapter presents an experimental investigation of vehicle dynamics control, focusing on an approach that uses a scaled vehicle model on a treadmill to test control algorithms. It begins with a description of the experimental setup, along with a brief literature overview. In the following section, the identification of the nonlinear vehicle model parameters is presented. It is carried out through experiments that include direct measurements and systematic tests, and is validated by comparing the experimental data with the model predictions. The chapter further presents the identification of the Koopman model using the EDMD algorithm. This section details the generation of datasets, the sampling strategy and the model learning process, demonstrating the effectiveness of the model using different datasets and test scenarios. A unique aspect of this discussion is the examination of the impact of dataset size and distribution on the predictive accuracy of the EDMD model. Further, the NMPC and KMPC strategies are presented and evaluated. Detailed experimental results are documented for both controllers, which includes the controller setup, execution times and performance analysis for two test manoeuvres, multiple and double lane change. Finally, the effects of different references and prediction horizons on the control performance are evaluated and some of the experimental results are compared to simulations, followed by concluding remarks.

6.1 Experimental setup

6.1.1 *Background*

Driven by the ambition to improve vehicle dynamics control systems, engineers and researchers have sought innovative techniques to experimentally validate control algorithms. Many research groups have recently employed scaled vehicle models for this validation, as highlighted in [119, 120,121,122].

With the increasing complexity of these systems, the need for more realistic and flexible test setups has also increased. To bypass the limitations associated with the use of scaled models and to facilitate the testing of control algorithms under different road conditions without the logistical problems of obtaining a track, several research teams have introduced the concept of the road simulator. This concept serves as a laboratory-based test platform for vehicle dynamics control, as described in [123, 124, 125, 126]. The setup includes a treadmill for testing control algorithms at high speeds in a limited area. The success of this approach depends critically on the integration of precise sensor systems to measure the position and orientation of the vehicle on the treadmill.

These sensors are essential for real-time feedback to the control algorithms, enabling precise changes and optimizations according to dynamic conditions. In addition, treadmill configurations can be augmented with additional degrees of freedom, including variable track inclinations, to better simulate a range of road conditions, from flat to downhill, and allow for a thorough evaluation of control strategies for vehicle dynamics.

In the mentioned papers, the dynamic similarity between the scaled and the full-size vehicle is achieved by applying the Buckingham Π theorem [127]. It allows the reduction of the original parameters to a (possibly smaller) set of dimensionless parameters. By ensuring that these dimensionless parameters are equivalent between the scaled model and the full-size vehicle, one can more accurately predict the behaviour of the full-size system based on tests conducted with the scaled model.

6.1.2 *Setup description*

The control algorithm developed in this research is evaluated using a scaled vehicle on a treadmill. Figures6.1and6.2show the treadmill and the scaled vehicle respectively. The same experimental setup with slightly different parameters such as tires and mass distribution was previously used in [128,129,130,115].

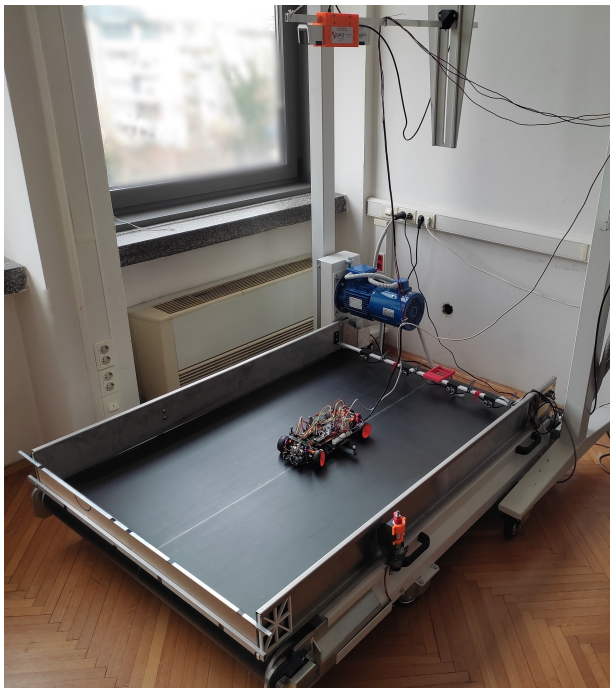

Figure 6.1: Treadmill with the scaled vehicle.

An *Optitrack V120 Duo* optical tracking system, equipped with two overhead cameras, is used to monitor the vehicle's position and velocity. Three markers are attached to the vehicle so that the system can precisely track the center of gravity. The wheel speeds are measured using Hall sensors on the motors. The reference for the longitudinal velocity is generated with a PI controller that controls the position of the vehicle. The setpoints for lateral position and yaw rate, on the other hand, are calculated based on the desired trajectory, which depends on the current position of the vehicle. The control algorithm runs on a *dSPACE MicroLabBox* with a dual-core 2 GHz processor.

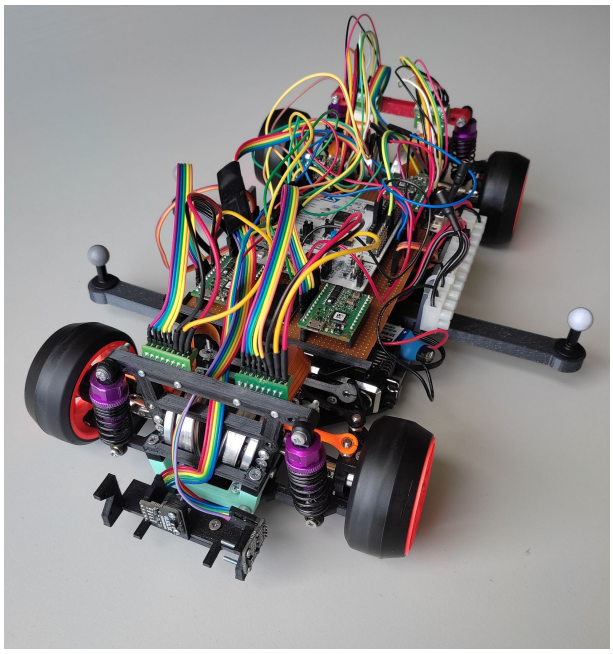

Figure 6.2: Scaled vehicle used in experiments.

The scaled vehicle is powered by four independent motors, with the steering angle controlled by a servo motor. To increase the complexity of the manoeuvres and reduce the friction between the treadmill and the vehicle, low-friction tires are used.

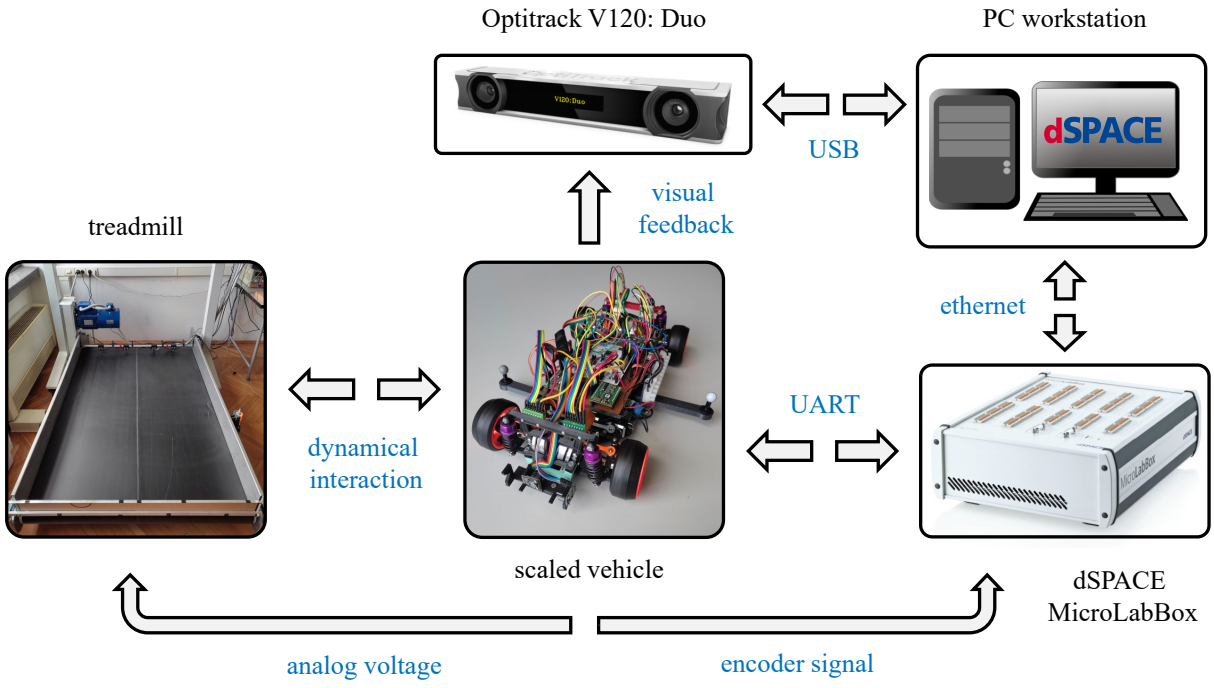

Figure 6.3: System diagram of the experimental setup.

Figure6.3illustrates the system diagram, and the operation process of the setup is outlined as follows:

1. The longitudinal and lateral position of the vehicle's center of gravity (CoG) is captured by stereo cameras, while wheel speeds are measured with Hall sensors.

2. The camera image is processed and filtered on a PC workstation. All calculated states (with the exception of wheel speeds) are then transmitted to the *MicroLabBox* platform via Ethernet. The wheel speeds are transmitted directly from the vehicle via UART communication. The operator uses the *dSPACE ControlDesk* software on the PC workstation for the input settings.

3. The control algorithm running on the *MicroLabBox* platform receives the measured values as input to calculate the optimal torque and steering signals, which are then sent back to the vehicle via UART. At the same time, the speed of the treadmill is monitored via an encoder and adjusted by changing the analog voltage signal at one of the terminals of the treadmill's motor power converter.

4. Responding to these signals, the vehicle modifies its movement, resulting in dynamic interaction with the treadmill.

5. This entire process is cyclical and starts again at step 1.

6.2 VEHICLE PARAMETERS IDENTIFICATION

Similar to the previous chapter, the parameters for the nonlinear vehicle model are initially determined through the analysis of experimental trajectories. The configuration of this model follows the structure outlined in2.1.2, incorporating a piecewise linear tire model, a coupling of tire forces and an alternative slip formulation. The model is compactly represented by the equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)). \tag{6.1}$$

Here, the state vector is defined as $\mathbf{x} = [v_x \; v_y \; \theta_z \; \dot{\theta}_z \; \omega_{fl} \; \omega_{fr} \; \omega_{rl} \; \omega_{rr} \; Y]^T$, and the input vector is defined as $\mathbf{u} = [\delta_f \; T_{fl} \; T_{fr} \; T_{rl} \; T_{rr}]^T$.

Important parameters such as the mass, length and moment of inertia of the vehicle are either measured directly or calculated using conventional experiments. However, other parameters such as the tire stiffness coefficients C_{yf} and C_{yr}, the rolling resistances f_f and f_r, the axle viscosity b_w and the tire friction coefficient μ have to be estimated. For this purpose, experimental input-output data is collected and the target parameters are determined using the System Identification Toolbox in MATLAB. The identified parameters are listed in Table6.1, and comparisons between the recorded experimental data used for identification and the predictions of the nonlinear model are shown in Figure6.4and Figure6.5.

The figures illustrate the responses of the vehicle in terms of longitudinal velocity, yaw angle, yaw rate and global lateral position. It can be observed that the responses for v_x (longitudinal velocity) and $\dot{\theta}_z$ (yaw rate) have very small prediction errors. In contrast, the errors for θ_z (yaw angle) and Y (lateral position) accumulate over time, which is due to the fact that these quantities are determined by integrating the yaw rate and vehicle velocities. However, when employing receding horizon strategies like MPC, the model is only used for short-term predictions. Therefore, Figures6.4and6.5also show a simulated response where the simulation is reset every $T_{res} = 2$ seconds, effectively minimizing the simulation errors.

The prediction accuracy for the experiments shown in the figures was quantitatively evaluated for different restart intervals using the MNPE (3.46) with $N_{sim} = T_{sim}/T_s$ as the number of samples.

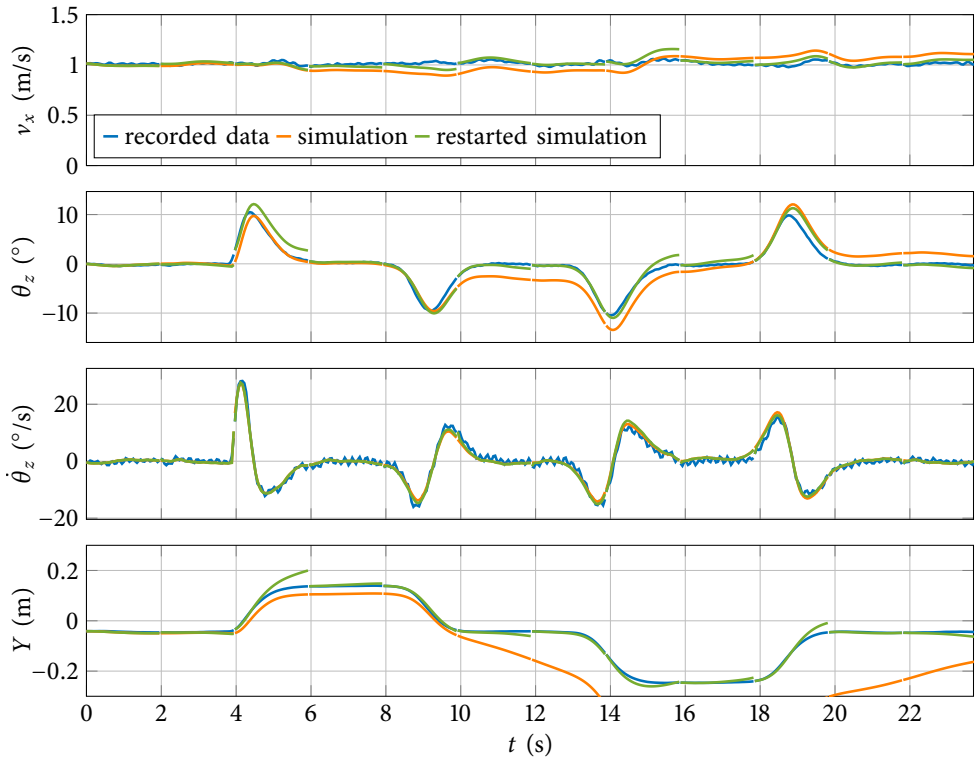

Figure 6.4: Recorded experimental compared to simulated data for $v_x \approx 1$ m/s.

