Master's Thesis

CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

F2

Faculty of Mechanical Engineering
Department of Mechanics, Biomechanics and Mechatronics

Torque Vectoring Predictive Control for Electric Vehicles

Bc. Jiří Tůma

Supervisor: Ing. Martin Nečas, MSc., Ph.D.
Field of study: Mechanical Engineering
Subfield: Mechatronics
May 2018

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tůma** Jméno: **Jiří** Osobní číslo: **419958**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Strojní inženýrství**

Studijní obor: **Mechatronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Prediktivní řízení točivých momentů pro elektrická vozidla

Název diplomové práce anglicky:

Torque Vectoring Predictive Control for Electric Vehicles

Pokyny pro vypracování:

1. Seznamte se s problematikou planární dynamiky vozidla, vektorování točivých momentů a prediktivního řízení
2. Sestavte simulační model planární dynamiky vozidla a model upravte pro syntézu řízení
3. Sestavte algoritmus prediktivního řízení pro upravený model
4. Proveďte simulační ověření navrženého algoritmu
5. Kriticky zhodnoťte dosažené výsledky

Seznam doporučené literatury:

- Reza N. Jazar. Vehicle Dynamics: Theory and Applications. Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA, 2008.
- F. Vlk. Dynamika motorových vozidel. 2. vydání, Brno, 2003.
- E. F. Camacho, C. Bordons. Model Predictive control. 2nd edition, Springer-Verlag London 2007.
- D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized Predictive Control. Part I. The Basic Algorithm. Automatica, 23(2):137-148, 1987.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Martin Nečas, MSc., Ph.D., odbor mechaniky a mechatroniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **24.04.2018** Termín odevzdání diplomové práce: **17.08.2018**

Platnost zadání diplomové práce:

[illegible]	[illegible]	[illegible]
Ing. Martin Nečas, MSc., Ph.D.	prof. Ing. Milan Růžička, CSc.	prof. Ing. Michael Valášek, DrSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

3.5.2018 — Datum převzetí zadání

[illegible] — Podpis studenta

Acknowledgements

On the occasion of presenting this thesis, I would like to express my acknowledgement and gratitude to my supervisor, Ing. Martin Nečas, MSc., Ph.D., for the long-term support, wonderful time spent together on solving problems of all kinds and leading of this thesis.
Further, I appreciate the support from the Ricardo Technical Centre in Prague, namely from my colleagues from the Control & Electronics department, and I would like to express my thanks to all of them in this form.
Finally, the honorable mention goes to my family, namely to my mother and brother, not only for the support during the master programme period, but for the lifetime guidance, love, patience, faith, belief and endless standby. Whatever I do, I do it for them!

Declaration

In Prague, 25. May 2018

Bibliographic citation:
TŮMA, Jiří. *Torque Vectoring Predictive Control for Electric Vehicles.* Master's Thesis. Prague: Czech Technical University in Prague, Faculty of Mechanical Engineering, 2018.

Abstract

The electric mobility and digitalisation have an immense potential and will write the automotive future. More and more sophisticated algorithms considering and handling safety, efficiency, sustainability and increasing transport demands will be implemented.

The torque vectoring algorithm designed especially for electric vehicles with independently controlled all-wheel drive is presented within this thesis while considering a simplified planar vehicle dynamic model and utilizing model predictive control strategies.

The thesis introduces construction of the dynamic model, its appropriate trimming for control synthesis and the control algorithm along with implementation in the environment Simulink®. The performed simulations resulted successfully in weighted optimal distribution of the individual wheel torques being applied to the vehicle electric motors.

Keywords: Torque Vectoring, Predictive Control, Vehicle dynamics, Electromobility, Simulation

Supervisor: Ing. Martin Nečas, MSc., Ph.D.
Department of Mechanics, Biomechanics and Mechatronics,
Technická 4,
160 00 Prague 6

Abstrakt

Elektromobilita a digitalizace mají obrovský potenciál a budou psát automobilovou budoucnost. Bude implementováno stále více sofistikovaných algoritmů zohledňujících bezpečnost, efektivitu, udržitelnost a zvyšující se požadavky dopravy.

V této práci je prezentován algoritmus vektorování točivých momentů navržený zvláště pro elektrická vozidla s nezávisle říditelným pohonem všech kol, vycházející ze zjednodušeného dynamického modelu vozidla a využívající strategie prediktivního řízení.

Práce představuje sestavení dynamického modelu, jeho vhodnou úpravu pro syntézu řízení a řídicí algoritmus spolu s vlastní realizací v prostředí Simulink®. Provedené simulace byly úspěšně zakončeny výsledným váhově optimálním rozložením jednotlivých hnacích točivých momentů.

Klíčová slova: vektorování točivých momentů, prediktivní řízení, dynamika vozidel, elektromobilita, simulace

Contents

Figures

Tables

Chapter 1
Introduction

1.1 Motivation

In recent years, there has been a larger impact of new technologies coming to our everyday lives than ever before. Major contribution surely belongs to the development of faster and more efficient data processing units and furthermore, the code complexity in all computers increases rapidly. We tend to replace more and more routine tasks in production and services, to better optimize time and resources and the world just gets quicker and more automated. It is all down to the sophisticated algorithms which operate the machines in order to increase safety, improve reliability, accelerate productivity and enhance functionality.

This transformation has inevitably affected the car industry as well. Two of the greatest challenges which drive the automotive society today are electromobility and use of artificial intelligence for autonomous driving. This certainly requires more complex and reliable embedded systems and more advanced algorithms, solutions and strategies.

1.2 Driver assistance systems

Lots of assistance systems have been developed over the years helping the driver to avoid accidents and to prevent potential vehicle uncontrollability in poor driving conditions. Nowadays, a completely new era of advanced driver assistance systems (ADAS) is coming up, relying upon very high-tech components and sensors, such as lidar (Light Detection and Ranging), utilizing laser pulses to detect objects in its range, or radar, cameras or ultrasonic sensors. Some assistance systems based upon these technologies are listed below:

- Adaptive cruise control (ACC)
- Lane change assistance
- Traffic sign recognition
- Automatic parking
- ...

Nevertheless, those control systems focus on data acquisition from the vehicle surroundings and upon object detection and recognition, but they themselves do not improve vehicle stability or maneuverability in any way.

The problems regarding the vehicle poor stability in certain driving or weather conditions were encountered much earlier, therefore, the following systems have been developed since 1970s and 1980s or even beforehand.

Anti-lock braking system (ABS) is a safety control system which prevents the wheels from locking up and skidding on the surface and ensures continuous traction while braking. In principle, the system measures speed of each wheel and compares it with the vehicle's speed. If the wheel speed is considerably lower than it should (often due to loss of traction on a slippery road), the system releases a hydraulic valve, which reduces hydraulic pressure and thus braking force to the corresponding wheel. ABS is considered as an essential system, according to EU law it is a compulsory element of all cars built in Europe since 2004.

Electronic stability control (ESC) is a safety control system which monitors loss of traction and steering while driving and it brakes individual wheels in order to steer the vehicle in driver's intentional direction. In principle, applying brakes independently while driving creates a yaw moment which helps (when correctly estimated) the vehicle to stabilize. It might be utilized in certain oversteer or understeer situations, etc., or it prevents the car from skidding on a wet or icy road surface. The vehicles in Europe have been obligatory fitted with ESC since 2014.

Since their first release, these systems have been further improved and extended, and other new related systems have been evolved. All the above mentioned solutions have been integrated into the ECUs (Electronic control units) with respect to quality standards and functional safety described for instance in well-known ISO 26262.

1.3 Torque vectoring system

The purpose of this thesis is to aim for the above indicated direction and to present, simulate and to develop a rather new torque vectoring system technology, which is ideally based upon a proper torque splitting and torque differentiating among the vehicle wheels while cornering in order to improve its stability, handling, performance and maneuverability and thus to reduce the driver's effort and increase passenger safety.