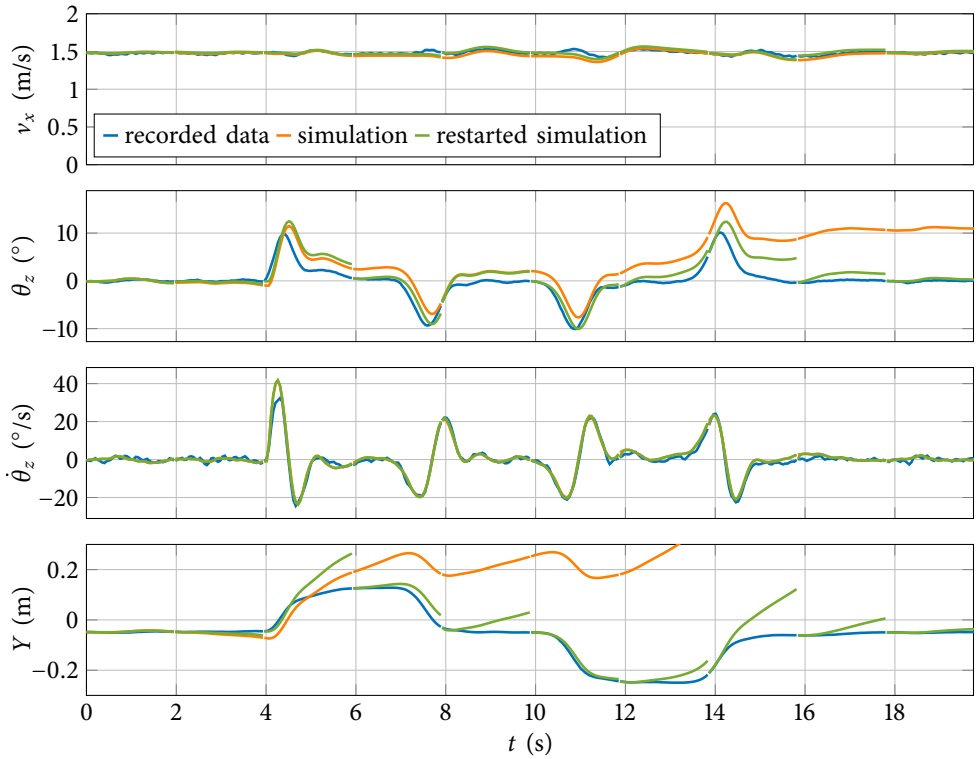

Figure 6.5: Recorded experimental compared to simulated data for $v_x \approx 1.5$ m/s.

Table 6.1: Scaled vehicle parameters

Parameter	Description	Value	Unit
m	mass of the vehicle	1.35	kg
l_f	front axle to CoG distance	0.128	m
l_r	rear axle to CoG distance	0.128	m
w	half of the wheel track	0.08	m
J_z	moment of inertia around yaw axis	0.03373	$kg \cdot m^2$
$b_{w\bullet}$	front/rear axle viscous friction	$5 \cdot 10^{-5}$	Nm/(rad/s)
$f_\bullet$	front/rear wheel rolling resistance	0.0017	m
$C_{x\bullet}$	front/rear axle longitudinal tire stiffness	24.995	N
$C_{y\bullet}$	front/rear axle lateral tire stiffness	21.4842	N/rad
$R_{w\bullet}$	front/rear axle effective tire radius	0.0315	m
$J_{w\bullet}$	front/rear wheel moment of inertia	$3.697 \cdot 10^{-5}$	$kg \cdot m^2$
μ	equivalent friction coefficient	0.9	-
ε_0	slip denominator coefficient	10^{-4}	-

Here, the sample time is $T_s = 1$ ms, and the total simulation time T_{sim} is adjusted to ensure a fair comparison between the two experiments. To facilitate the comparison, the errors are normalized to the highest observed error, namely MNPE($T_{res} = 20$s). The results, shown as a bar chart in Figure6.6, support the claim that reducing the prediction horizon also reduces the prediction error, making this model suitable for MPC algorithms.

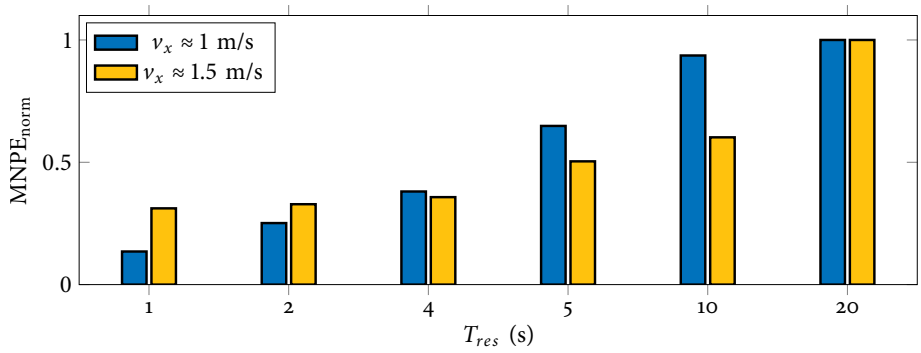

Figure 6.6: Normalized MNPE for two different experiments and various simulation restart times.

6.3 KOOPMAN MODEL IDENTIFICATION

As mentioned in Section5.2, the dataset for learning the Koopman model can be created in one of the following ways: 1) performing simulations of multiple trajectories in a high-fidelity simulation environment (or using experiments with real vehicles) or 2) using a nonlinear model according to the first principles, whose parameters are determined using data obtained either from a high-fidelity simulation environment or a real vehicle, and then simulating different scenarios with this model. As in the previous chapter, the second approach is chosen here, but the first approach is used for comparison.

6.3.1 *Data collection*

The dataset is generated by uniformly sampling the initial state vector and the input vector sequence over p steps. The system (6.1) is then simulated using the sampled initial states and excited by the sampled input sequences. The initial states v_{x0}, v_{y0}, θ_{z0}, $\dot{\theta}_{z0}$ and Y_0 are sampled from certain intervals while the wheels were assumed to be rolling freely. The input sequence is sampled within specific intervals. In addition, the torque rates are restricted. In this way, both the continuity and smoothness of the sample input trajectory are taken into account. The steering wheel angle rate $\Delta\delta_f$ can also be restricted. In this chapter however, this condition is as the servomotor can reach the maximum angle within a few milliseconds. This means that the limit values for the steering rate are always complied with and are therefore redundant.

The dataset comprises a total of $N_s = 200000$ sample trajectories, each consisting of $p = 25$ steps with a sample time of $T_s = 0.05s$. The values and ranges of the numerical data are as follows: $v_{x0} \in [0.1, 2.5]$ m/s, $v_{y0} \in [-0.5, 0.5]$ m/s, $\theta_{z0} \in [-30, 30]$ °, $\dot{\theta}_{z0} \in [-90, 90]$ °/s, $Y_0 \in [-1, 1]$ m, $\omega_{\bullet\star 0} = v_{x0}/R_{wf,r}$, $T_{\bullet\star} \in [-50, 50]$ mNm, $\delta_{sw} \in [-15, 15]$ ° and $\Delta T_{\bullet\star} \in [-12.5, 12.5]$ mNm. The number of sample points in the dataset is equal to $N_{total} = N_s \cdot p = 5 \cdot 10^6$. Of these points, $N_{nl} = 1488470$, i.e. about 29.77% covers nonlinear ranges of tire slip angles. The definition of the nonlinear ranges is the same as in Section5.2.2, i.e. they are the regions outside the linear force range, which in the model (2.41a) means that the forces are saturated. The dataset was additionally divided into learning and test sets, with the learning set containing $N_{learn} = 160000$ and the test set $N_{test} = 40000$ trajectories.

6.3.2 *Learning Koopman model*

Similar to the one in the Section5.2.3, the polynomial basis P_d, defined by (3.11), is chosen to form the set of basis functions $\phi(\cdot)$. The difference here is that the side slip angles are not added to the basis.

A linear state-input relationship is achieved by extending the state-space of the nonlinear system (6.1) by the steering angle, while the steering rate becomes a new input. In this case, the extended state and input vectors are

$$\begin{aligned} \tilde{\mathbf{x}} &= [v_x \; v_y \; \theta_z \; \dot{\theta}_z \; \omega_{fl} \; \omega_{fr} \; \omega_{rl} \; \omega_{rr} \; Y \; \delta_{sw}]^T, \\ \tilde{\mathbf{u}} &= [\Delta\delta_{sw} \; T_{fl} \; T_{fr} \; T_{rl} \; T_{rr}]^T. \end{aligned} \tag{6.2}$$

To improve the numerical stability of the learning algorithm, the collected data was normalized to the range $[-1, \; 1]$ using (5.6).

By specifying the order d of the basis P_d, the size of the extended state-space is determined. The larger the original state-space is, the faster the size of the extended state-space grows with increasing order d. In (6.2) the size of the state-space is $n_x = 10$.

Based on the collected data described in the previous subsection, different EDMD models are identified for different order values. The performance of the identified models is evaluated on learning and test sets using the MNPE (3.46) averaged over all trajectories of a given set. The MNPE values for different orders are given in the Table6.2together with the corresponding size of the extended state-space n_Φ.

By examining the numerical data in the table, one can see how quickly n_Φ increases. In contrast, MNPE does not seem to show a monotonic behaviour and has a local minimum for the

Table 6.2: MNPE errors evaluated on learning and test set

d	1	2	3	4	5
n_Φ	11	66	286	1001	3003
$\text{MNPE}_{\text{learn}}$ [%]	0.9409	0.8452	0.8692	0.8930	0.8828
$\text{MNPE}_{\text{test}}$ [%]	0.9464	0.8447	0.8702	0.8935	0.8832

order $d = 2$. Here *local* is emphasised because it is not certain that this is actually the global error minimum, as the error could decrease for a higher order. However, due to the rapid growth of the extended state space, it is pointless for practical use in control applications to increase the order further, as this would greatly increase the memory and computational hardware requirements.

6.3.3 *Dataset distribution*

In the following, the Koopman model with polynomial basis of order $d = 2$ is used. To justify its applicability to a real system, it is compared to the experimentally recorded data using a restarted simulation in a similar way as in Section6.2. It is also compared to Koopman models of the same size learned from the following data:

1. experimental data used for the identification of the nonlinear model in Section6.2, with a total of 852 samples, of which 9.9532% are in nonlinear tire regions,
2. partial dataset 1 (PD1): $5 \cdot 10^5$ samples from the nominal dataset described in Section6.3.1 with 29.75% of the samples in nonlinear regions,
3. partial dataset 2 (PD2): $5 \cdot 10^4$ samples from the nominal dataset with 30.16% of the samples in nonlinear regions,
4. partial dataset 3 (PD3): 5000 samples from the nominal dataset with 29.98% of the samples in nonlinear regions,
5. partial dataset 4 (PD4): 500 samples from the nominal dataset with 34.8% of the samples in nonlinear regions.

The distribution of the sampled state and input trajectories from the nominal and experimental datasets as well as from PD3 and PD4 is shown in Figure6.7and Figure6.8. The datasets PD1 and PD2 are not shown in order to keep the graphs clear and visible.

Figure6.7shows the distribution of a triplet $\{v_x, v_y, \dot{\theta}_z\}$, while the other states are omitted for the sake of simplicity. These omitted states can be derived by integrating the represented states (θ_z and Y) or are approximately proportional to them ($\omega_{\bullet\star}$). Likewise, Figure6.8shows the distribution of the inputs. However, as not all inputs can be shown in a single plot, only the triplet $\{\delta_f, T_{sum}, T_{diff}\}$ is presented. Torque sum $T_{sum} = T_{fl} + T_{fr} + T_{rl} + T_{rr}$ is responsible for the propulsion of the vehicle in the longitudinal direction, while $T_{diff} = T_{fr} + T_{rr} - T_{fl} - T_{rl}$ is proportionally related to the yaw moment and supports the steering system in manoeuvreing the vehicle.

It is obvious from the graphs that PD3 is a subset of the nominal dataset, while PD4 is a subset of PD3. However, both of them cover reasonably large portion of the nominal dataset, especially

PD3. On the other hand, data collected from the experiment is concentrated around few points and covers small portion of the state and input space.

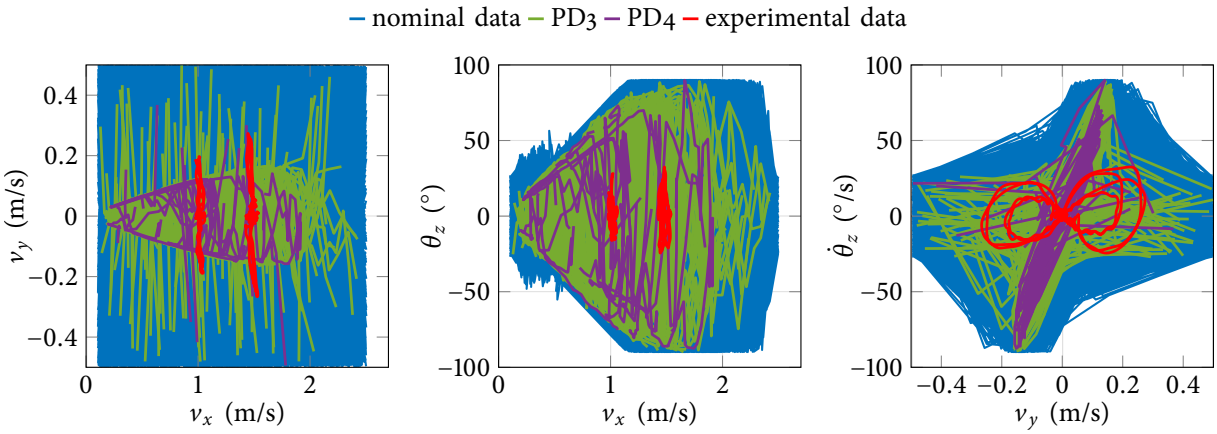

Figure 6.7: The distribution of the sampled state trajectories from different datasets.

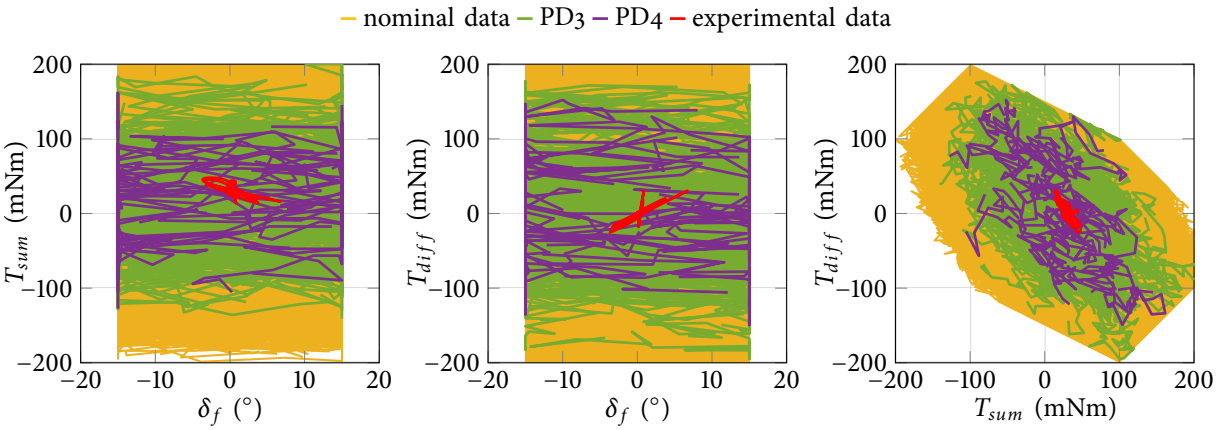

Figure 6.8: The distribution of the sampled input trajectories from different datasets.

6.3.4 *Predictor performance analysis*

In this section, the performance of the predictors is evaluated using three different test scenarios, all of which contain the *multiple lane change* manoeuvre used in [115]:

1. trajectory from the experimental dataset at $v_x = 1.5$ m/s,
2. slow drive trajectory at $v_x = 0.5$ m/s,
3. trajectory including acceleration.

1. *Trajectory from the experimental dataset at $v_x = 1.5$ m/s*

This test contains predictor comparison using one of the experimental trajectories contained in the experimental dataset, in other words, a training set for a predictor based on experimental data. The performance of the predictor for different sizes of the prediction horizon p is presented in

Table6.3. It is evident that as the prediction horizon decreases, the prediction error also decreases, highlighting a fundamental relationship between the prediction horizon and the associated accuracy. This trend is consistent across all datasets. In terms of dataset size, models learned from nominal data, PD1 and PD2 datasets have very similar errors for all sizes of prediction horizon. For PD3 and PD4-based models, the errors are larger, which is probably due to the size of the dataset and the random selection of data points.