Such systems are not entirely new, they have been used mostly in premium car segment and have been working largely on a mechanical basis with electronic control as an additional feature of the differential. The torque vectoring system could be further classified according to the drive axles. Driving only front or rear axle and varying torque between two wheels makes the system less complex. However, torque vectoring is mostly implemented on cars with all-wheel drive, as for example a quattro® system from Audi AG.

Ricardo plc, a British based, global engineering and technical consultancy company, has been offering one of the most compact torque vectoring solutions available and demonstrated with it the improvement in handling and stability both on and off-road. This master's thesis has been created in an association with Ricardo Technical Centre in Prague.

1.4 Electric vehicles

Regarding the climate change and the environmental issues caused by an increasing level of CO_2 emissions, it is our mission in a few decades to transform the energy exploitation into something sustainable. The legislation process in Europe massively supports pure electric and hybrid vehicles, all major car manufacturers present every year a new electric fleet and invest immensely into the development. We can certainly expect this trend to be further ongoing. But we cannot just skip over the so far very advanced internal combustion engines (ICEs), because a ton of predictable and possibly even unexpected problems will emerge.

Supersport cars are usually developed by companies demonstrating as much of an ultra-advanced and futuristic technology as they can. Mainly because the interested customers pay the huge price regardless of its height. For instance, the Rimac Concept One or Concept Two developed and manufactured in Croatia by Rimac Automobili or the Mercedes-Benz SLS AMG Electric Drive coming from Stuttgart, they have in common a four-wheel electric drive, so the wheels are completely independent and utilize the torque vectoring solution. As the development costs of this system decline, it is still going to be offered in more affordable car segments spreading throughout the automotive industry.

Electric vehicles have overall lots of advantages, they operate very quietly, have huge performance figures in terms of acceleration, which is instant comparing it to the vehicles with ICEs. They also have zero CO_2 emissions, treat energy flows better, they can recuperate while braking and their powertrain is mechanically less complex, which reduces its production costs. On the downside, there is a critical point of storing the electrical energy. The so far developed and produced electric vehicles rely mostly on using Li-Ion battery packs. And their production is such energy-intensive, that according to Fraunhofer Institute for Building Physics the amount of overall energy required to produce the electric car is twice as much as to produce the conventional one. And if we consider electric vehicles being recharged with electricity coming from coal-fired power stations, there is certainly no point of proclaiming them as eco-friendly, the emissions may decrease merely locally, but it generally does not solve the issue. Taking into account a problematic battery packs disposal and mining and quarrying rare earth minerals, it might overall seem that we are still not heading in the right direction.

Regardless of how exactly the world will look like in 20-year-horizon, it is clear that this "eco-transformation" will affect us more than anybody can presume. At this point I would like to challenge everybody to take true responsibility in every way for contributing to our shared living environment and appeal on the importance of being and behaving smarter, which might help, but not paying attention and not taking care will make everything worse definitely.

Part I

Modelling vehicle dynamics

Chapter 2
Planar Vehicle Dynamics

2.1 Body frame and coordinates

In order to derive vehicle model dynamics equations, we must postulate several simplifications which we mention throughout the text. A vehicle has generally 6 degrees of freedom (dof), for which we define 3 translational (longitudinal x, lateral y, normal z) and 3 rotational movements (roll φ, pitch ϕ, yaw ψ). For our case, we consider a planar vehicle body frame with 3 dof in coordinates x, y and ψ (with respect to the global coordinate system) as expressed in fig. 2.1 which simplifies force vectors acting on the body and velocity vectors describing its motion accordingly.

$$ {}^{\boldsymbol{B}}\boldsymbol{F} = \begin{bmatrix} F_x \\ F_y \\ 0 \end{bmatrix} \tag{2.1} $$

$$ {}^{\boldsymbol{B}}\boldsymbol{M} = \begin{bmatrix} 0 \\ 0 \\ M_z \end{bmatrix} \tag{2.2} $$

$$ {}^{\boldsymbol{B}}\boldsymbol{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} \tag{2.3} $$

$$ {}^{\boldsymbol{B}}\boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.4} $$

2.2 Newton-Euler equations

We assume the body frame to be the principal coordinate frame to obtain the principal moment of inertia matrix, which further reduces complexity of the motion dynamics.

$$ {}^{\boldsymbol{B}}\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.5} $$

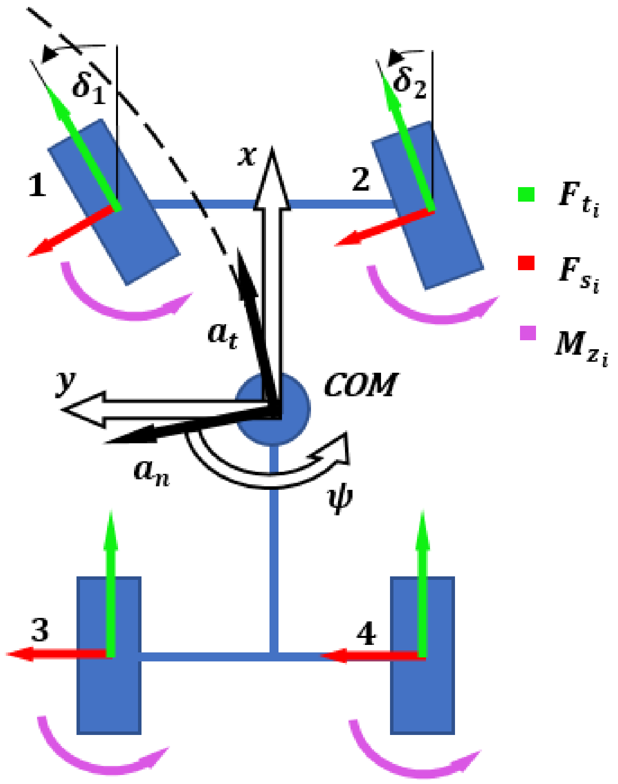

Figure 2.1: Vehicle model dynamics

The rigid body dynamics concentrated to the *COM* - center of mass is according to standard convention without loss of generality described by these (Newton-Euler) equations:

$$ {}^{B}\boldsymbol{F} = m {}^{B}\dot{\boldsymbol{v}} = m({}^{B}\boldsymbol{a_t} + {}^{B}\boldsymbol{a_n}) = m({}^{B}\dot{v}\hat{\boldsymbol{u}}_{\boldsymbol{t}} + {}^{B}\boldsymbol{\omega} \times {}^{B}\boldsymbol{v}) \tag{2.6}$$

$$ {}^{B}\boldsymbol{M} = {}^{B}\boldsymbol{I}^{B}\dot{\boldsymbol{\omega}} + {}^{B}\boldsymbol{\omega} \times {}^{B}\boldsymbol{I}^{B}\boldsymbol{\omega} \tag{2.7}$$

where m is weight of a vehicle, ${}^{B}\boldsymbol{a_t}$ and ${}^{B}\boldsymbol{a_n}$ represent tangential and normal acceleration vectors, respectively, and with $\hat{\boldsymbol{u}}_{\boldsymbol{t}}$ being the unit vector in tangential acceleration direction. And by substituting the above vectors and matrices we obtain:

$$ {}^{B}\boldsymbol{F} = m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ 0\end{bmatrix} + m\begin{bmatrix}0\\ 0\\ \dot{\psi}\end{bmatrix} \times \begin{bmatrix}\dot{x}\\ \dot{y}\\ 0\end{bmatrix} = \begin{bmatrix}m\ddot{x} - m\dot{\psi}\dot{y}\\ m\ddot{y} + m\dot{\psi}\dot{x}\\ 0\end{bmatrix} \tag{2.8}$$

$$ {}^{B}\boldsymbol{M} = \begin{bmatrix}I_{xx} & 0 & 0\\ 0 & I_{yy} & 0\\ 0 & 0 & I_{zz}\end{bmatrix}\begin{bmatrix}0\\ 0\\ \ddot{\psi}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \dot{\psi}\end{bmatrix} \times \begin{bmatrix}I_{xx} & 0 & 0\\ 0 & I_{yy} & 0\\ 0 & 0 & I_{zz}\end{bmatrix}\begin{bmatrix}0\\ 0\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}0\\ 0\\ I_{zz}\ddot{\psi}\end{bmatrix} \tag{2.9}$$