Furthermore, the Koopman model learned from the experimental data shows a much smaller prediction error for all horizons p, which is to be expected since it was trained on the same data, while the other predictors used datasets that do not contain the given trajectory. Although the prediction errors of the other Koopman models are larger, they are still reasonably small, especially for control system design, which will be shown later in this chapter.

Table 6.3: MNPE [%] errors for different prediction horizon p using identification data

p	2	5	10	25	50	100
nominal data	0.0108	0.0662	0.1801	0.6293	1.6144	5.1431
experimental data	0.0013	0.0032	0.0041	0.0067	0.0073	0.0106
PD1	0.0109	0.0666	0.1810	0.6319	1.6130	5.0204
PD2	0.0111	0.0674	0.1834	0.6428	1.6819	6.5342
PD3	0.0138	0.0789	0.2049	0.6920	1.8370	5.9777
PD4	0.0198	0.0810	0.1916	0.6734	1.6759	20.706

The corresponding trajectory and the comparison of some predictors are shown in Figure6.9. The simulations are restarted at intervals of $T_{res} = p \cdot T_s = 1.25$s. This resulted in a MNPE(T_{res} = 1.25 s) = 0.6293% for the model trained on the nominal dataset, a MNPE(T_{res} = 1.25 s) = 0.6734% for the model trained on the PD4 dataset and a MNPE(T_{res} = 1.25 s) = 0.0067% for the model derived from experimental data.

2. *Slow drive trajectory at v_x = 0.5 m/s*

For the second test, a trajectory with a constant velocity of v_x = 0.5 m/s is used. Table6.4lists the prediction errors for different prediction horizons. As in the previous test, the errors increase with increasing prediction horizon. In addition, the models learned from the nominal dataset, PD1 and PD2 again have similar performance, while the performance for PD3 decreases slightly. The PD4-based model deteriorates the performance for $p \in \{2, 5\}$, improves it for $p \in \{10, 25, 50\}$ and then deteriorates strongly for $p = 100$.

The cardinality of the experimental dataset (852 samples) is between the cardinality of PD3 (5000 samples) and PD4 (500 samples), but the model learned from the experimental dataset demonstrates much larger errors than the other models (except for the model trained on PD4 for $p = 100$). This is to be expected since the experimental dataset is sparse and does not contain the test trajectory.

The Figure6.10shows the predictor comparison for T_{res} = 1.25s with MNPE(T_{res} = 1.25 s) = 8.0793% for the model trained on the nominal dataset, MNPE(T_{res} = 1.25 s) = 4.3313% for the model trained on the PD4 dataset and a MNPE(T_{res} = 1.25 s) = 23.423% for the model derived from experimental data. Looking at the responses of the predictors, one can conclude that while

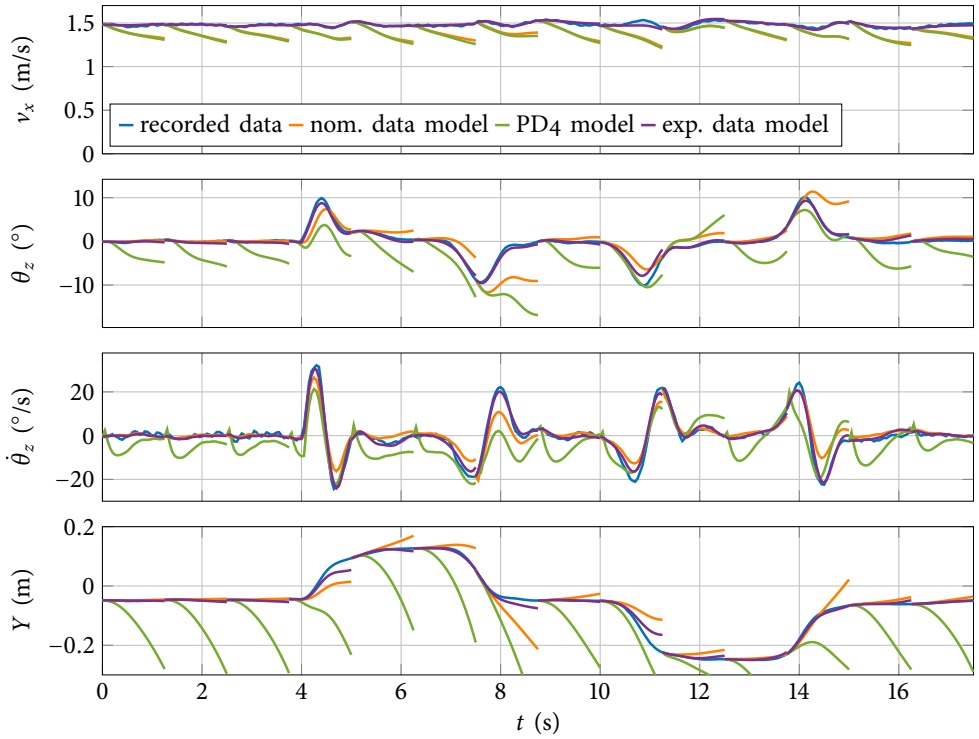

Figure 6.9: Recorded identification test data compared to predition of simulation and experimental data based Koopman models for $T_{res} = 1.25$.

Table 6.4: MNPE [%] errors for different prediction horizon p during slow drive test

p	2	5	10	25	50	100
nominal data	0.1227	0.5376	1.6350	8.0793	24.840	69.004
experimental data	0.8083	1.6208	4.0043	23.4229	73.0029	233.392
PD1	0.1228	0.5425	1.6552	8.1661	25.080	70.419
PD2	0.1276	0.5628	1.7095	8.4225	25.978	71.547
PD3	0.1316	0.5663	1.7068	8.7397	27.698	69.017
PD4	0.2396	0.5691	1.2553	4.3313	10.365	$1.55 \cdot 10^3$

the models for nominal data and PD4 perform better than the model for experimental data, none of them seem to be great (the responses look quite inaccurate). However, the following chapter shows that this can still work quite well when used in a closed loop.

3. *Trajectory including acceleration*

The last test scenario includes a trajectory with acceleration and the error results are given in Table6.5. It shows that the errors increase with increasing prediction horizon, just like in the two previous test cases. Models learned from nominal data, PD1 and PD2 datasets show very similar errors for all prediction horizons, while the errors increase for the PD3 and PD4 datasets due to the significant decrease in sample points they contain. In this scenario, the trend is consistent, in contrast to what was previously observed with other test data.

The model for the experimental data performs the worst at all lengths of the prediction horizon, as indicated by very large errors.

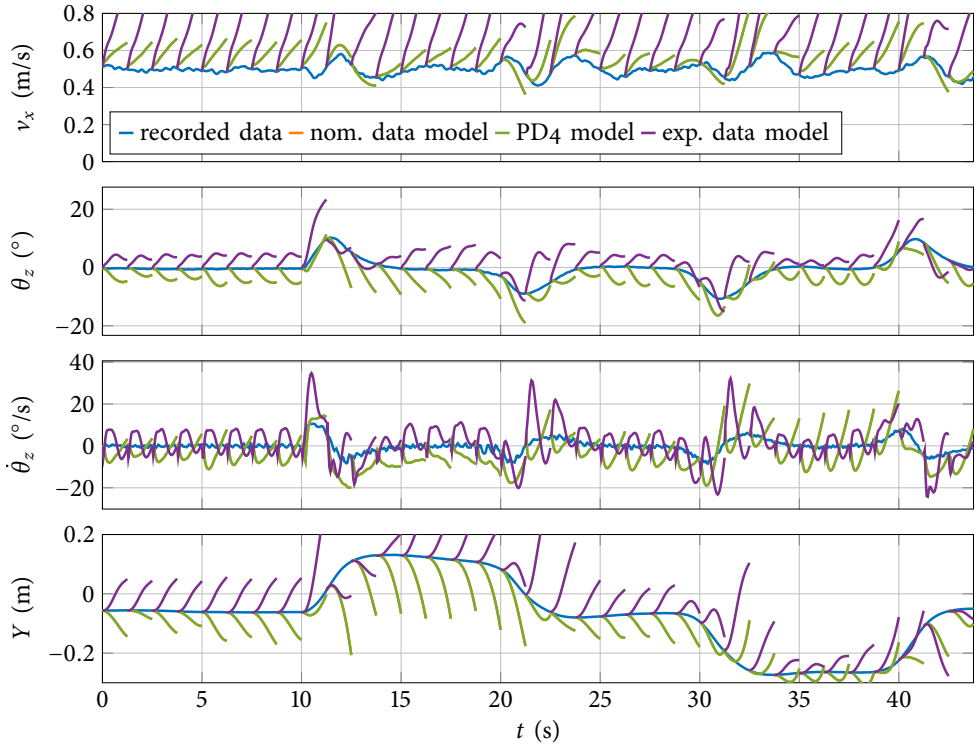

Figure 6.10: Recorded slow drive test data compared to predition of simulation and experimental data based Koopman models for $T_{res} = 1.25$.

Table 6.5: MNPE [%] errors for different prediction horizon p during acceleration test

p	2	5	10	25	50	100
nominal data	0.0089	0.0556	0.1823	0.8386	2.4539	6.7576
experimental data	11.904	49.322	72.562	81.543	94.885	$2.45 \cdot 10^3$
PD1	0.0089	0.0553	0.1807	0.8275	2.4098	6.4734
PD2	0.0089	0.0548	0.1797	0.8239	2.3804	6.7743
PD3	0.0101	0.0606	0.1910	0.8494	2.4690	7.2220
PD4	0.0138	0.0770	0.2682	1.5699	5.0420	276.67

The simulation comparison for $T_{res} = 1.25$s is shown in Figure6.11. The errors for the given test case are MNPE($T_{res} = 1.25$ s) = 0.8386% for the model trained on the nominal dataset, MNPE($T_{res} = 1.25$ s) = 1.5699% for the model trained on the PD4 dataset and a MNPE($T_{res} = 1.25$ s) = 81.543% for the model derived from experimental data.

The figures, tables and corresponding analysis show that the Koopman models learned from simulation data perform better than those learned from experimental data, except when evaluated on experimental data itself. The errors increase with increasing prediction horizon and with decreasing cardinality of the dataset (with a few exceptions that are probably due to randomness). This information indicates that the developed model accurately describes nonlinear scaled vehicle dynamics and can therefore be used for the development of control systems. Furthermore, this justifies the approach proposed at the beginning of the section (and in5.2), where known model information can be used to augment a smaller experimental dataset and generate a larger dataset through simulations.

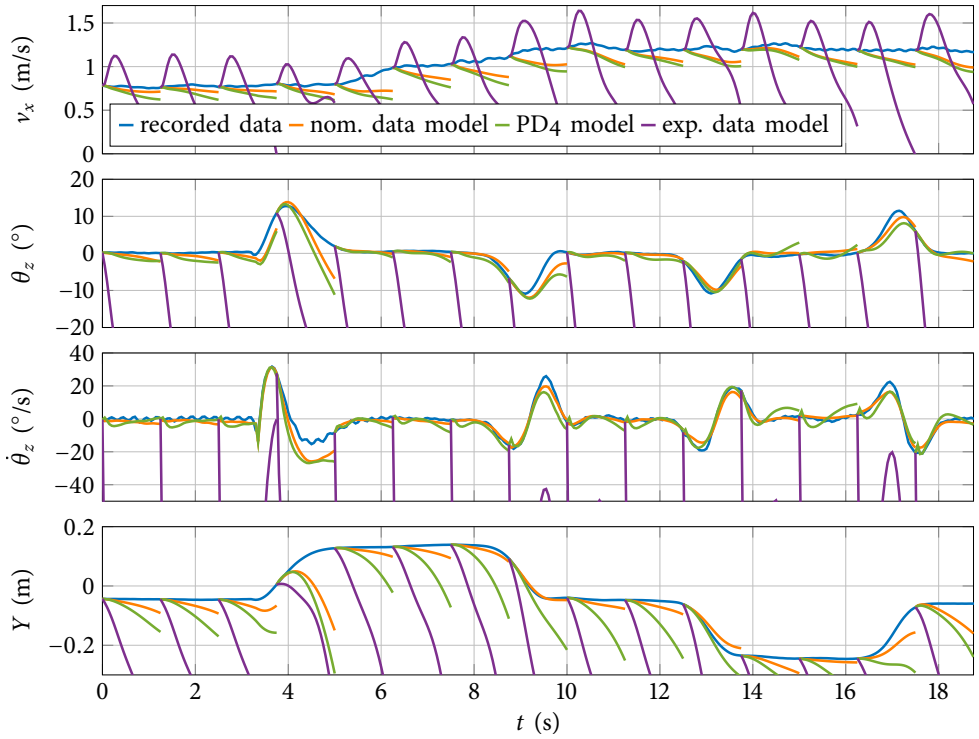

Figure 6.11: Recorded acceleration test data compared to predition of simulation and experimental data based Koopman models for $T_{res} = 1.25$.

The identification of nonlinear models in this context can be regarded as a well-structured data compression method in which the original experimental data is the one which is being compressed, the identified nonlinear model is the compressed data itself, and other datasets are uncompressed data generated from the compressed data (model) using the data decompression method (nonlinear model simulation). While the information about the experimental data is encoded in the parameters of the nonlinear model, the model also contains additional information about the physical process encoded in its structure. For this reason, much less data is required to determine the parameters compared to an unstructured model based on the Koopman operator, for example.

6.4 MODEL PREDICTIVE CONTROL

This section deals with the derivation of two versions of MPC for torque vectoring applications: 1) NMPC and 2) KMPC. The former uses a nonlinear vehicle model, while the latter relies on the Koopman model obtained by EDMD from the nominal dataset described in Section6.3.1.

6.4.1 *Nonlinear MPC*

The ordinary differential equation in (6.1) represents the nonlinear dynamics of the system. In order to formulate the corresponding optimization problem as a nonlinear program, the equation must be discretized. By applying some of the techniques from5.3.3, a discrete-time nonlinear model is obtained, which is denoted as $\mathbf{f}_{nd}$.

The resulting optimization problem can be expressed as a nonlinear program, given by:

$$\min_{\mathbf{U}_t} \quad J(\mathbf{x}_t, \mathbf{u}_{t-1}, \mathbf{Y}_t^{ref}, \mathbf{U}_t) \tag{6.3a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}_d(\mathbf{x}_k, \mathbf{u}_k), \tag{6.3b}$$

$$\mathbf{y}_k = C_{nl}\mathbf{x}_k, \tag{6.3c}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max}, \tag{6.3d}$$

$$\Delta\mathbf{u}_{min} \leq \Delta\mathbf{u}_k \leq \Delta\mathbf{u}_{max}, \tag{6.3e}$$

$$\mathbf{x}_t = \mathbf{x}(t), \tag{6.3f}$$

$$\mathbf{u}_{t-1} = \mathbf{u}(t-1). \tag{6.3g}$$

Here $\mathbf{x}(t)$ represents the current state measurements, and $\mathbf{u}(t-1)$ denotes the previously applied control inputs. The change in control inputs is represented by $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$. The control input sequence is denoted as $\mathbf{U}_t = [\mathbf{u}_t, \mathbf{u}_{t+1}, ..., \mathbf{u}_{t+N-1}]$, and the reference sequence is denoted as $\mathbf{Y}_t^{ref} = \left[\mathbf{y}_t^{ref}, \mathbf{y}_{t+1}^{ref}, ..., \mathbf{y}_{t+N}^{ref}\right]$.

6.4.2 *Koopman operator-based MPC*

KMPC uses the model of the form (3.4) learned by EDMD as described in Section6.3.1. The control algorithm is formulated as a quadratic problem with the following structure:

$$\min_{\tilde{\mathbf{U}}_t} \quad J(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_{t-1}, \mathbf{Y}_t^{ref}, \tilde{\mathbf{U}}_t) \tag{6.4a}$$

$$\text{s.t.} \quad \mathbf{z}_{k+1} = A\mathbf{z}_k + B\mathbf{u}_k, \tag{6.4b}$$

$$\mathbf{y}_k = C\mathbf{z}_k, \tag{6.4c}$$

$$\tilde{\mathbf{u}}_{min} \leq \tilde{\mathbf{u}}_k \leq \tilde{\mathbf{u}}_{max}, \tag{6.4d}$$

$$\Delta\tilde{\mathbf{u}}_{min} \leq \Delta\tilde{\mathbf{u}}_k \leq \Delta\tilde{\mathbf{u}}_{max}, \tag{6.4e}$$

$$\delta_{min} \leq \delta_{sw,k} \leq \delta_{max}, \tag{6.4f}$$

$$\mathbf{z}_t = \Phi\left(\tilde{\mathbf{x}}(t)\right), \tag{6.4g}$$

$$\tilde{\mathbf{u}}_{t-1} = \tilde{\mathbf{u}}(t-1). \tag{6.4h}$$

In the equations (6.4g) and (6.4h), the variables $\tilde{\mathbf{x}}(t)$ and $\tilde{\mathbf{u}}(t-1)$ represent the current extended state measurement or the extended control input that was applied in the previous time step. In addition, $\Delta\tilde{\mathbf{u}}_k = \tilde{\mathbf{u}}_k - \tilde{\mathbf{u}}_{k-1}$ denotes the change in the extended input. The extended control input sequence is referred to as $\tilde{\mathbf{U}}_t = [\tilde{\mathbf{u}}_t, \tilde{\mathbf{u}}_{t+1}, ...\tilde{\mathbf{u}}_{t+N-1}]$. The main difference between the KMPC and the NMPC described in the equations (6.3a) - (6.3g) (and already mentioned in previous chapters) is that the KMPC leads to a quadratic optimization problem, which is convex and generally easier to solve. However, since the Koopman-based model is an approximation of the nonlinear model, a higher prediction error is to be expected. In addition, the extended state-space is often much larger than the original one, which can lead to problems with certain KMPC formulations. To solve this problem, a dense MPC formulation is used, as described in Section2.3.3, to mitigate the problems arising from the high dimensionality of the system.

6.4.3 *Cost function and constraints*

The goal of the controller is to find a balance between minimizing the deviations of the system state from the reference trajectory and minimizing the control effort, which represents the total energy consumption. The cost function (6.3a) can be expressed as follows:

$$J(\mathbf{x}_t, \mathbf{u}_{t-1}, \mathbf{Y}_t^{ref}, \mathbf{U}_t) = \quad J_{t+N}\left(\mathbf{x}_{t+N}, \mathbf{y}_{t+N}^{ref}\right) + \sum_{k=t}^{t+N-1} J_k\left(\mathbf{x}_k, \mathbf{u}_k, \Delta\mathbf{u}_k, \mathbf{y}_k^{ref}\right), \tag{6.5}$$

where

$$J_k\left(\mathbf{x}_k, \mathbf{u}_k, \Delta\mathbf{u}_k, \mathbf{y}_k^{ref}\right) = \left\|\mathbf{y}_k - \mathbf{y}_k^{ref}\right\|_Q^2 + \left\|\mathbf{u}_k\right\|_R^2 + \left\|\Delta\mathbf{u}_k\right\|_{R_\Delta}^2 \tag{6.6}$$

represents the stage cost, and

$$J_{t+N}\left(\mathbf{x}_{t+N}, \mathbf{y}_{t+N}^{ref}\right) = \left\|\mathbf{y}_{t+N} - \mathbf{y}_{t+N}^{ref}\right\|_P^2 \tag{6.7}$$

is the terminal cost. In these equations, Q, R, R_Δ and P are positive definite weight matrices. The system states being tracked in this case are the longitudinal velocity, the yaw angle and the lateral position in the global coordinate system. These states can be obtained from the full state variables by a linear transformation, namely $\mathbf{y} = C_{nl}\mathbf{x} = [v_x\ \theta_z\ Y]^T$, where C_{nl} represents the corresponding selection matrix.
The cost function (6.4a) in KMPC is defined in the same way as (6.5), with the following differences in the notation: the state vector $\mathbf{x}$ is replaced by the extended state vector $\tilde{\mathbf{x}}$, the input vector $\tilde{\mathbf{u}}$ is replaced by the extended input vector $\tilde{\mathbf{x}}$ and the output equation is $\mathbf{y} = C\mathbf{z}$.

The constraints for the control inputs are determined by the actuator dynamics and can be summarized as follows:

$$\begin{aligned} \delta_{fmin} \le \delta_f \le \delta_{fmax}, \\ T_{min} \le T_{\bullet\star} \le T_{max}, \\ \Delta\delta_{fmin} \le \Delta\delta_f \le \Delta\delta_{fmax}, \\ \Delta T_{min} \le \Delta T_{\bullet\star} \le \Delta T_{max}. \end{aligned} \tag{6.8}$$

To ensure feasibility and computational efficiency, the system states are unconstrained and kept within a feasible working region by enforcing reference tracking through the definition of the cost function.

6.5 EXPERIMENTAL RESULTS AND DISCUSSION

6.5.1 *Controller setup*

The control algorithms are exectuted on the *dSPACE MicroLabBox* platform and solved with the FORCESPRO solver [117], [118]. The sampling time is set to $T_s = 50$ ms, while the weight matrices and constraints are as follows:

$$\begin{aligned} Q = \mathrm{diag}\,(1, 5, 20), \quad R = \mathrm{diag}\,(2, 25, 25, 25, 25), \\ R_\Delta = \mathrm{diag}\,(5, 100, 100, 100, 100), \\ \Delta\delta_{fmax} = -\Delta\delta_{fmin} = 54.55\ ^\circ, \\ T_{max} = -T_{min} = 50\ \mathrm{mNm}, \\ \Delta T_{max} = -\Delta T_{min} = 12.5\ \mathrm{mNm}. \end{aligned}$$

Although the terminal weight matrix P is normally used to approximate the costs for the infinite horizon, P is chosen here as $P = Q$ for the sake of simplicity.

The NMPC optimization problem (6.3) is solved in real time using the sequential quadratic programming (SQP) algorithm. The discretization is performed using the FORCESPRO *continuous-time dynamics* equality option, with the integrator type set to *ERK4* and the number of intermediate integration nodes set to 60. The KMPC optimization problem (6.4) is formulated with YALMIP [86] and solved with the primal-dual interior-point method. In both cases, the optimization of the linear algebra operations is used to accelerate the code execution on the *dSPACE* platform.

6.5.2 *Multiple lane change manoeuvre*

Figure6.12and Figure6.14illustrate the performance of reference tracking and control inputs during a user-defined test manoeuvre referred to as a *multiple lane change* (MLC). This manoeuvre is defined by the following expressions:

$$\begin{aligned} Y_{ref} &= -\frac{Y_{max}}{2}(2\tanh(\kappa X) - \tanh(4\kappa(X - dX_1)) + \tanh(4\kappa(X - dX_1 - dX_2))), \\ \theta_{z,ref} &= \arctan\left(\frac{\dot{Y}_{ref}}{\dot{X}}\right). \end{aligned} \tag{6.9}$$

Here the distance parameters are defined as $Y_{max} = 0.15$ m, $dX_1 = 15$ m, $dX_2 = 5$ m, and the curvature factor is $\kappa = 2\ \mathrm{m}^{-1}$. The purpose of this manoeuvre was to simulate a lane change in the initial section, followed by a fast obstacle avoidance in the second section. The experiment is performed for $v_x \approx 1$ m/s and $v_x \approx 2$ m/s with prediction horizon $N = 5$ and steering angle limited to $\delta_{fmax} = -\delta_{fmin} = 10\ °$.

The right-hand plots in Figure6.13and Figure6.15show the differential torque T_{diff}, i.e. the difference between the torques acting on the right and left wheels. This difference leads to an additional yaw moment of the vehicle (as previously defined in Section6.3.3). The similarity with the steering angle graph indicates that the steering system and torque vectoring work together to control the yaw angle of the vehicle. Figure6.12shows that NMPC has a larger overshoot when tracking the longitudinal velocity $v_x \approx 1$ m/s and the lateral position, but overall behaves quite similarly to KMPC. One can see that the references for NMPC and KMPC are not identical, which is due to the fact that they are influenced by the higher-level PI position controller.

Larger discrepancies arise when the same test manoeuvre is performed at a longitudinal velocity of $v_x \approx 2$ m/s, as in Figure6.14. In this case, the overshoot of the NMPC is much larger than that of the KMPC when tracking all three output signals, which is probably due to an increased numerical sensitivity to environmental disturbances, such as vibrations of the treadmill, that occur at higher velocity. The corresponding input signals are given in Figure6.15and show an increased steering wheel and differential torque effect in the case of the NMPC, confirming previous observations.

The described behaviour is quantitatively supported in the Table6.6. In it, the costs of the KMPC control loop are lower than those of the NMPC for both test cases. In addition, the execution times of KMPC are about 10 times lower than those of NMPC. Due to these two characteristics, KMPC is more suitable for real-time predictive control.

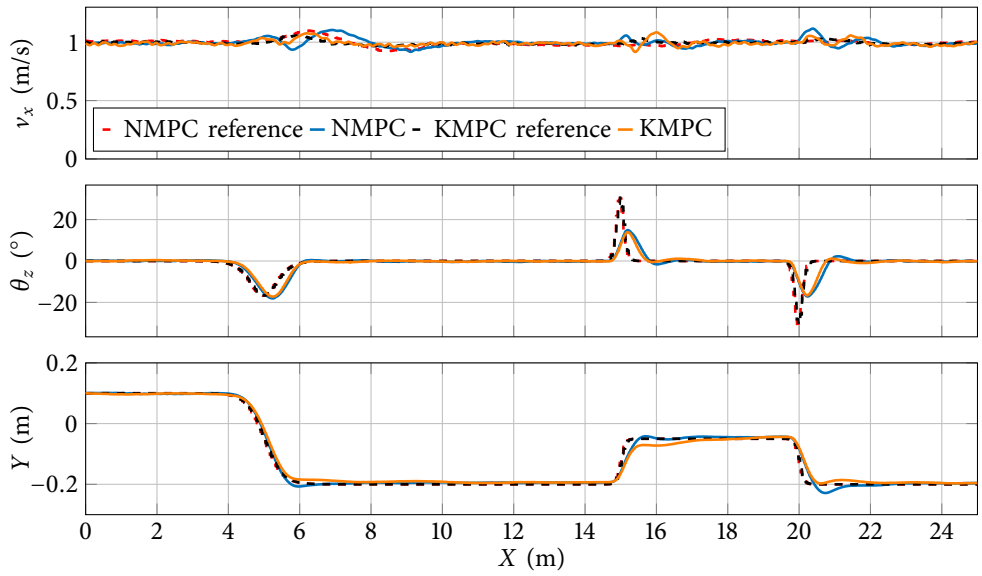

Figure 6.12: Tracked states during MLC manoeuvre for $N = 5$ at $v_x \approx 1$ (m/s).

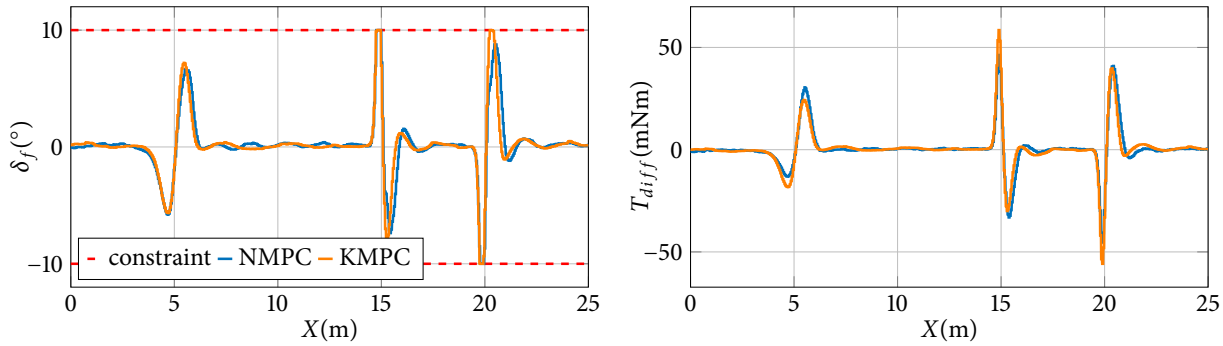

Figure 6.13: Inputs during MLC manoeuvre for $N = 5$ at $v_x \approx 1$ (m/s).

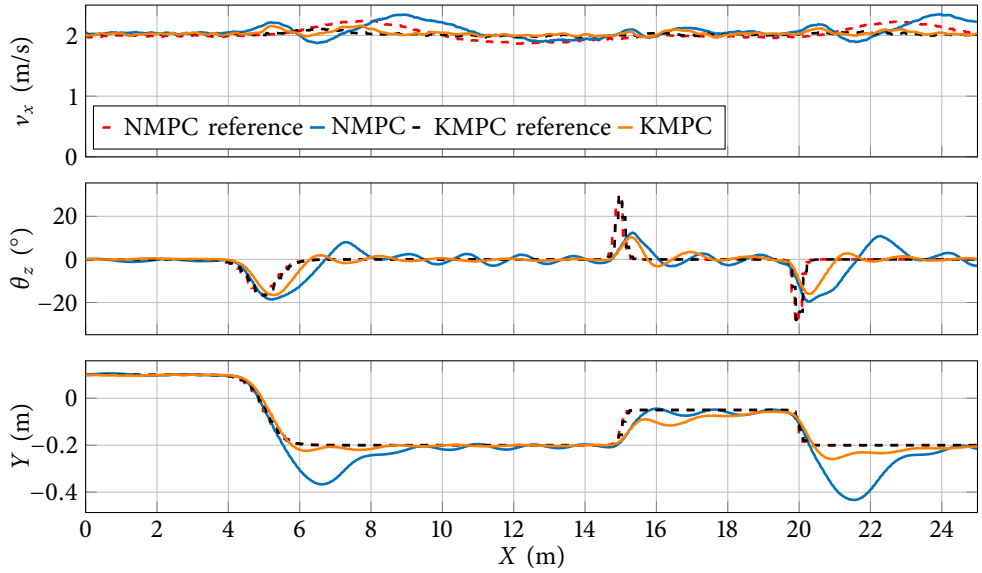

Figure 6.14: Tracked states during MLC manoeuvre for $N = 5$ at $v_x \approx 2$ (m/s).

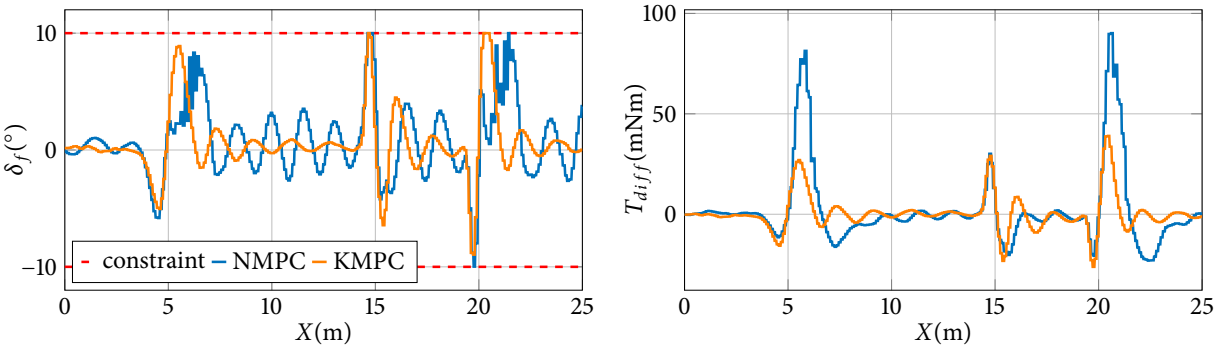

Figure 6.15: Inputs during MLC manoeuvre for $N = 5$ at $v_x \approx 2$ (m/s).