The entire planar model is then represented by 3 equations:

$$\begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} = \begin{bmatrix} m\ddot{x} - m\dot{\psi}\dot{y} \\ m\ddot{y} + m\dot{\psi}\dot{x} \\ I_{zz}\ddot{\psi} \end{bmatrix} \tag{2.10}$$

These applied forces are distributed among 4 wheels as depicted in fig. 2.1. We define the steering angles δ_1 and δ_2 on the front axle and compute the forces summation accordingly ($\delta_3 = \delta_4 = 0$).

$$F_x = \sum_{i=1}^{4} F_{x_i} = \sum_{i=1}^{4} (F_{t_i} \cos\delta_i - F_{s_i} \sin\delta_i) \tag{2.11}$$

$$F_y = \sum_{i=1}^{4} F_{y_i} = \sum_{i=1}^{4} (F_{t_i} \sin\delta_i + F_{s_i} \cos\delta_i) \tag{2.12}$$

$$\begin{aligned} M_z = \sum_{i=1}^{4} (M_{z_i} + x_i F_{y_i} - y_i F_{x_i}) = \\ = \sum_{i=1}^{4} (M_{z_i} + x_i (F_{t_i} \sin\delta_i + F_{s_i} \cos\delta_i) - y_i (F_{t_i} \cos\delta_i - F_{s_i} \sin\delta_i)) \end{aligned} \tag{2.13}$$

Forces F_{t_i} and F_{s_i} are summations of all forces in the wheel's longitudinal and lateral direction, respectively, x_i, y_i represent wheel's coordinates and M_{z_i} are sums of all yaw moments acting on the individual wheel.

2.3 Forward vehicle dynamics

As we can predict, some of the previously derived tyre forces depend on the ground reaction forces. In order to determine the weight distribution, we come out of the fig. 2.2, considering x-axis symmetry, omitting the steering angle dependence for this instance ($\delta_1 \approx \delta_2 \approx 0$) and lateral y motion. This simplification significantly reduces the complexity of overall kinematics and dynamics. Newton-Euler equations for this instance can be described by:

$$\sum F_x = m\ddot{x} \tag{2.14}$$

$$\sum F_z = 0 \tag{2.15}$$

$$\sum M_y = 0 \tag{2.16}$$

Since we assume equal force distribution between the wheels on the same axle and $x_1 = x_2$, $x_3 = x_4$, we elaborate the expressions further:

$$2F_{x_1} + 2F_{x_3} = m\ddot{x} \tag{2.17}$$

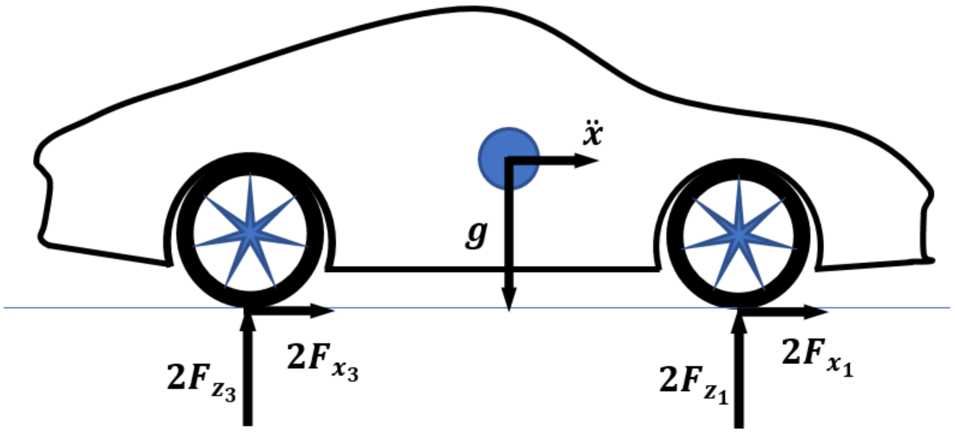

Figure 2.2: Forward vehicle dynamics

$$2F_{z_1} + 2F_{z_3} - mg = 0 \tag{2.18}$$

$$(2F_{x_1} + 2F_{x_3})h + 2F_{z_1}x_1 - 2F_{z_3}|x_3| = 0 \tag{2.19}$$

where h is the center of gravity height. By simplifying x-forces into one unknown, we obtain solutions:

$$F_{z_1} = F_{z_2} = \frac{1}{2}mg\frac{|x_3|}{x_1 + |x_3|} - \frac{1}{2}m\ddot{x}\frac{h}{x_1 + |x_3|} \tag{2.20}$$

$$F_{z_3} = F_{z_4} = \frac{1}{2}mg\frac{x_1}{x_1 + |x_3|} + \frac{1}{2}m\ddot{x}\frac{h}{x_1 + |x_3|} \tag{2.21}$$

The expressions 2.20 and 2.21 contain terms with $\ddot{x}$ dependence, which represent contributions to the reaction forces appearing either by accelerating or decelerating. But as we assume torque optimization rather for lateral vehicle motion, we state $\ddot{x} \approx 0$ and use in all cases following reaction forces.

$$F_{z_1} = F_{z_2} = \frac{1}{2}mg\frac{|x_3|}{x_1 + |x_3|} \tag{2.22}$$

$$F_{z_3} = F_{z_4} = \frac{1}{2}mg\frac{x_1}{x_1 + |x_3|} \tag{2.23}$$

2.4 Vehicle kinematics

The highest impact on maneuverability has the vehicle's steering mechanism. There are several steering types. Most two-axle-vehicles utilize front-wheel-steering (passenger vehicles), but forklifts usually have rear-wheel-steering and there are also some premium-segment-cars which are capable of steering both axles simultaneously. The other division is according to the steered-wheel-angles when considering fig. 2.1. We may deliberately choose $\delta_1 > \delta_2$ and fulfill further the Ackermann condition 2.24 for δ_2 being the outer wheel, or select $\delta_1 = \delta_2$ for parallel steering, or $\delta_1 < \delta_2$ for reverse.

$$\cot\delta_2 - \cot\delta_1 = \frac{y_1 + |y_2|}{x_2 + |x_3|} \tag{2.24}$$

The advantage of Ackermann steering is that the tyres produce no slip at low speeds because of the common turning center. We will derive other expressions utilizing this mechanism, considering lower vehicle speeds in order to minimize the contribution of slip forces onto the final dynamics. Our independent input is a steering wheel angle which we consider as a ”mean” value δ (heading angle measured from the center of gravity):

$$\cot\delta = \frac{\cot\delta_2 + \cot\delta_1}{2} \tag{2.25}$$

Regarding the lateral vehicle velocity, wheel velocities can be generally derived [1] in the body coordinate frame 2.26 and for our case specifically 2.27. Lateral tyre slip forces begin to appear as a result of the lateral tyre slipping on the surface, described by so-called global 2.28 or local 2.29 tyre slip angles, which are according to standard convention defined for each wheel (i = 1 - 4) and generate the nonholonomic constraint to our system.

$$^{\boldsymbol{B}}\boldsymbol{v_i} = {}^{\boldsymbol{B}}\boldsymbol{v} + {}^{\boldsymbol{B}}\boldsymbol{\omega} \times {}^{\boldsymbol{B}}\boldsymbol{r_i} \tag{2.26}$$

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{x} - \dot{\psi} y_i \\ \dot{y} + \dot{\psi} x_i \\ 0 \end{bmatrix} \tag{2.27}$$

$$\beta_i = \arctan\left(\frac{\dot{y}_i}{\dot{x}_i}\right) = \arctan\left(\frac{\dot{y} + \dot{\psi} x_i}{\dot{x} - \dot{\psi} y_i}\right) \tag{2.28}$$

$$\alpha_i = \delta_i - \arctan\left(\frac{\dot{y} + \dot{\psi} x_i}{\dot{x} - \dot{\psi} y_i}\right) \tag{2.29}$$

2.5 Wheel and tyre dynamics

Every wheel has in general case 6 dof as well, we consider the corresponding forces in longitudinal F_{t_i} and lateral F_{s_i} direction and pitch moment M_{y_i} (also called rolling resistance torque). Remaining forces F_{z_i} (reaction force itself), M_{x_i} (roll moment, tilting moment) and M_{z_i} (yaw moment, aligning moment) will be neglected further for the dynamic model.

We derive the motion equation for a wheel regarding the actual inputs of our torque

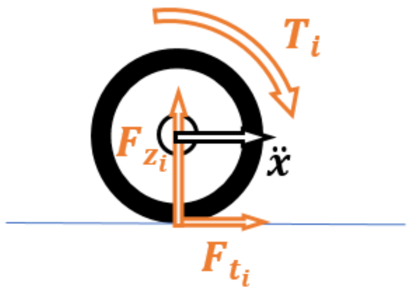

Figure 2.3: Wheel dynamics

vectoring model T_i (depicted in fig. 2.3). Adding up the kinematic condition for no longitudinal tyre slip allowed, we claim 2.31.

$$I_w \ddot{\theta}_{w_i} = T_i - F_{z_i}\xi - F_{t_i} r_w \tag{2.30}$$

$$\ddot{\theta}_{w_i} r_w = \ddot{x}_{w_i} \tag{2.31}$$

$$\mu_r = \frac{\xi}{r_w} \tag{2.32}$$

$$\ddot{x}_{w_i} \approx \ddot{x} \tag{2.33}$$

$$\implies I_w \frac{\ddot{x}}{r_w} = T_i - F_{z_i}\mu_r r_w - F_{t_i} r_w \tag{2.34}$$

where r_w and I_w are the wheel's radius and inertia, $\ddot{\theta}_{w_i}$ and $\ddot{x}_{w_i}$ are wheel's coordinates, μ_r is a dimensionless rolling friction coefficient 2.32 and ξ an alternative coefficient with dimension of length corresponding to an offset caused by contact pressure deformation. We must emphasize at this point that 2.34 could only be derived under the assumption of forward wheel dynamics 2.33, when no lateral vehicle motion is considered and also $\delta_1 \approx \delta_2 \approx 0$, which is obviously in conflict with 2.10 for front (steered) wheels. As previously, we state $\ddot{x} \approx 0$, neglect rolling resistance torque and obtain final expression.

$$F_{t_i} = \frac{T_i}{r_w} \tag{2.35}$$

The remaining unknowns to be determined are the wheel or tyre lateral forces. This itself is a complex topic and is influenced by many parameters. So we simplify our consideration for this part as well, as much as possible. A tyre possesses very anisotropic properties, it contains many different material layers with complex viscoelastic attributes, the local structure varies significantly and the resulting behaviour may be observed as elastoplatic nonlinear deflections.

In reality, tyre modelling appears to be empirically determined and there exist more or less complicated equations describing the tyre-road interaction. One of the

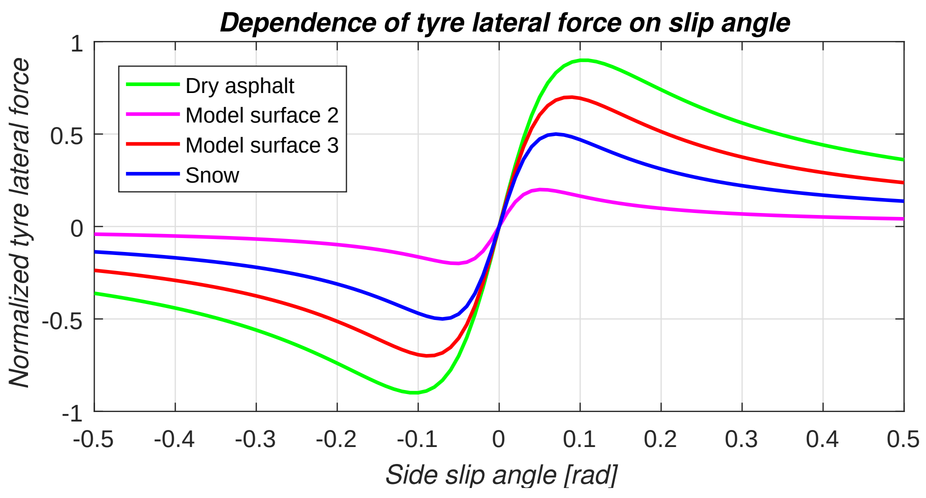

Figure 2.4: Lateral tyre force determination

most popular is certainly the Pacejka's *Magic Formula* 2.36, where the lateral force depend on unique coefficients for each type of a tyre and moreover, in nonlinear fashion. Coefficient B represents stiffness factor, C shape factor, D peak value and E curvature factor. This solution can be nowadays achieved with software modelling aid much faster, for instance with the product Delft-Tyre from company TASS International. Another approach is to utilize any type of estimator to determine the friction or tyre characteristics as it was shown in master's thesis in cooperation with Volvo Group [2].

$$F_{s_i} = D \sin \{C \arctan [B\alpha_i - E(B\alpha_i - \arctan (B\alpha_i))]\} \tag{2.36}$$

In our model we use a very similar dependence, as explained in [4], and described by following equation set

$$F_{s_i} = F_{z_i}\mu(\alpha_i) \tag{2.37}$$

$$\mu(\alpha_i) = \frac{2\alpha_p\mu_p}{\alpha_p^2 + \alpha_i^2}\alpha_i \tag{2.38}$$

where α_p and μ_p are peak values of the curve depicted in fig. 2.4

2.6 Nonlinear dynamic model

We put together the above mentioned principles and dynamic equations. Substituting 2.35 and 2.37 into 2.10, we obtain the following equation set:

$$m\ddot{x} - m\dot{\psi}\dot{y} = \sum_i \left\{ \frac{T_i}{r_w} \cos\delta_i - F_{z_i}\mu(\alpha_i)\sin\delta_i \right\} \tag{2.39}$$

$$m\ddot{y} + m\dot{\psi}\dot{x} = \sum_i \left\{ \frac{T_i}{r_w} \sin\delta_i + F_{z_i}\mu(\alpha_i)\cos\delta_i \right\} \tag{2.40}$$

$$I_{zz}\ddot{\psi} = \sum_i \left\{ x_i \left(\frac{T_i}{r_w} \sin\delta_i + F_{z_i}\mu(\alpha_i)\cos\delta_i \right) - y_i \left(\frac{T_i}{r_w}\cos\delta_i - F_{z_i}\mu(\alpha_i)\sin\delta_i \right) \right\} \tag{2.41}$$

We trim the equations further into a proper form containing merely first order differential equations with definitions as follows:

$$\boldsymbol{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{2.42}$$

$$\boldsymbol{u} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{2.43}$$

$$\dot{z}_1 = \left(\frac{1}{m}\right) \left\{ \sum_i \left[\frac{u_i}{r_w}\cos\delta_i - F_{z_i}\mu(\alpha_i)\sin\delta_i \right] + m z_2 z_3 \right\} \tag{2.44}$$

$$\dot{z}_2 = \left(\frac{1}{m}\right) \left\{ \sum_i \left[\frac{u_i}{r_w}\sin\delta_i + F_{z_i}\mu(\alpha_i)\cos\delta_i \right] - m z_1 z_3 \right\} \tag{2.45}$$

$$\dot{z}_3 = \left(\frac{1}{I_{zz}}\right) \left\{ \sum_i \left[x_i \left(\frac{u_i}{r_w}\sin\delta_i + F_{z_i}\mu(\alpha_i)\cos\delta_i \right) - y_i \left(\frac{u_i}{r_w}\cos\delta_i - F_{z_i}\mu(\alpha_i)\sin\delta_i \right) \right] \right\} \tag{2.46}$$

$$\alpha_i = \delta_i - \arctan\left(\frac{z_2 + z_3 x_i}{z_1 - z_3 y_i} \right) \tag{2.47}$$

where 2.42 defines system state variables and 2.43 input torques and 2.47 tyre local side slip angle nonintegrable constraint, which makes the whole dynamic system nonholonomic.