Table 6.6: MLC manoeuvre closed-loop costs and execution times for $N = 5$

Velocity		$v_x \approx 1$(m/s)		$v_x \approx 2$(m/s)	
Controller		NMPC	KMPC	NMPC	KMPC
Cost		16.401	16.096	42.618	12.393
Execution time [ms]	Mean	21.1	2.1	21.5	2.1
	Median	21.0	2.1	21.5	2.1
	Max	22.8	2.8	23.5	2.5
	Min	20.5	1.9	20.7	1.9

6.5.3 *Double lane change manoeuvre*

The second manoeuvre chosen was a double lane change described in [131]. The desired orientation and lateral position of the vehicle were governed by the following set of equations:

$$\begin{aligned} Y_{ref} &= 0.1 \arctan(\Phi), \\ \theta_{z,ref} &= \frac{D_{y_1}}{2}(\tanh(Z_1)+1) - \frac{D_{y_2}}{2}(\tanh(Z_2)+1), \\ Z_1 &= \frac{S}{d_{x_1}}(X - X_{s_1}) - \frac{S}{20}, \\ Z_2 &= \frac{S}{d_{x_2}}(X - X_{s_2}) - \frac{S}{20}, \\ \Phi &= \frac{1.2 D_{y_1}}{D_{x_1}}\left(\frac{1}{\cosh(Z_1)}\right)^2 - \frac{1.2 D_{y_2}}{D_{x_2}}\left(\frac{1}{\cosh(Z_2)}\right)^2, \end{aligned} \tag{6.10}$$

where $S = 24$, $D_{x_1} = 25$, $D_{x_2} = 21.95$, $D_{y_1} = 4.05$, $D_{y_2} = 5.7$, $X_{s_1} = 5.719$ and $X_{s_2} = 8.645$. As in the previous section, the test is performed for $v_x \approx 1$ m/s and $v_x \approx 2$ m/s, the prediction horizon is $N = 5$ and the steering angle is constrained to $\delta_{fmax} = -\delta_{fmin} = 10$ °.

Figure6.16shows the state tracking for both NMPC and KMPC at a longitudinal velocity of $v_x \approx 1$ m/s. Apart from minor differences in the longitudinal velocity tracking, the responses of both controllers look almost the same. However, it can be seen in the Figure6.17that the NMPC uses a larger differential torque to steer the vehicle along the desired path, which consequently leads to a velocity increase.

When the same manoeuvre is repeated at $v_x \approx 2$ m/s, the tracking performance of the NMPC deteriorates, as yaw angle fluctuations and lateral overshoot of the position occur (or rather undershoot, depending on how one looks at it). This behaviour can be seen in Figure6.18. The

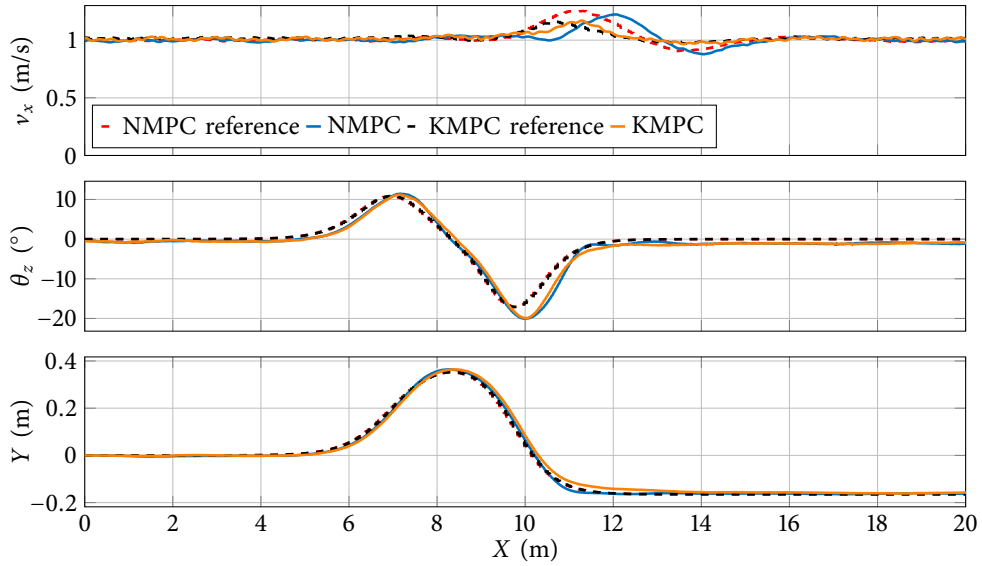

Figure 6.16: Tracked states during DLC manoeuvre for $N = 5$ at $v_x \approx 1$ (m/s).

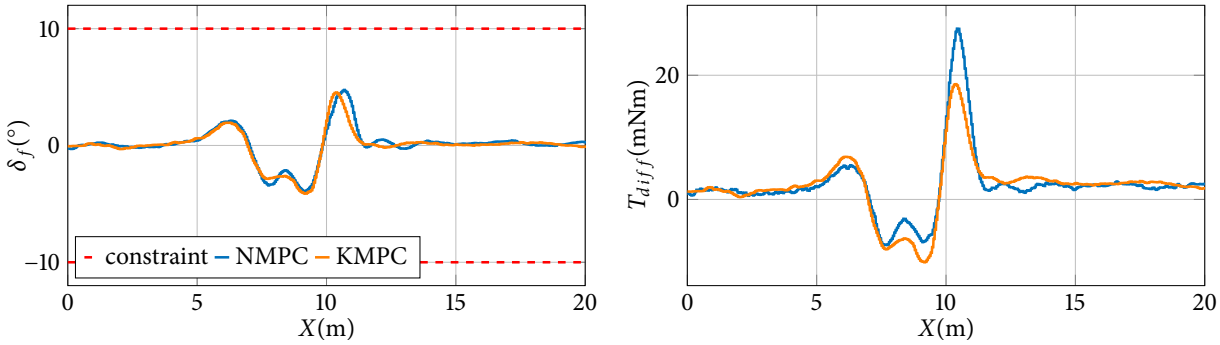

Figure 6.17: Inputs during DLC manoeuvre for $N = 5$ at $v_x \approx 1$ (m/s).

oscillations and undershoot are caused by steering oscillations and a large differential torque, as shown in Figure6.19. A direct analysis of the tracking performance of the KMPC cannot be carried out using the graphs provided.

The results from the figures are supported by the numbers from Table6.7, which also shows the already mentioned performance degradation of the NMPC with increasing velocity. Additionally, the closed-loop cost analysis indicates performance improvement for KMPC with the velocity increase. For $v_x \approx 1$ m/s, the performance of both controllers is almost the same.

In terms of execution time, the results confirm what is mentioned for the MLC experiment, i.e. the computational speed of KMPC is approximately 10 times higher than that of NMPC for both test runs.

6.5.4 *Prediction horizon change effect*

In this section, the effect of changing the length of the prediction horizon for KMPC is investigated. The experiment was done at $v_x \approx 1.5$ m/s for prediction horizon of $N = 5$ and $N = 10$. The steering angle limit was lowered to $\delta_{fmax} = -\delta_{fmin} = 3$ ° to force the torque vectoring system to engage. The same test is omitted for NMPC as it cannot run in real time for the prediction horizon $N = 10$.

In Figure6.20it is shown that the vehicle tracks the desired references better with a larger

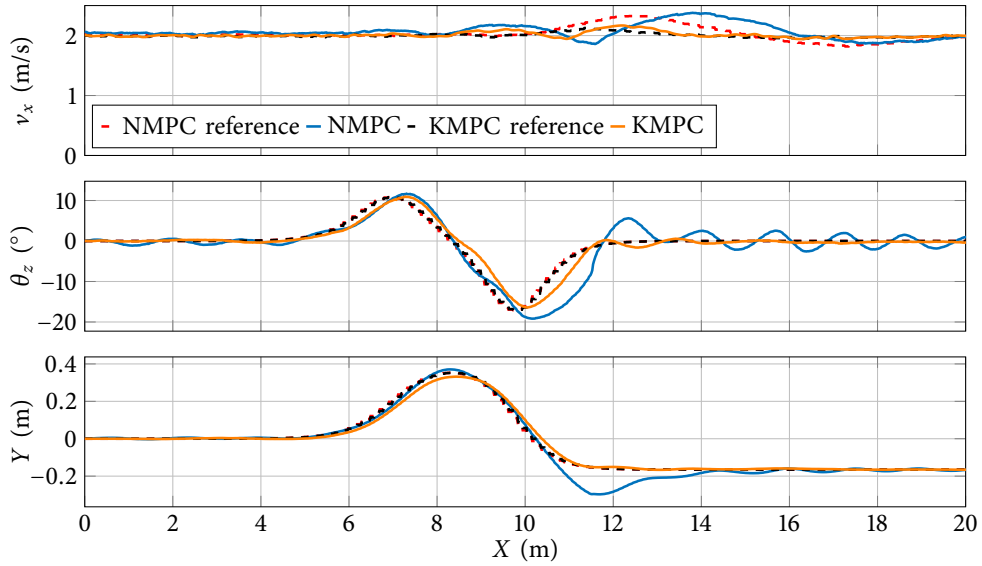

Figure 6.18: Tracked states during DLC manoeuvre for $N = 5$ at $v_x \approx 2$ (m/s).

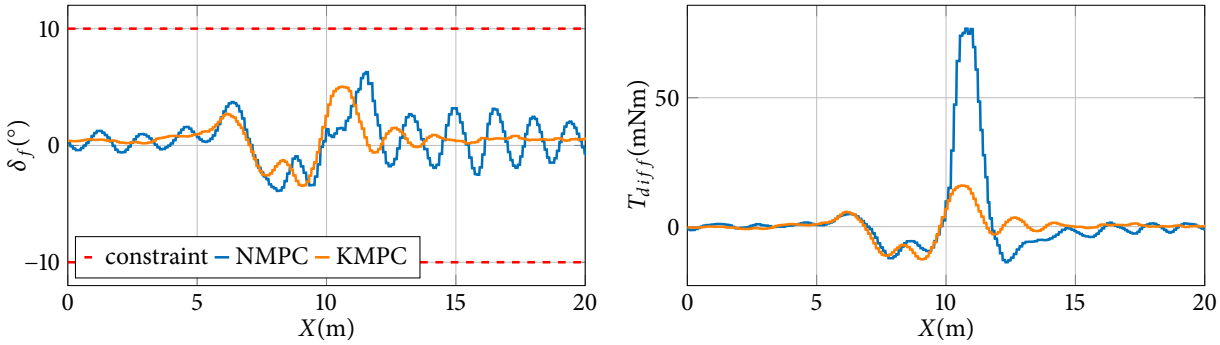

Figure 6.19: Inputs during DLC manoeuvre for $N = 5$ at $v_x \approx 2$ (m/s).

Table 6.7: DLC manoeuvre closed-loop costs and execution times for $N = 5$

Velocity		$v_x \approx 1$(m/s)		$v_x \approx 2$(m/s)	
Controller		NMPC	KMPC	NMPC	KMPC
Cost		5.469	5.414	13.326	3.981
Execution time [ms]	Mean	20.9	2.1	21.3	2.1
	Median	20.9	2.1	21.3	2.1
	Max	21.7	2.4	22.5	2.3
	Min	20.4	1.9	20.6	1.9

prediction horizon, which is to be expected. This is illustrated by Figure6.21, which shows how the controller with $N = 10$ provides a larger differential torque when the steering angle limit is reached and allows the vehicle to stay on the desired path. Consequently, the closed-loop cost decreases as the prediction horizon increases, as indicated in the Table6.8. On the contrary, as the prediction horizon increases, computational time also increases. In this case, it increased more than 10 times on average.

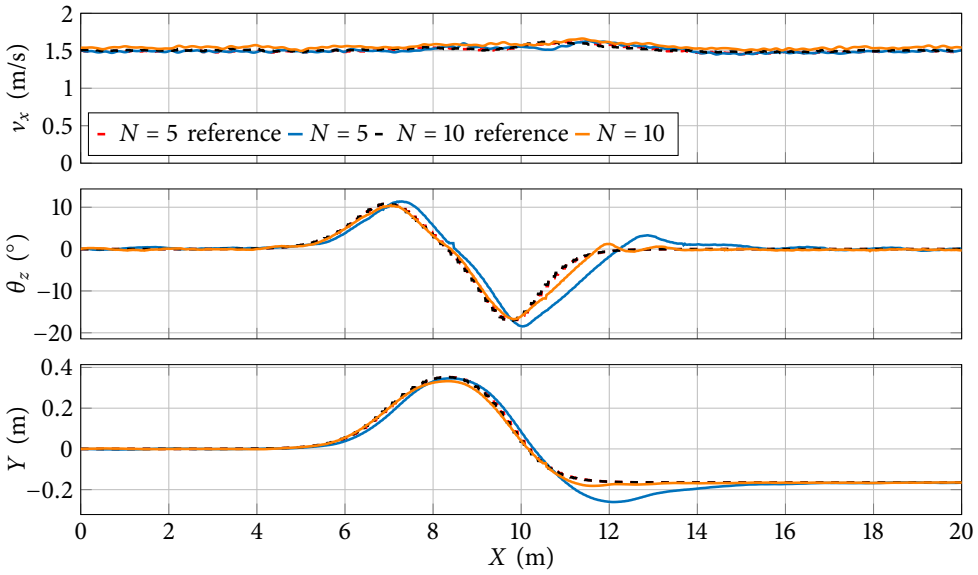

Figure 6.20: Tracked states during DLC manoeuvre for KMPC with $N = 5$ and $N = 10$ at $v_x \approx 1.5$ (m/s).

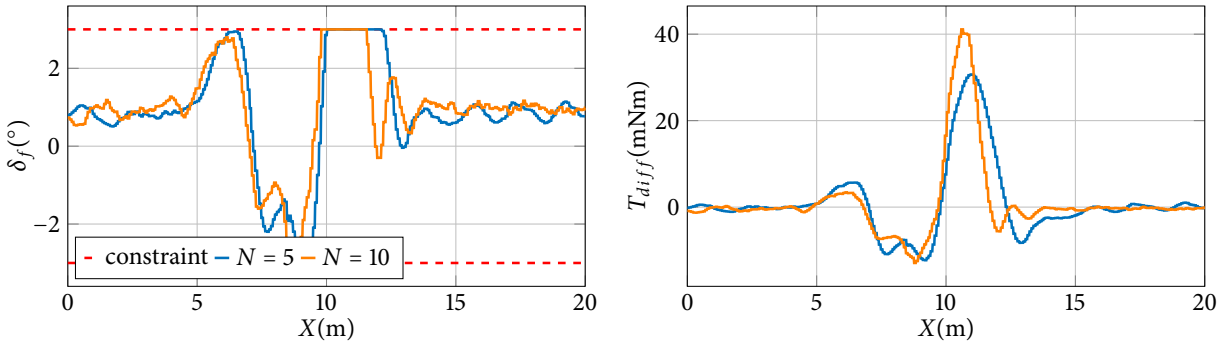

Figure 6.21: Inputs during DLC manoeuvre for KMPC with $N = 5$ and $N = 10$ at $v_x \approx 1.5$ (m/s).