Chapter 3

Model verification and state space form

3.1 Model simulation

We test and verify the dynamic model described by equations 2.44 - 2.47 in Simulink environment. Firstly, we declare vehicle and tyre parameters, as the tab. 3.1 shows.

	Parameter	Value	Unit
vehicle weight	m	1500	kg
vehicle inertia	I_{zz}	2000	kgm^2
wheel radius	r_w	0.28	m
vehicle wheelbase	wb	3	m
axle track	tr	1.6	m
slip angle coefficient	α_p	$6\pi/180$	rad
surface coefficient	μ_p	0.9	1

Table 3.1: Vehicle test parameters

We create function *ack*, see 3.1, computing individual wheel steering angles explicitly from expressions which can be easily derived according to Ackermann conditions and we feed the output angles into our dynamic model function. The test steering angles are demonstrated in fig. 3.1, where the Ackermann angle distribution on both wheels can be observed.

Listing 3.1: Ackermann steering

```
function [delta1,delta2] = ack(d)

% individual wheel coordinates from center of gravity
y1 = 0.8; y2 = -0.8; x2 = 1.3; x3 = -1.7;

w = (y1 + abs(y2));    % axle track
l = (x2 + abs(x3));    % wheelbase
delta1 = acot(cot(d) - w/(2*l));
delta2 = acot(cot(d) + w/(2*l));
```

Listing 3.2: Dynamic model computation

```
function [z1t, z2t, z3t] = dyn_model(d1,d2,u1,u2,u3,u4,z1,
   z2,z3)

% parameter declaration
m = 1500; g = 9.81; rw = 0.28; x1 = 1.3; x2 = 1.3;
x3 = -1.7; x4 = -1.7; y1 = 0.8; y2 = -0.8; y3 = 0.8;
y4 = -0.8; Izz = 2000; d3 = 0; d4 = 0;

% calculating reaction forces
Fz12 = 0.5*m*g*(abs(x3)/(x1 + abs(x3)));
Fz34 = 0.5*m*g*(x1/(x1 + abs(x3)));

% calculating longitudinal forces
F1t = u1/rw; F2t = u2/rw; F3t = u3/rw; F4t = u4/rw;

% calculating lateral forces
F1s = Fz12*tyre(d1,x1,y1,z1,z2,z3);
F2s = Fz12*tyre(d2,x2,y2,z1,z2,z3);
F3s = Fz34*tyre(d3,x3,y3,z1,z2,z3);
F4s = Fz34*tyre(d4,x4,y4,z1,z2,z3);

% dynamic model
z1t = (1/m)*(F1t*cos(d1) - F1s*sin(d1) + F2t*cos(d2)
    - F2s*sin(d2) + F3t + F4t + m*z2*z3);
z2t = (1/m)*(F1t*sin(d1) + F1s*cos(d1) + F2t*sin(d2)
    + F2s*cos(d2) + F3s + F4s - m*z1*z3);
z3t = (1/Izz)*(x1*(F1t*sin(d1) + F1s*cos(d1))
    - y1*(F1t*cos(d1) - F1s*sin(d1)) + x2*(F2t*sin(d2)
    + F2s*cos(d2)) - y2*(F2t*cos(d2) - F2s*sin(d2))
    + x3*F3s - y3*F3t + x4*F4s - y4*F4t);

% function computing normalized tyre lateral force
function mu = tyre(d,x,y,z1,z2,z3)
alpha_p = 6*pi/180;          % peak alpha value
mu_p = 0.9;                  % peak mu value
alpha = d - atan((z2 + z3*x)/(z1 - z3*y));
mu = ((2*alpha_p*mu_p)/(alpha_p^2 + alpha^2))*alpha;
```

We add a state integration and feedback and validate our model considering four equal torque inputs as depicted in fig. A.1. Since we consider all states expressed in a body frame, we need to perform transformation into global coordinate system to get global position. The vehicle trajectory corresponding to the test steering angles and small equal torques is shown in fig. 3.2. As we can see, although the input torques are negligible, the output longitudinal velocity in the long term increases (fig. 3.3), thus the model with symmetric steering does not return to the original zero y position.

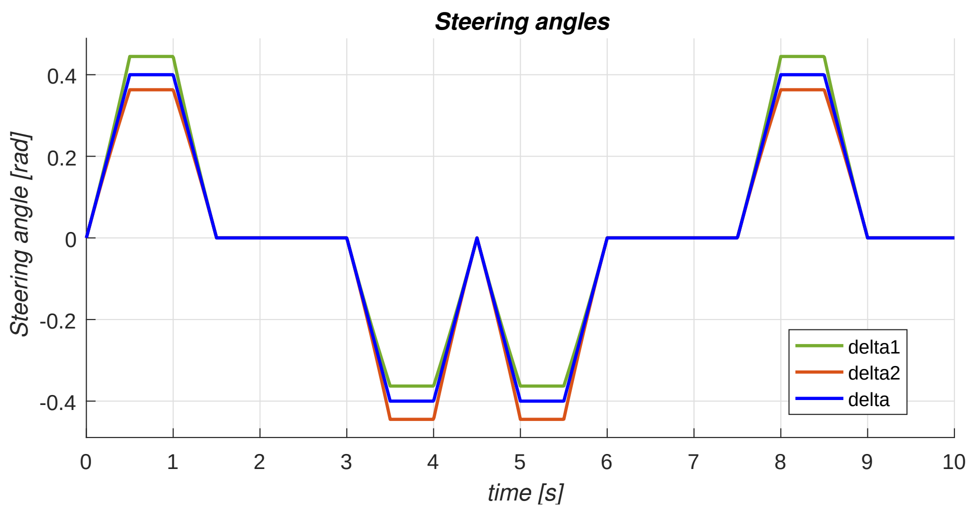

Figure 3.1: Steering angle inputs

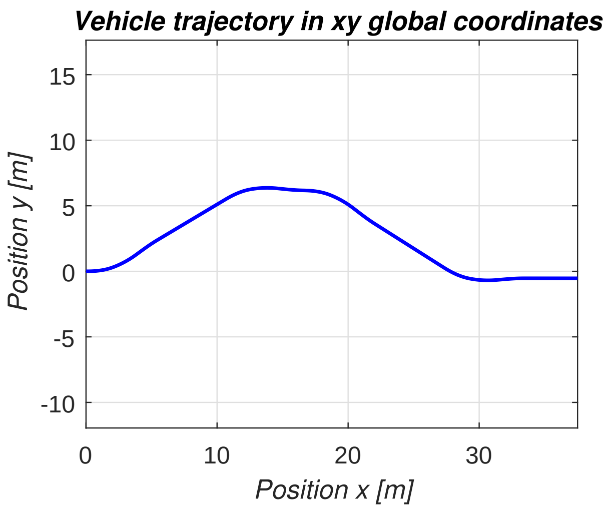

Figure 3.2: Model verification

Our simplified model does not take into account aerodynamic forces or even rolling resistance, it merely has to overcome lateral forces acting on the vehicle's front axle, projected into the longitudinal direction. And as can be observed in fig. 3.3, the speed locally declines. The initial conditions for state space integrator are chosen in the equation form 3.1 according to our previous convention.

$$z_{i_0} = \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{\psi}_0 \end{bmatrix} = \begin{bmatrix} v_{x_0} \\ 0 \\ 0 \end{bmatrix} \tag{3.1}$$

The validation example in fig. 3.2 is considered as a lane change maneuver.

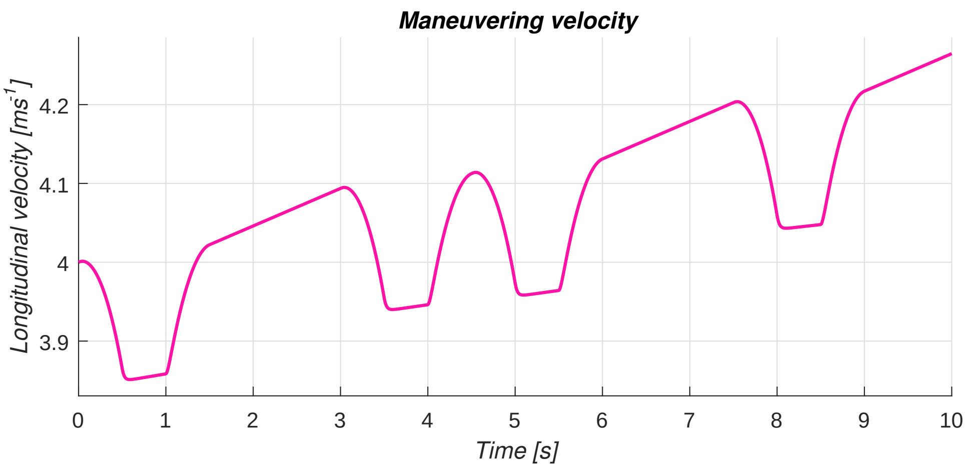

Figure 3.3: Maneuvering velocity

3.2 Trimming and linearization

In order to apply a linear predictive control to our nonlinear system, we need to perform a linearization around an appropriate point. We commonly choose equilibrium points, where all our state derivatives become zero. If we rewrite equations 2.44 - 2.47 considering being in the equilibrium point and simplify the expressions, we get:

$$0 = \sum_i \left[\frac{u_{i_{eq}}}{r_w} \cos \delta_{i_{eq}} - F_{z_i} \mu(\alpha_i) \sin \delta_{i_{eq}} \right] + m z_{2_{eq}} z_{3_{eq}} \tag{3.2}$$

$$0 = \sum_i \left[\frac{u_{i_{eq}}}{r_w} \sin \delta_{i_{eq}} + F_{z_i} \mu(\alpha_i) \cos \delta_{i_{eq}} \right] - m z_{1_{eq}} z_{3_{eq}} \tag{3.3}$$

$$0 = \sum_i \left[x_i \left(\frac{u_{i_{eq}}}{r_w} \sin \delta_{i_{eq}} + F_{z_i} \mu(\alpha_i) \cos \delta_{i_{eq}} \right) - y_i \left(\frac{u_{i_{eq}}}{r_w} \cos \delta_{i_{eq}} - F_{z_i} \mu(\alpha_i) \sin \delta_{i_{eq}} \right) \right] \tag{3.4}$$

$$\alpha_i = \delta_{i_{eq}} - \arctan \left(\frac{z_{2_{eq}} + z_{3_{eq}} x_i}{z_{1_{eq}} - z_{3_{eq}} y_i} \right) \tag{3.5}$$

So far, we have considered our inputs to be the applied torques, which we want to control and perform the optimization with. Nevertheless, the other input to our system is a steering angle δ, which has to be distinguished and perceived as a different input, namely uncontrollable. We will discuss consequences of this issue in next chapters.

In order to compute system states corresponding to the equilibrium point, a common practice is to deliberately choose inputs to be zero ($u_{i_{eq}} = 0 \wedge \delta_{i_{eq}} = 0$). As the script 3.3 shows, we utilize Symbolic Math Toolbox™ in MATLAB® to express our dynamic system and find the equilibrium point symbolically. Function *precompute* precalculates all necessary variables used in the script. Similarly, the model can be trimmed directly from Simulink by *trim* function.

Listing 3.3: Finding an equilibrium point

```
syms z1 z2 z3 u1 u2 u3 u4 d1 d2
precompute; % symbolic precalculations

fz1 = (1/m)*(F1t*cos(d1) - F1s*sin(d1) + F2t*cos(d2)
    - F2s*sin(d2) + F3t + F4t + m*z2*z3);
fz2 = (1/m)*(F1t*sin(d1) + F1s*cos(d1) + F2t*sin(d2)
    + F2s*cos(d2) + F3s + F4s - m*z1*z3);
fz3 = (1/Izz)*(x1*(F1t*sin(d1) + F1s*cos(d1))
    - y1*(F1t*cos(d1) - F1s*sin(d1)) + x2*(F2t*sin(d2)
    + F2s*cos(d2)) - y2*(F2t*cos(d2) - F2s*sin(d2))
    + x3*F3s - y3*F3t + x4*F4s - y4*F4t);

% find an equilibrium point
assume(z1 > 0)
u1 = 0; u2 = 0; u3 = 0; u4 = 0; d1 = 0; d2 = 0;
f1 = eval(fz1); f2 = eval(fz2); f3 = eval(fz3);
[r1, r2, r3] = solve([0 == f1, 0 == f2, 0 == f3],
                [z1, z2, z3]);
```

We linearize now the nonlinear system around the computed equilibrium point to obtain a continuous-time model time-invariant state space representation 3.6, 3.7.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A_C}\boldsymbol{x}(t) + \boldsymbol{B_C}\boldsymbol{u}(t) \tag{3.6}$$

$$\boldsymbol{y}(t) = \boldsymbol{C_C}\boldsymbol{x}(t) + \boldsymbol{D_C}\boldsymbol{u}(t) \tag{3.7}$$

Considering state space generally in the form 3.8, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^p$, we approximate the right term of the state equation with Taylor series expansion around the equilibrium point, neglecting second and higher order terms 3.9.

$$\begin{bmatrix} \dot{\boldsymbol{x}}(t) \\ \boldsymbol{y}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) \end{bmatrix} \tag{3.8}$$

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{x_{eq}}, \boldsymbol{u_{eq}}) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}}\right|_{(\boldsymbol{x_{eq}}, \boldsymbol{u_{eq}})} (\boldsymbol{x} - \boldsymbol{x_{eq}}) + \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{u}}\right|_{(\boldsymbol{x_{eq}}, \boldsymbol{u_{eq}})} (\boldsymbol{u} - \boldsymbol{u_{eq}}) \tag{3.9}$$

$$\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x},\boldsymbol{u})}{\partial x_1} & \frac{\partial f_1(\boldsymbol{x},\boldsymbol{u})}{\partial x_2} & \cdots & \frac{\partial f_1(\boldsymbol{x},\boldsymbol{u})}{\partial x_n} \\ \frac{\partial f_2(\boldsymbol{x},\boldsymbol{u})}{\partial x_1} & \frac{\partial f_2(\boldsymbol{x},\boldsymbol{u})}{\partial x_2} & \cdots & \frac{\partial f_2(\boldsymbol{x},\boldsymbol{u})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\boldsymbol{x},\boldsymbol{u})}{\partial x_1} & \frac{\partial f_n(\boldsymbol{x},\boldsymbol{u})}{\partial x_2} & \cdots & \frac{\partial f_n(\boldsymbol{x},\boldsymbol{u})}{\partial x_n} \end{bmatrix} \tag{3.10}$$

$$\frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x},\boldsymbol{u})}{\partial u_1} & \frac{\partial f_1(\boldsymbol{x},\boldsymbol{u})}{\partial u_2} & \cdots & \frac{\partial f_1(\boldsymbol{x},\boldsymbol{u})}{\partial u_p} \\ \frac{\partial f_2(\boldsymbol{x},\boldsymbol{u})}{\partial u_1} & \frac{\partial f_2(\boldsymbol{x},\boldsymbol{u})}{\partial u_2} & \cdots & \frac{\partial f_2(\boldsymbol{x},\boldsymbol{u})}{\partial u_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\boldsymbol{x},\boldsymbol{u})}{\partial u_1} & \frac{\partial f_n(\boldsymbol{x},\boldsymbol{u})}{\partial u_2} & \cdots & \frac{\partial f_n(\boldsymbol{x},\boldsymbol{u})}{\partial u_p} \end{bmatrix} \tag{3.11}$$