Table 6.8: DLC manoeuvre comparison for KMPC with $N = 5$ and $N = 10$ at $v_x \approx 1.5$(m/s)

Prediction horizon		$N = 5$	$N = 10$
Cost		8.716	3.094
Execution time [ms]	Mean	2.1	23.5
	Median	2.1	23.5
	Max	2.6	29.9
	Min	1.9	21.4

6.5.5 *Sensitivity to delays*

The information that was not mentioned before is that the experimental setup has a delay of approximately $T_{delay} \approx 100$ ms in the control loop, which is caused by the discretization and the communication channels. In this section, the consequences of this delay are investigated by comparing some of the experimental results with those of the corresponding simulations. Specifically, both MLC and DLC manoeuvre tests are simulated for the default horizon $N = 5$ and longitudinal velocity $v_x \approx 2$ m/s and their output responses and closed-loop costs are reported.

Figure6.22shows the results for the MLC manoeuvre for NMPC and KMPC. The experimental results are compared with the simulation results with and without the delay T_{delay}. First of all, it

is observed that the performance of both controllers is different in simulation and experiment, which is probably due to unmodeled dynamics and environmental influences such as vibrations of the treadmill. In addition, the responses of NMPC and KMPC deteriorate when the delay is added to the simulation and are slightly more similar to the experimental results. However, the NMPC leads to greater overshoot (especially in the experiment) in the longitudinal velocity and lateral position, as well as oscillations in the steady state, which does not occur with the KMPC.

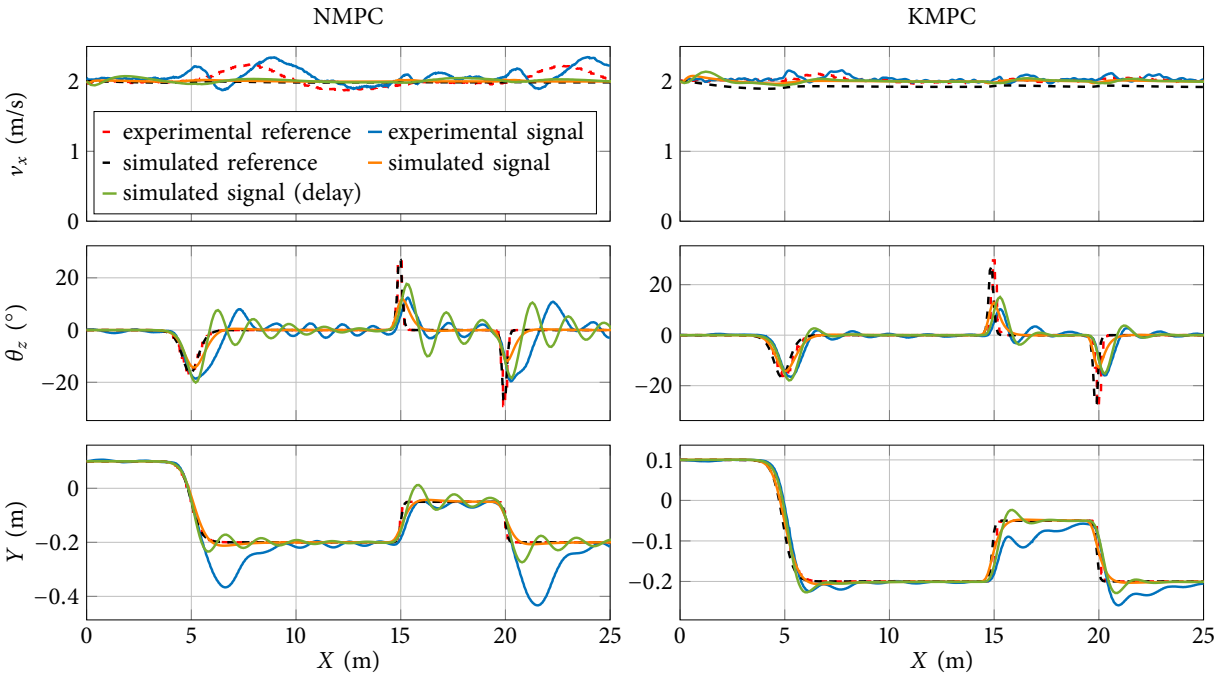

Figure 6.22: Experiment and simulation comparison of MLC manoeuvre for NMPC (left) and KMPC (right) with $N = 5$ at $v_x \approx 2$ (m/s).

The results for the DLC manoeuvre are shown in Figure6.23, where a similar behaviour can be seen. The simulation results without delay are similar for both NMPC and KMPC, while in the results of the simulation with delay and the experiment NMPC again induces oscillations. Especially in the experiment, it also causes larger overshoots of v_x and Y.

The closed-loop costs are listed in Table6.9and also illustrate the described behaviour. The results of the simulation without delay indicate that NMPC performs better than KMPC for both manoeuvres, which makes sense since the Koopman model is only an approximation of the corresponding nonlinear model. On the other hand, the results from the simulation with delay and the experimental results confirm what has already been said in previous sections and show that NMPC can perform much worse than KMPC. This indicates the sensitivity of NMPC to system delays. However, a more detailed investigation of this effect is beyond the scope of this thesis.

Table 6.9: Experiment and simulation closed-loop cost comparison

manoeuvre	Simulation		Simulation with delay		Experiment	
	NMPC	KMPC	NMPC	KMPC	NMPC	KMPC
MLC	8.934	10.227	21.687	15.827	42.618	12.393
DLC	3.475	5.817	4.127	6.401	13.326	3.981

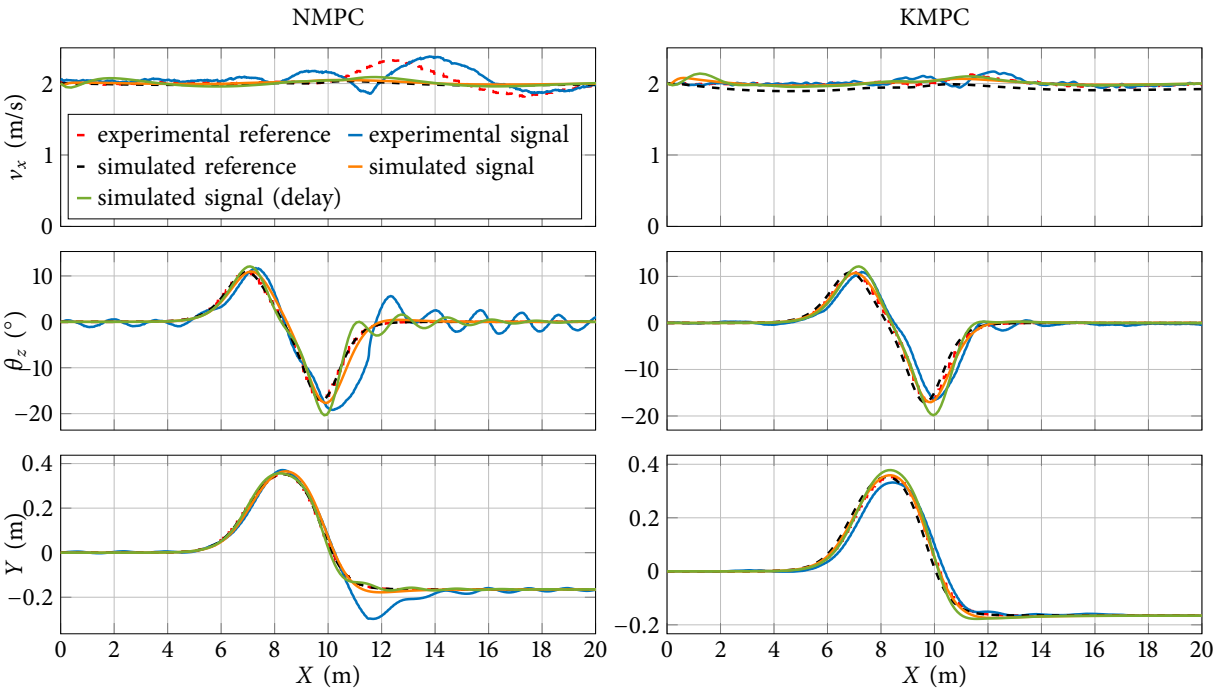

Figure 6.23: Experiment and simulation comparison of DLC manoeuvre for NMPC (left) and KMPC (right) with $N = 5$ at $v_x \approx 2$ (m/s).

6.5.6 *Concluding remarks*

The overall conclusion of this analysis is that KMPC can outperform NMPC in terms of stability and efficiency, which is particularly noticeable when large time delays are present in the system. While NMPC has greater potential under simulated ideal conditions, it is less effective under real-world conditions due to higher computational demand and communication delays. KMPC, on the other hand, demonstrates lower closed-loop costs and significantly faster execution times, making it more suitable for real-time applications. Even if the prediction horizon is extended, which improves the tracking performance of KMPC, the increase in computation time remains within acceptable limits for real-time operation.

To summarise, despite the theoretical advantages of NMPC, KMPC is a more feasible, robust and computationally efficient solution for real-time control applications, especially under conditions with higher speeds and communication delays.

6.6 SUMMARY

This chapter focuses on the evaluation of the Kooopman-based MPC in a real-world test setup. The experimental validation of the control algorithms is performed using a scaled vehicle model and a treadmill that serves as a road simulator. These innovative techniques allow testing under different road conditions without the logistical challenges of a real (race) track. The setup includes a scaled vehicle on a treadmill monitored by an optical tracking system and wheel speed sensors. A control algorithm running on a *dSPACE MicroLabBox* platform evaluates the vehicle's performance based on input from these sensors.

The chapter further deals with the identification of vehicle parameters using experimental data and the MATLAB System Identification Toolbox. This method is used to determine critical parameters such as tire stiffness, rolling resistance or tire friction, which are essential for the accurate simulation of vehicle dynamics but can be difficult to measure directly.

After the identification process, a Koopman model is learned using the EDMD algorithm. In this case, the data for learning the Koopman model was generated by simulating multiple trajectories with the previously identified nonlinear vehicle model. Different datasets, including real experimental data and simulations, are used to train and validate these models and compare their effectiveness in different driving scenarios.

Two MPC strategies, namely NMPC and KMPC, are formulated for the application of torque vectoring. These strategies aim to optimize the driving performance of the vehicle by adjusting the torque distribution between the wheels considering constraints such as actuator dynamics. The performance of these MPC strategies is evaluated through experimental tests, including multiple and double lane change manoeuvres, under different conditions.

The experiments reveal the strengths and limitations of NMPC and KMPC. While NMPC offers theoretical advantages in terms of control accuracy, in practice it faces problems such as computational complexity and sensitivity to time delays, especially at higher speeds. On the other hand, KMPC shows promising results in real-time applications, which are characterized by lower computational complexity and robustness to time delays.

In conclusion, this chapter presents a comprehensive approach for the development and testing of advanced vehicle dynamics control systems. It highlights the potential of using scale models and road simulators for experimental validation, the importance of accurate identification of vehicle parameters, and the effectiveness of Koopman-based MPC strategies in improving vehicle handling and performance.

7

Conclusion and future research directions

This thesis introduces a novel approach to advanced vehicle dynamics control systems, focusing on the integration of the Koopman operator with model predictive control (MPC). Each section builds on a comprehensive framework that aims to improve both the understanding and practical application of Koopman operator-based vehicle control, especially in real-time scenarios.

The thesis begins by providing a theoretical basis for key topics in Chapter2. First, models of vehicle dynamics such as the two-track and bicycle models were examined, with aspects such as tire models and alternative slip formulations. The intricacies of model predictive control were then explained, distinguishing between linear and nonlinear MPC and exploring both dense and sparse formulations. In addition, key vehicle dynamics control systems such as ABS, ESC and torque vectoring were introduced and their important role in improving vehicle safety and performance was emphasized. Furthermore, the thesis addresses the Koopman operator, which provides a method for converting nonlinear dynamics into linear representations, although its use is limited by its infinite-dimensional scope. The overview of the theoretical background provides essential tools for the discussions in the rest of the thesis.

The first contribution is the development of a method for identifying a vehicle dynamics model based on the Koopman operator suitable for applications in predictive control algorithms. To achieve this, three different numerical methods were used to approximate the Koopman operator: extended dynamic mode decomposition (EDMD), deep dynamic mode decomposition (Deep-DMD) and a newly proposed method, called enhanced extended dynamic mode decomposition (E^2DMD). This novel method incorporates an affine transform to reduce the dimensionality of the lifted state vector in traditional EDMD, maintaining model accuracy while simplifying the process and reducing the hyperparameter complexity compared to Deep-DMD. Three different numerical approaches for the implementation of E^2DMD were elaborated: basis function reduction by discrete selection (E^2DMD-DS), which selects specific elements from the basis function vector; a multiple step prediction learning algorithm (E^2DMD-MS), which modifies the encoder structure similarly to Deep-DMD; and basis function reduction as a hyperparameter optimization problem (E^2DMD-HO), which integrates well with existing hyperparameter optimization frameworks. The evaluation of these methods by simulations with the Van der Pol oscillator, the damped Duffing oscillator and the bilinear motor yielded several conclusions. First, there was a consistent improvement in prediction performance with increasing dimensionality of the state-space. However, unexpected fluctuations in the performance of Deep-DMD point to possible architectural improvements from which the E^2DMD-MS method could also benefit. Furthermore,

the hypothesis that E²DMD simplifies the reduction of the lifted state-space compared to manual selection in EDMD was confirmed. It also contains fewer hyperparameters than Deep-DMD, which facilitates fine-tuning. Finally, the results indicate that the effectiveness of each method varies depending on the dynamical system, suggesting that the choice of numerical method should be tailored to the system under investigation.

Following the initial tests performed with benchmark dynamical models, the application of the Koopman operator, in particular using the EDMD algorithm, to improve the control of vehicle dynamics by using simple models is presented in Chapter4. Steering angle and longitudinal slip were used as input signals. First, the Koopman operator was applied to a bicycle model, where EDMD successfully approximates a higher-dimensional linear model that outperforms traditional linearization methods in predicting system trajectories. The effectiveness of this model was further validated by integrating it into an MPC setup, highlighting its potential for vehicle motion control. The second part of the chapter deals with a two-track vehicle model using the Koopman operator with an MPC algorithm for torque vectoring. The EDMD-based model showed superior accuracy and efficiency in predictive control and outperforms the linear time-varying MPC (LTV-MPC) in various metrics in test scenarios. These results underline the effectiveness of Koopman MPC in vehicle dynamics control, which offers significant advantages over conventional methods. This result relates to both the first and second contributions, which involves the development of an MPC algorithm for wheel torque distribution using a vehicle model identified with the Koopman operator. However, the contributions are not fully addressed here, as the reliance on simplified models highlights the need for additional research to confirm the validity of these methods in more complex, real-world scenarios.

The validation in more complex scenarios is carried out in Chapter5and Chapter6. First, through detailed simulations in MATLAB Simulink and CarMaker, strategies such as E²DMD-MPC, Deep-DMD-MPC, EDMD-MPC, LTV-MPC and NMPC were investigated for their efficiency and control performance in dynamic manoeuvres and experiments at the Nürburgring and Hockenheimring race tracks. NMPC showed superior manoeuvreing capabilities at the cost of higher computational effort, especially at longer prediction horizons, illustrating the classic trade-off between computational efficiency and control performance. Conversely, Koopman-based controllers, namely E²DMD-MPC and Deep-DMD-MPC, offered a promising balance by achieving competitive control performance at significantly lower computational cost, making them suitable for scenarios with real-time requirements. Deep-DMD-MPC performed particularly well at longer prediction horizons, although there is some variability in performance at shorter horizons. While LTV-MPC is less computationally intensive than NMPC, it generally performed worse than all other MPC versions, except at low speeds and predictable conditions.