By substituting the Jacobian matrices 3.10, 3.11 into the Taylor expansion, then since:

$$\boldsymbol{f}(\boldsymbol{x_{eq}}, \boldsymbol{u_{eq}}) = \boldsymbol{0} \tag{3.12}$$

and by comparing the terms with general state equation we obtain:

$$\boldsymbol{A_C} = \frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{x}}\bigg|_{(\boldsymbol{x_{eq}},\boldsymbol{u_{eq}})} \tag{3.13}$$

$$\boldsymbol{B_C} = \frac{\partial \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}{\partial \boldsymbol{u}}\bigg|_{(\boldsymbol{x_{eq}},\boldsymbol{u_{eq}})} \tag{3.14}$$

The output equation could be derived and compared similarly in general case. As we consider the states to be the observed outputs and assume no feedthrough dependence, we define $\boldsymbol{C_C}$ as an identity matrix and $\boldsymbol{D_C}$ as a zero matrix.

$$\boldsymbol{C_C} = \boldsymbol{I_n} \tag{3.15}$$

$$\boldsymbol{D_C} = \boldsymbol{0_{n,p}} \tag{3.16}$$

Listing 3.4: Computing continuous-time linearized state space

```
eqp; % load symbolic equations and previous results
syms z1 z2 z3 u1 u2 u3 u4 d1 d2 d

% compute A, B, C matrices
A = jacobian([fz1; fz2; fz3], [z1 z2 z3]);
B = jacobian([fz1; fz2; fz3], [u1 u2 u3 u4 d1 d2]);
C = diag(ones(3,1));

% substitute one steering angle input
d1 = acot(cot(d) - w/(2*l));
d2 = acot(cot(d) + w/(2*l));
B(:,5) = B(:,5) + B(:,6); B(:,6) = []; % merge d1,d2
A = eval(A); B = eval(B);

% feed equilibrium point in
z1 = r1; z2 = r2; z3 = r3; u1 = 0; u2 = 0;
u3 = 0; u4 = 0; d = 0;
A = eval(A); B = eval(B);
```

The script 3.4 returns matrices for continuous-time state space. We obtain the exact same results by applying function *linmod* with arguments corresponding to the same equilibrium point.

3.3 Discrete state space

Since the predictive control is computationally demanding, we often use computing devices to implement it. And although there are some methods of how to control continuous or hybrid systems, we are going to control a discrete plant, into which we need to transfer our system.

If we rearrange and multiply the equation 3.6 with matrix $e^{-\boldsymbol{A_C}t}$, we get:

$$e^{-\boldsymbol{A_C}t}\dot{\boldsymbol{x}}(t) - e^{-\boldsymbol{A_C}t}\boldsymbol{A_C}\boldsymbol{x}(t) = \frac{d}{dt}\left(e^{-\boldsymbol{A_C}t}\boldsymbol{x}(t)\right) = e^{-\boldsymbol{A_C}t}\boldsymbol{B_C}\boldsymbol{u}(t) \tag{3.17}$$

which after integration yields the analytical solution in continuous time in the following form.

$$\boldsymbol{x}(t) = e^{\boldsymbol{A_C}t}\boldsymbol{x}(0) + \int_0^t e^{\boldsymbol{A_C}(t-\tau)}\boldsymbol{B_C}\boldsymbol{u}(\tau)d\tau \tag{3.18}$$

A discrete equivalent for difference between two samples considering zero-order hold ($\boldsymbol{u}(\tau) = \boldsymbol{u}(kT) = const.$) and while defining variable $\lambda = (k+1)T - \tau$ for $\lambda \in [0, kT)$ can be easily foreseen:

$$\boldsymbol{x_{(k+1)T}} = e^{\boldsymbol{A_C}T}\boldsymbol{x_{kT}} + \int_0^T e^{\boldsymbol{A_C}\lambda}\boldsymbol{B_C}\boldsymbol{u_{kT}}d\lambda \tag{3.19}$$

And by comparing 3.19 with discrete state equation 3.20, we calculate discrete system matrix and discrete input matrix as the functions of sample time period.

$$\boldsymbol{x_{(k+1)T}} = \boldsymbol{A}\boldsymbol{x_{kT}} + \boldsymbol{B}\boldsymbol{u_{kT}} \tag{3.20}$$

$$\boldsymbol{A} = \boldsymbol{A}(T) = e^{\boldsymbol{A_C}T} \tag{3.21}$$

$$\boldsymbol{B} = \boldsymbol{B}(T) = \int_0^T e^{\boldsymbol{A_C}\lambda}\boldsymbol{B_C}d\lambda = (e^{\boldsymbol{A_C}T} - \boldsymbol{I_n})\boldsymbol{A_C^{-1}}\boldsymbol{B_C} \tag{3.22}$$

We further use a following expansion into power series while neglecting higher order terms in order to avoid computing inversion from a singular matrix, which occurs, if the system is not stable.

$$e^{\boldsymbol{A_C}\lambda} = \boldsymbol{I_n} + \boldsymbol{A_C}\lambda + \frac{\boldsymbol{A_C^2}\lambda^2}{2!} + ... + \frac{\boldsymbol{A_C^k}\lambda^k}{k!} + ... \tag{3.23}$$

Since there is no integration in equation 3.7, the equivalent discrete output equation takes the algebraic form 3.24 and output and feedthrough matrices are straightforward.

$$\boldsymbol{y_{kT}} = \boldsymbol{C}\boldsymbol{x_{kT}} + \boldsymbol{D}\boldsymbol{u_{kT}} \tag{3.24}$$

$$\boldsymbol{C} = \boldsymbol{C_C} \tag{3.25}$$

$$\boldsymbol{D} = \boldsymbol{D_C} \tag{3.26}$$

Listing 3.5: Continuous to discrete state space conversion

```
function [A, B] = conti2disc(Ac,Bc,Ts)
[rA,cA] = size(Ac); [rB,cB] = size(Bc);
if (rA == cA && rA == rB)         % check matrix dimensions
    In = diag(ones(rA,1));
    A = In + Ac*Ts + Ac^2*Ts^2/2;
    B = In*Bc*Ts + Ac*Bc*Ts^2/2 + Ac^2*Bc*Ts^3/6;
else
    return
end
end
```

We get the same discrete linearized state space model by applying function *dlinmod*, or alternatively, we can convert continuous to discrete system by calling *c2d* function.