The experiments with real-world scenarios were conducted on a test setup that includes a scaled vehicle model on a treadmill that mimics real road conditions, without the logistical overhead of actual track testing. The conclusions drawn here differ from those based on simulation results. Despite the theoretical advantages of NMPC under ideal conditions, its practical application is hindered by its high computational cost and sensitivity to time delays (due to the discretization of communication), making it less suitable for real-world scenarios in its current form. In contrast, KMPC not only offers faster execution times, but is also more robust to time delays and scales well with extended prediction horizons while remaining within feasible limits for real-time operation. This may seem counterintuitive at first glance and definitely requires further investigation.

Finally, the third contribution, namely a technique for generating a Koopman operator learning dataset by first creating a nonlinear model from experimental data and then simulating different scenarios with this model, was also applied in Chapter5, while a more detailed presentation is given in Chapter6. The analysis supported by figures and tables shows that the Koopman models derived from simulation data generally perform better than the models based on experimental data, with the exception of the evaluations on the experimental data itself. These results confirm the effectiveness of simulation-based models in capturing nonlinear, scaled vehicle dynamics and validate their application in the development of control systems. They also support the approach of replacing small experimental datasets with simulated data to increase the size and improve the information content of the dataset, which in turn increases the accuracy of the models trained on it. In this context, the identification of nonlinear models acts as a structured method for data compression, where the experimental data is compressed into a model and other datasets are generated from this model, which is similar to a process of data decompression by nonlinear model simulation.

Overall, this thesis highlights the critical importance of choosing the right control strategy that not only takes into account the nonlinearity of vehicle dynamics, but also aligns with the computational constraints of real-time applications. The insights gained here emphasise the potential of Koopman-based controllers as a viable middle ground that offers a strategic compromise between the high control performance of NMPC and the necessary computational efficiency for practical implementation. Furthermore, some novel Koopman operator identification methods as well as a new approach to generate learning data by using nonlinear model identification as an intermediate step are presented.

Future research could focus on further improving the proposed E^{2}DMD and Deep-DMD methods as well as the numerical algorithms used to learn such models. It would also be valuable to further explore the dataset generation process discussed in Section6.3.3and investigate how to create a dataset of minimal size that is still sufficiently informative to accurately identify a Koopman model based on its samples. Furthermore, future studies may focus on the integration of physics-informed machine learning (PIML) with the Koopman operator framework to improve the accuracy of complex vehicle dynamics models. The main goal of PIML is to develop machine learning models that integrate physical laws as constraints, ensuring compliance with the fundamental principles of the systems under investigation. Such approaches have already been explored in control engineering [132], for modelling vehicle dynamics [133] and even in the context of the Koopman operator [134, 135]. The combination of PIML with the Koopman operator could lead to an improvement in the prediction accuracy, computational efficiency and interpretability of Koopman-based models, which would undoubtedly benefit the field of modern vehicle control systems.

BIBLIOGRAPHY

[1] R. R. Kumar and K. Alok, "Adoption of electric vehicle: A literature review and prospects for sustainability," *Journal of Cleaner Production*, vol. 253, p. 119911, 2020.

[2] M. Muratori, M. Alexander, D. Arent, M. Bazilian, P. Cazzola, E. M. Dede, J. Farrell, C. Gearhart, D. Greene, A. Jenn *et al.*, "The rise of electric vehicles—2020 status and future expectations," *Progress in Energy*, vol. 3, no. 2, p. 022002, 2021.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[4] D. Schramm, M. Hiller, and R. Bardini, "Vehicle dynamics," *Modeling and Simulation. Berlin, Heidelberg*, vol. 151, 2014.

[5] F. Micheli, M. Bersani, S. Arrigoni, F. Braghin, and F. Cheli, "NMPC trajectory planner for urban autonomous driving," *Vehicle System Dynamics*, pp. 1–23, 2022.

[6] H. Pacejka, *Tire and vehicle dynamics*, 3rd ed. Elsevier, 2012.

[7] W. Hirschberg, G. Rill, and H. Weinfurter, "Tire model TMeasy," *Vehicle System Dynamics*, vol. 45, no. Sup. 1, pp. 101–119, 2007.

[8] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle System Dynamics*, vol. 21, no. Sup. 001, pp. 1–18, 1992.

[9] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[10] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[11] S. V. Rakovi ć and W. S. Levine, *Handbook of model predictive control*. Springer, 2018.

[12] M. N. Zeilinger, "Real-time model predictive control," Ph.D. dissertation, ETH Zurich, 2011.

[13] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[14] L. Grüne, "NMPC without terminal constraints," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 1–13, 2012.

[15] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available:https://www.gurobi.com

[16] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020. [Online]. Available:https://doi.org/10.1007/s12532-020-00179-2

[17] I. M. Bomze, V. F. Demyanov, R. Fletcher, T. Terlaky, I. Pólik, and T. Terlaky, "Interior point methods for nonlinear optimization," *Nonlinear Optimization: Lectures given at the CIME Summer School held in Cetraro, Italy, July 1-7, 2007*, pp. 215–276, 2010.

[18] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[19] A. Grancharova and T. A. Johansen, *Explicit nonlinear model predictive control: Theory and applications*. Springer Science & Business Media, 2012, vol. 429.

[20] D. Limon, J. Calliess, and J. M. Maciejowski, "Learning-based nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7769–7776, 2017.

[21] D. Masti and A. Bemporad, "Learning nonlinear state-space models using autoencoders," *Automatica*, vol. 129, p. 109666, 2021.

[22] M. N. Zeilinger, M. Morari, and C. N. Jones, "Soft constrained model predictive control with robust stability guarantees," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1190–1202, 2014.

[23] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proceedings of the national academy of sciences of the united states of america*, vol. 17, no. 5, p. 315, 1931.

[24] B. O. Koopman and J. v. Neumann, "Dynamical systems of continuous spectra," *Proceedings of the National Academy of Sciences*, vol. 18, no. 3, pp. 255–263, 1932.

[25] I. Mezić and A. Banaszuk, "Comparison of systems with complex behavior," *Physica D: Nonlinear Phenomena*, vol. 197, no. 1-2, pp. 101–133, 2004.

[26] I. Mezić, "Spectral properties of dynamical systems, model reduction and decompositions," *Nonlinear Dynamics*, vol. 41, pp. 309–325, 2005.

[27] M. Budišić, R. Mohr, and I. Mezić, "Applied koopmanism," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, 2012.

[28] I. Mezić, "Analysis of fluid flows via spectral properties of the Koopman operator," *Annual review of fluid mechanics*, vol. 45, pp. 357–378, 2013.

[29] Y. Lan and I. Mezić, "Linearization in the large of nonlinear systems and Koopman operator spectrum," *Physica D: Nonlinear Phenomena*, vol. 242, no. 1, pp. 42–53, 2013.

[30] S. L. Brunton and J. N. Kutz, *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[31]S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PloS one*, vol. 11, no. 2, p. e0150171, 2016.

[32] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[33] A. Surana and A. Banaszuk, "Linear observer synthesis for nonlinear systems using Koopman operator framework," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 716–723, 2016.

[34] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6492–6499.

[35] A. Mauroy, I. Mezić, and Y. Susuki, *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*. Springer Nature, 2020, vol. 484.

[36] I. Abraham, G. De La Torre, and T. D. Murphey, "Model-based control using Koopman operators," *arXiv preprint arXiv:1709.01568*, 2017.

[37] M. Rahmani and S. Redkar, "Data-driven Koopman fractional order PID control of a MEMS gyroscope using bat algorithm," *Neural Computing and Applications*, pp. 1–10, 2023.

[38] X. Zhang, W. Pan, R. Scattolini, S. Yu, and X. Xu, "Robust tube-based model predictive control with Koopman operators," *Automatica*, vol. 137, p. 110114, 2022.

[39] G. Mamakoukas, S. Di Cairano, and A. P. Vinod, "Robust model predictive control with data-driven Koopman operators," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3885–3892.

[40] A. Narasingam and J. S.-I. Kwon, "Koopman Lyapunov-based model predictive control of nonlinear chemical process systems," *AIChE Journal*, vol. 65, no. 11, p. e16743, 2019.

[41] M. Korda, Y. Susuki, and I. Mezić, "Power grid transient stabilization using Koopman model predictive control," *IFAC-PapersOnLine*, vol. 51, no. 28, pp. 297–302, 2018.

[42] S. Hanke, S. Peitz, O. Wallscheid, S. Klus, J. Böcker, and M. Dellnitz, "Koopman operator-based finite-control-set model predictive control for electrical drives," *arXiv preprint arXiv:1804.00854*, 2018.

[43] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-Driven Control of Soft Robots Using Koopman Operator Theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2020.

[44] P. S. Cisneros, A. Datar, P. Göttsch, and H. Werner, "Data-Driven quasi-LPV Model Predictive Control Using Koopman Operator Techniques," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6062–6068, 2020.

[45] H. Arbabi, M. Korda, and I. Mezic, "A data-driven Koopman model predictive control framework for nonlinear flows," *arXiv preprint arXiv:1804.05291*, 2018.

[46] A. Narasingam and J. S.-I. Kwon, "Application of Koopman operator for model-based control of fracture propagation and proppant transport in hydraulic fracturing operation," *Journal of Process Control*, vol. 91, pp. 25–36, 2020.

[47] B. Chen, Z. Huang, R. Zhang, W. Liu, H. Li, J. Wang, Y. Fan, and J. Peng, "Data-driven Koopman model predictive control for optimal operation of high-speed trains," *IEEE Access*, vol. 9, pp. 82 233–82 248, 2021.

[48] X. Wang, Y. Kang, and Y. Cao, "Deep Koopman operator based model predictive control for nonlinear robotics systems," in *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE, 2021, pp. 931–936.

[49] J. Wang, B. Xu, J. Lai, Y. Wang, C. Hu, H. Li, and A. Song, "An Improved Koopman-MPC Framework for Data-Driven Modeling and Control of Soft Actuators," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 616–623, 2022.

[50] H. Wang, W. Liang, B. Liang, H. Ren, Z. Du, and Y. Wu, "Robust position control of a continuum manipulator based on selective approach and Koopman operator," *IEEE Transactions on Industrial Electronics*, 2023.

[51] M. Soleimani, F. N. Irani, M. Yadegar, and M. Davoodi, "Multi-objective optimization of building HVAC operation: Advanced strategy using Koopman predictive control and deep learning," *Building and Environment*, vol. 248, p. 111073, 2024.

[52] K. Reif, "Brakes, brake control and driver assistance systems," *Weisbaden, Germany, Springer Vieweg*, 2014.

[53]P. Lugner *et al.*, *Vehicle dynamics of modern passenger cars*. Springer, 2019.

[54] M. Vignati, E. Sabbioni, and D. Tarsitano, "Torque vectoring control for IWM vehicles," *International Journal of Vehicle Performance*, vol. 2, no. 3, pp. 302–324, 2016.

[55] G. Park, K. Han, K. Nam, H. Kim, and S. B. Choi, "Torque Vectoring Algorithm of Electronic-Four-Wheel Drive Vehicles for Enhancement of Cornering Performance," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3668–3679, 2020.

[56] Q. Wang, Y. Zhuang, J. Wei, and K. Guo, "A driver model–based direct yaw moment controller for in-wheel motor electric vehicles," *Advances in Mechanical Engineering*, vol. 11, no. 9, p. 1687814019877319, 2019.

[57] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[58] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[59]J. P. Boyd, *Chebyshev and Fourier spectral methods*. Courier Corporation, 2013.

[60]L. N. Trefethen, *Spectral methods in MATLAB*. SIAM, 2000.

[61] H. Wendland, "Meshless Galerkin methods using radial basis functions," *Mathematics of computation*, vol. 68, no. 228, pp. 1521–1531, 1999.

[62] G. Karniadakis and S. J. Sherwin, *Spectral/Hp element methods for computational fluid dynamics*. Oxford University Press, USA, 2005.

[63] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, p. 4950, 2018.

[64] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4832–4839.

[65] Y. Han, W. Hao, and U. Vaidya, "Deep learning of Koopman representation for control," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 1890–1895.

[66] Y. Xiao, "DDK: A deep Koopman approach for dynamics modeling and trajectory tracking of autonomous vehicles," *arXiv preprint arXiv:2110.14700*, 2021.

[67] H. Shi and M. Q.-H. Meng, "Deep Koopman operator with control for nonlinear systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7700–7707, 2022.

[68] Y. Xiao, X. Zhang, X. Xu, X. Liu, and J. Liu, "Deep neural networks with Koopman operators for modeling and control of autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 135–146, 2022.

[69] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis, "A kernel-based approach to data-driven Koopman spectral analysis," *arXiv preprint arXiv:1411.2260*, 2014.

[70] M. Švec, Š. Ileš, and J. Matuško, "Predictive Direct Yaw Moment Control Based on the Koopman Operator," *IEEE Transactions on Control Systems Technology*, 2023.

[71] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated annealing: From basics to applications," *Handbook of metaheuristics*, pp. 1–35, 2019.

[72] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of metaheuristics*. Springer, 2003, pp. 320–353.

[73] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[74] N. Hansen, "The CMA evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[75] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[76] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems," in *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 533–541.

[77] Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Nomura, and M. Onishi, "Multiobjective tree-structured Parzen estimator," *Journal of Artificial Intelligence Research*, vol. 73, pp. 1209–1250, 2022.

[78] A. Mauroy, I. Mezić, and J. Moehlis, "Isostables, isochrons, and Koopman spectrum for the action–angle representation of stable fixed point dynamics," *Physica D: Nonlinear Phenomena*, vol. 261, pp. 19–30, 2013.

[79] M. Korda and I. Mezić, "Optimal construction of Koopman eigenfunctions for prediction and control," *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5114–5129, 2020.

[80] J. Ng and H. H. Asada, "Data-Driven Encoding: A New Numerical Method for Computation of the Koopman Operator," *IEEE Robotics and Automation Letters*, 2023.

[81] P. Bevanda, M. Beier, S. Kerz, A. Lederer, S. Sosnowski, and S. Hirche, "Diffeomorphically learning stable Koopman operators," *IEEE Control Systems Letters*, vol. 6, pp. 3427–3432, 2022.

[82] V. Cibulka, M. Korda, and T. Haniš, "Dictionary-free Koopman model predictive control with nonlinear input transformation," *arXiv preprint arXiv:2212.13828*, 2022.

[83] V. Cibulka, T. Haniš, and M. Hromčík, "Data-driven identification of vehicle dynamics using Koopman operator," in *2019 22nd International Conference on Process Control (PC19)*. IEEE, 2019, pp. 167–172.

[84] M. Švec, Š. Ileš, and J. Matuško, "Model predictive control of vehicle dynamics based on the Koopman operator with extended dynamic mode decomposition," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1. IEEE, 2021, pp. 68–73.

[85] Y.-x. Yuan, "Recent advances in trust region algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 249–281, 2015.

[86] J. Löfberg, "YALMIP : A Toolbox for Modeling and Optimization in MATLAB," in *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[87] F. Bruzelius, "A theoretical justification of the sine with dwell manoeuvre," *Vehicle System Dynamics*, vol. 53, no. 4, pp. 493–505, 2015.

[88] M. Švec, Š. Ileš, and J. Matuško, "Predictive approach to torque vectoring based on the Koopman operator," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1341–1346.

[89] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2980–2985.

[90] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. US Government printing office, 1970, vol. 55.

[91] B. A. H. Vicente, S. S. James, and S. R. Anderson, "Linear System Identification Versus Physical Modeling of Lateral-Longitudinal Vehicle Dynamics," *IEEE Transactions on Control Systems Technology*, 2020.

[92] S. James and S. R. Anderson, "Linear system identification of longitudinal vehicle dynamics versus nonlinear physical modelling," in *2018 UKACC 12th International Conference on Control (CONTROL)*. IEEE, 2018, pp. 146–151.

[93] L. Zhang, H. Ding, Y. Huang, H. Chen, K. Guo, and Q. Li, "An analytical approach to improve vehicle maneuverability via torque vectoring control: theoretical study and experimental validation," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4514–4526, 2019.

[94] L. De Novellis, A. Sorniotti, P. Gruber, and A. Pennycott, "Comparison of feedback control techniques for torque-vectoring control of fully electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 8, pp. 3612–3623, 2014.

[95] Y. Chen and J. Wang, "Adaptive energy-efficient control allocation for planar motion control of over-actuated electric ground vehicles," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1362–1373, 2013.

[96] Q. Lu, A. Sorniotti, P. Gruber, J. Theunissen, and J. De Smet, "H_∞ loop shaping for the torque-vectoring control of electric vehicles: Theoretical design and experimental assessment," *Mechatronics*, vol. 35, pp. 32–43, 2016.

[97] A. Parra, A. Zubizarreta, J. Pérez, and M. Dendaluce, "Intelligent torque vectoring approach for electric vehicles with per-wheel motors," *Complexity*, vol. 2018, 2018.

[98] A. Parra, D. Tavernini, P. Gruber, A. Sorniotti, A. Zubizarreta, and J. Pérez, "On nonlinear model predictive control for energy-efficient torque-vectoring," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 173–188, 2020.

[99] E. Siampis, E. Velenis, S. Gariuolo, and S. Longo, "A real-time nonlinear model predictive control strategy for stabilization of an electric vehicle at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 1982–1994, 2017.

[100] M. Dalboni, D. Tavernini, U. Montanaro, A. Soldati, C. Concari, M. Dhaens, and A. Sorniotti, "Nonlinear model predictive control for integrated energy-efficient torque-vectoring and anti-roll moment distribution," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 3, pp. 1212–1224, 2021.

[101] T. Fu, H. Zhou, and Z. Liu, "NMPC-based path tracking control strategy for autonomous vehicles with stable limit handling," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 12, pp. 12 499–12 510, 2022.

[102] G. Palmieri, O. Barbarisi, S. Scala, and L. Glielmo, "A preliminary study to integrate LTV-MPC lateral vehicle dynamics control with a slip control," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 4625–4630.

[103] H. Kanchwala and C. Bordons, "Improving handling performance of an electric vehicle using model predictive control," SAE Technical Paper, Tech. Rep., 2015.

[104] V. Cibulka, T. Haniš, M. Korda, and M. Hromčík, "Model Predictive Control of a Vehicle using Koopman Operator," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 4228–4233, 2020.

[105] J. S. Kim, Y. S. Quan, and C. C. Chung, "Data-Driven Modeling and Control for Lane Keeping System of Automated Driving Vehicles: Koopman Operator Approach," in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2022, pp. 1049–1055.

[106] ——, "Koopman Operator-based Model Identification and Control for Automated Driving Vehicle," *International Journal of Control, Automation and Systems*, vol. 21, no. 8, pp. 2431–2443, 2023.

[107] J. Buzhardt and P. Tallapragada, "A Koopman operator approach for the vertical stabilization of an off-road vehicle," *IFAC-PapersOnLine*, vol. 55, no. 37, pp. 675–680, 2022.

[108] S. Yu, C. Shen, and T. Ersal, "Autonomous driving using linear model predictive control with a Koopman operator based bilinear vehicle model," *IFAC-PapersOnLine*, vol. 55, no. 24, pp. 254–259, 2022.

[109] S. Yu, E. Sheng, Y. Zhang, Y. Li, H. Chen, and Y. Hao, "Efficient nonlinear model predictive control of automated vehicles," *Mathematics*, vol. 10, no. 21, p. 4163, 2022.

[110] S. Gupta, D. Shen, D. Karbowski, and A. Rousseau, "Koopman model predictive control for eco-driving of automated vehicles," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 2443–2448.

[111] D. Shen, J. Han, D. Karbowski, and A. Rousseau, "Data-driven design of model predictive control for powertrain-aware eco-driving considering nonlinearities using Koopman analysis," *IFAC-PapersOnLine*, vol. 55, no. 24, pp. 117–122, 2022.

[112] A. Sassella, V. Breschi, M. Korda, and S. Formentin, "Model-based and Koopman-based predictive control: a braking control systems comparison," *IFAC-PapersOnLine*, vol. 56, no. 3, pp. 325–330, 2023.

[113] W. Guo, S. Zhao, H. Cao, B. Yi, and X. Song, "Koopman operator-based driver-vehicle dynamic model for shared control systems," *Applied Mathematical Modelling*, vol. 114, pp. 423–446, 2023.

[114] H. Chen, X. He, S. Cheng, and C. Lv, "Deep Koopman Operator-Informed Safety Command Governor for Autonomous Vehicles," *IEEE/ASME Transactions on Mechatronics*, 2024.

[115] M. Švec, J. K. Hromatko, and Š. Ileš, "Testing Nonlinear Predictive Torque Vectoring on a Scaled Car Driving on a Roadway Simulator," in *2023 31st Mediterranean Conference on Control and Automation (MED)*. IEEE, 2023, pp. 920–925.

[116] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," *Nonlinear model predictive control: towards new challenging applications*, pp. 391–417, 2009.

[117] Embotech AG, "FORCESPRO," 2014–2023. [Online]. Available:https://forces.embotech.com

[118] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, pp. 1–17, 2017.

[119] A. Mehra, W.-L. Ma, F. Berg, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 1411–1418.

[120] Z. Xu, M. Wang, F. Zhang, S. Jin, J. Zhang, and X. Zhao, "PaTAVTT: A hardware-in-the-loop scaled platform for testing autonomous vehicle trajectory tracking," *Journal of Advanced Transportation*, vol. 2017, 2017.

[121] K. Berntorp, T. Hoang, R. Quirynen, and S. Di Cairano, "Control architecture design for autonomous vehicles," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2018, pp. 404–411.

[122] A. Verma, S. Bagkar, N. V. S. Allam, A. Raman, M. Schmid, and V. N. Krovi, "Implementation and Validation of Behavior Cloning Using Scaled Vehicles," SAE Technical Paper, Tech. Rep., 2021.

[123] S. Brennan and A. Alleyne, "The Illinois Roadway Simulator: A mechatronic testbed for vehicle dynamics and control," *IEEE/ASME Transactions on Mechatronics*, vol. 5, no. 4, pp. 349–359, 2000.

[124] ——, "Using a scale testbed: Controller design and evaluation," *IEEE Control Systems Magazine*, vol. 21, no. 3, pp. 15–26, 2001.

[125] ——, "Dimensionless robust control with application to vehicles," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 624–630, 2005.

[126] S. Lapapong, V. Gupta, E. Callejas, and S. Brennan, "Fidelity of using scaled vehicles for chassis dynamic studies," *Vehicle System Dynamics*, vol. 47, no. 11, pp. 1401–1437, 2009.

[127] E. Buckingham, "On physically similar systems; illustrations of the use of dimensional equations," *Physical review*, vol. 4, no. 4, p. 345, 1914.

[128] P. Makarun, G. Josipović, M. Švec, and Š. Ileš, "Testing predictive vehicle dynamics control algorithms using a scaled remote controlled car and a roadway simulator," in *2021 International Conference on Electrical Drives & Power Electronics (EDPE)*. IEEE, 2021, pp. 177–182.

[129] I. Šolc, P. Makarun, J. K. Hromatko, and Š. Ileš, "Testing direct yaw moment control using a scaled car and a roadway simulator," in *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 2022, pp. 800–805.

[130] Š. Ileš, M. Švec, P. Makarun, and J. K. Hromatko, "Predictive direct yaw moment control with active steering based on polytopic linear parameter-varying model," in *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, vol. 1. IEEE, 2022, pp. 920–925.

[131] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-Based Approach to Active Steering for Autonomous Vehicle Systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, pp. 265–291, 2005.

[132] T. X. Nghiem, J. Drgoňa, C. Jones, Z. Nagy, R. Schwan, B. Dey, A. Chakrabarty, S. Di Cairano, J. A. Paulson, A. Carron *et al.*, "Physics-Informed Machine Learning for Modeling and Control of Dynamical Systems," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 3735–3750.

[133] W. Wei, G. Yin, and T. He, "Physics-Informed Data-Based LPV Modeling and Validations of Lateral Vehicle Dynamics," *IEEE Transactions on Intelligent Vehicles*, 2023.

[134] J. Rice, W. Xu, and A. August, "Analyzing Koopman approaches to physics-informed machine learning for long-term sea-surface temperature forecasting," *arXiv preprint arXiv:2010.00399*, 2020.

[135] P. J. Baddoo, B. Herrmann, B. J. McKeon, J. Nathan Kutz, and S. L. Brunton, "Physics-informed dynamic mode decomposition," *Proceedings of the Royal Society A*, vol. 479, no. 2271, p. 20220576, 2023.

LIST OF FIGURES

LIST OF TABLES

CURICULUM VITAE

Marko Švec was born in Zagreb, Croatia, in 1994. He completed the natural sciences and mathematics program at the Lucijan Vranjanin Gymnasium in Zagreb in 2013. He earned the academic title of master of science in electrical engineering and information technology (summa cum laude) in 2018 after completing his graduate studies at the University of Zagreb, Faculty of Electrical Engineering and Computing (FER). He completed the first semester of his graduate program at Chalmers University of Technology in Gothenburg, Sweden, as part of the Erasmus+ student exchange program. Between the first and second year of his graduate studies, he attended a summer internship at GlobalLogic in Zagreb.

During his studies, he received the University of Zagreb scholarship multiple times, and for outstanding achievement in his second year of undergraduate studies, he was given the "Josip Lončar" award in 2015.

After his graduate studies, he was employed as a researcher in the Laboratory for Mechatronic Systems at the Department of Electric Machines, Drives and Automation at FER. During his employment at FER, he participated in scientific projects Advanced Methods and Technologies in Data Science and Cooperative Systems (DATACROSS), Dynamic Predictive Health Protection of an Electric Vehicle Battery (EVBattPredtect), and Predictive Vehicle Dynamics Control (PVDC). Besides these projects, he also worked on the development of a torque vectoring system for Rimac Technology. Since August 2023, he has been employed as a research and development engineer at Visage Technologies, where he works on visual safety systems in the automotive industry.

His main areas of interest are model predictive control and advanced methods for nonlinear systems identification. He has published his research results in one journal and eleven conference papers, and his work on the EVBattPredtect project resulted in one patent. His other interests include mechatronics, intelligent systems with a focus on their application in the automotive industry, and mathematical optimization.

PUBLICATIONS

PATENTS

1. J. Matuško, Š. Ileš, M. Švec, A. Krishnakumar. System for electric vehicle dynamics control which considers calculated restrictions for protection of vehicle battery integrity. Patent No. P20220282, 2024.

JOURNAL PUBLICATIONS

1. M. Švec, Š. Ileš and J. Matuško, "Predictive Direct Yaw Moment Control Based on the Koopman Operator," in *IEEE Transactions on Control Systems Technology*, 2023, vol. 31, no. 6, pp. 2912-2919.

CONFERENCE PUBLICATIONS

1. J. K. Hromatko, M. Švec, and Š. Ileš, "Autonomous Path Following Using Data-Driven Predictive Control," in *2023 27th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2023, pp. 368–373.
2. M. Švec, B. V. Belina, Š. Ileš, and J. Matuško, "Nonlinear Predictive Torque Vectoring with Brake Blending for Electric Road Vehicles," in *2023 IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE, 2023.
3. M. Švec, J. K. Hromatko, and Š. Ileš, "Testing Nonlinear Predictive Torque Vectoring on a Scaled Car Driving on a Roadway Simulator," in *2023 31st Mediterranean Conference on Control and Automation (MED)*. IEEE, 2023, pp. 920–925.
4. Š. Ileš, M. Švec, P. Makarun, and J. K. Hromatko, "Stabilizing direct yaw moment control based on a flexible set-membership constraint," in *2022 30th Mediterranean Conference on Control and Automation (MED)*. IEEE, 2022, pp. 289–294.
5. Š. Ileš, M. Švec, P. Makarun, and J. K. Hromatko, "Predictive direct yaw moment control with active steering based on polytopic linear parameter-varying model," in *2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*, vol. 1. IEEE, 2022, pp. 920–925.
6. P. Makarun, G. Josipović, M. Švec, and Š. Ileš, "Testing predictive vehicle dynamics control algorithms using a scaled remote controlled car and a roadway simulator," in *2021*

International Conference on Electrical Drives & Power Electronics (EDPE). IEEE, 2021, pp. 177–182.

7. M. Švec, Š. Ileš, and J. Matuško, "Predictive approach to torque vectoring based on the Koopman operator," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1341–1346.

8. M. Švec, Š. Ileš, and J. Matuško, "Model predictive control of vehicle dynamics based on the Koopman operator with extended dynamic mode decomposition," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1. IEEE, 2021, pp. 68–73.

9. M. Švec, Š. Ileš, and J. Matuško, "Sliding Mode Control of Custom Built Rotary Inverted Pendulum," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 2020, pp. 943–947.

10. B. Spahija, M. Švec, J. Matuško, and Š. Ileš, "Successive Linearization Based Predictive Vehicle Torque Vectoring," in *2019 International Conference on Electrical Drives & Power Electronics (EDPE)*. IEEE, 2019, pp. 267–271.

11. M. Švec, K. Hrvatinić, Š. Ileš, and J.Matuško, "Predictive Torque Vectoring Vehicle Control Based on a Linear Time Varying Model," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2019, pp. 960–965.

ŽIVOTOPIS

Marko Švec rođen je u Zagrebu, Hrvatska, 1994. godine. Prirodoslovno-matematički smjer Gimnazije Lucijana Vranjanina završio je 2013. godine u Zagrebu. Akademski naziv magistar inženjer elektrotehnike i informacijske tehnologije (summa cum laude) stekao je 2018. godine završivši diplomski studij na Sveučilištu u Zagrebu, Fakultetu elektrotehnike i računarstva (FER). Prvi semestar diplomskog studija završio je na Chalmers University of Technology u Göteborgu, Švedska, u sklopu Erasmus+ studentske razmjene. Između prve i druge godine diplomskog studija pohađao je ljetnu praksu u kompaniji GlobalLogic u Zagrebu.

Tijekom studija više je puta nagrađen stipendijom Sveučilišta u Zagrebu, a za izvrstan uspjeh na drugoj godini preddiplomskog studija uručeno mu je priznanje "Josip Lončar" (2015.).

Nakon diplomskog studija zaposlen je kao istraživač u Laboratoriju za mehatroničke sustave na Zavodu za elektrostrojarstvo i automatizaciju na FER-u. Tijekom zaposlenja na FER-u sudjeluje na znanstvenim projektima Napredne metode i tehnologije u znanosti o podatcima i kooperativnim sustavima (DATACROSS), Dinamička prediktivna zaštita integriteta baterije električnog vozila (EVBattPredtect) i Prediktivno upravljanje dinamikom vozila (PVDC). Osim navedenih projekata, sudjelovao je i na razvoju sustava upravljanja raspodjelom pogonskog momenta za kompaniju Rimac Technology. Od kolovoza 2023. godine zaposlen je kao inženjer istraživanja i razvoja u kompaniji Visage Technologies gdje se bavi vizualnim sigurnosnim sustavima u automobilskoj industriji.

Njegova glavna područja interesa su modelsko prediktivno upravljanje i napredne metode identifikacije nelinearnih dinamičkih sustava. Rezultate svojih istraživanja objavio je u jednom časopisnom i jedanaest konferencijskih znanstvenih radova, a njegov rad na projektu EVBattPredtect rezultirao je jednim patentom. Ostali mu interesi uključuju mehatroniku, inteligentne sustave s naglaskom na njihovu primjenu u automobilskoj industriji i matematičku optimizaciju.