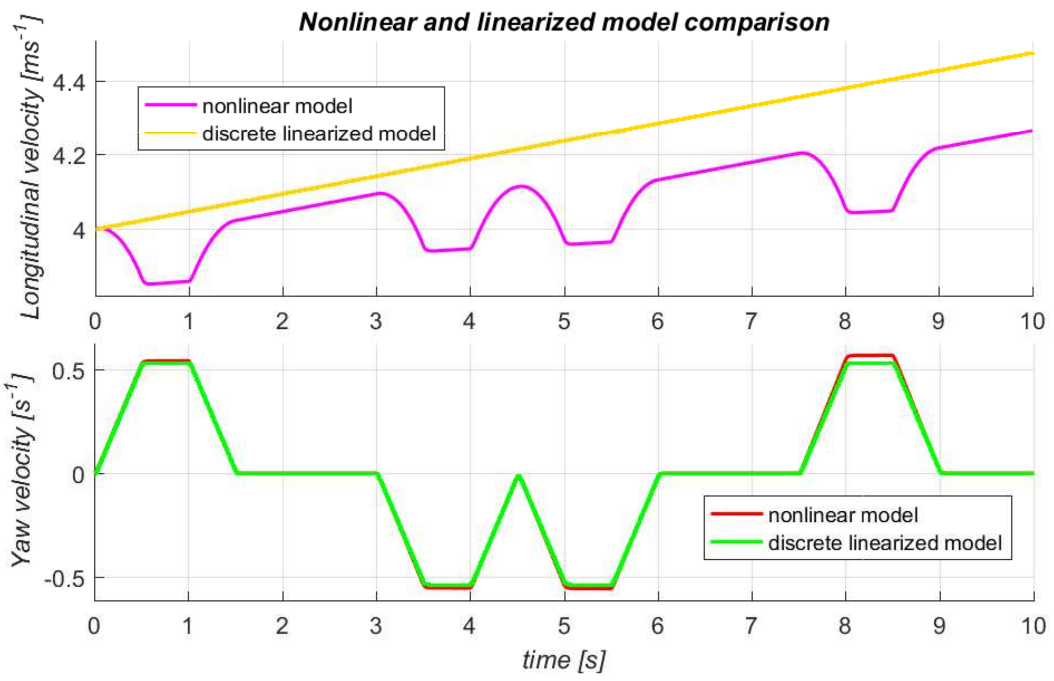

Figure 3.4: Comparing nonlinear and linearized dynamic model

In order to prove consistency between the original nonlinear dynamic model and our obtained discrete linearized state space system, we create a simple model substitution of A.1 with block *Discrete State-Space* in A.2 by specifying control matrices, sample time period and the same initial conditions. The results of comparison can be observed in fig. 3.4, where the states z_1 and z_3 according to our convention are monitored. As expected, the linear model approximates our system well around initial conditions and by time the error increases. Nevertheless, strong nonlinearities cannot be tracked at all.

Part II

Predictive control

Chapter 4

Algorithm design and solving

4.1 Introduction

Predictive control or Model predictive control (MPC) is an advanced approach of control methods having been largely developed since around 1980s. It does not represent a specific strategy or algorithm, but rather describes an explicit utilization of system dynamic model to obtain a control input while optimizing a cost function.

MPC has been successfully implemented in many different application areas, originally in process industry and more recently in robotics and electronics. This is a logical progression, since predictive control methods are more computationally demanding and in order to grasp shorter time constants in electronic systems, the effective implementation is greatly determined by hardware performance. MPC can handle various dynamic systems even with large complexity, transportation delays or instability and is suitable to be deployed onto multivariable systems. The control law is relatively easy to derive and can be further extended by constraints. The important attribute, which it possesses, is that it outputs a control signal with respect to future behaviour of the system, similarly to a linear-quadratic regulator (LQR). In contrast, a common PID regulator reflects only the past errors.

At this point we introduce a concept of prediction horizon, which must be defined with substantial care to be able to track the key dynamics of a process, otherwise the MPC may not see in advance an important dynamic direction and will certainly not respond well or even at all. The term *receding horizon* is commonly used for predicting future outputs, which means that by going forward at every step we update the predictions into future and take into account most recent measurements, thus the prediction horizon stays the same and moves relatively with the current control step.

MPC adopts a system state feedback as a term appearing in the cost function, which we minimize, generally in the following form:

$$J_{MPC}(\boldsymbol{u}) = \sum_{i=0}^{N} \left[(\boldsymbol{y_{k+i}})^T \boldsymbol{Q} (\boldsymbol{y_{k+i}}) + (\boldsymbol{u_{k+i-1}})^T \boldsymbol{R} (\boldsymbol{u_{k+i-1}}) \right] \tag{4.1}$$

where $\boldsymbol{y_{k+i}}$ designates system output at sample $k+i$, similarly for $\boldsymbol{u_{k+i-1}}$, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are weighting matrices. It is important to notice that we sum over N elements and to distinguish MPC from objective function used for LQR 4.2.

$$J_{LQR}(\boldsymbol{u}) = \sum_{i=0}^{\infty} \left[(\boldsymbol{y_{k+i}})^{\boldsymbol{T}} \boldsymbol{Q} (\boldsymbol{y_{k+i}}) + (\boldsymbol{u_{k+i-1}})^{\boldsymbol{T}} \boldsymbol{R} (\boldsymbol{u_{k+i-1}}) \right] \tag{4.2}$$

It can be demonstrated that by applying LQR the stability is achieved because of the infinite horizon. However, in case of finite-horizon MPC the stability is not inherently guaranteed and thus we either analyse its stability a posteriori, or we extend the cost function with terminal penalty weight $\boldsymbol{Q_P}$, which is basically solution of the Riccatti equation.

4.2 Prediction within state space model

We derive the expression for output predictions of a general discrete MIMO (multiple-input multiple-output) state space model without any noise, commonly described by following equations:

$$\boldsymbol{x_{k+1}} = \boldsymbol{A x_k} + \boldsymbol{B u_k} \tag{4.3}$$

$$\boldsymbol{y_k} = \boldsymbol{C x_k} + \boldsymbol{D u_k} \tag{4.4}$$

where $\boldsymbol{u} \in \mathbb{R}^p$ is the input vector with p inputs, $\boldsymbol{x} \in \mathbb{R}^n$ the state vector with n states, $\boldsymbol{y} \in \mathbb{R}^q$ the output vector with q outputs, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ the state matrix, $\boldsymbol{B} \in \mathbb{R}^{n \times p}$ the input matrix, $\boldsymbol{C} \in \mathbb{R}^{q \times n}$ the output matrix and $\boldsymbol{D} \in \mathbb{R}^{q \times p}$ is the feedthrough matrix. The index k indicates the sample at time k, index $k+1$ at time $k+1$, ...

We may come out of 4.3 and from this one-step ahead prediction we may find recursively an n-step ahead prediction.

$$\boldsymbol{x_{k+n}} = \boldsymbol{A}^n \boldsymbol{x_k} + \boldsymbol{A}^{n-1} \boldsymbol{B u_k} + \boldsymbol{A}^{n-2} \boldsymbol{B u_{k+1}} + ... + \boldsymbol{A B u_{k+n-2}} + \boldsymbol{B u_{k+n-1}} \tag{4.5}$$

If we can afford to further assume $\boldsymbol{D} = \boldsymbol{0}$ and we substitute 4.5 into 4.4, the n-step ahead prediction for system outputs is then:

$$\boldsymbol{y_{k+n}} = \boldsymbol{C A}^n \boldsymbol{x_k} + \boldsymbol{C}(\boldsymbol{A}^{n-1} \boldsymbol{B u_k} + \boldsymbol{A}^{n-2} \boldsymbol{B u_{k+1}} + ... + \boldsymbol{A B u_{k+n-2}} + \boldsymbol{B u_{k+n-1}}) \tag{4.6}$$

At this point it is rather appropriate to introduce a modified notation to our expressions, because if we want to predict into future at a different sample than k as above, it might not be obvious which samples shall we use. Therefore, the literature comes up with double subscript $\boldsymbol{y_{k+l|k+m}}$, which tells us we predict the $(k+l)$th sample of $\boldsymbol{y}$ at sample $k+m$. With reference to double subscripts, we rewrite 4.5 and 4.6 for the same sample k accordingly.

$$\boldsymbol{x_{k+n|k}} = \boldsymbol{A}^n \boldsymbol{x_k} + \boldsymbol{A}^{n-1} \boldsymbol{B u_{k|k}} + \boldsymbol{A}^{n-2} \boldsymbol{B u_{k+1|k}} + ... + \boldsymbol{A B u_{k+n-2|k}} + \boldsymbol{B u_{k+n-1|k}} \tag{4.7}$$

$$\begin{aligned} \boldsymbol{y_{k+n|k}} = \boldsymbol{C A}^n \boldsymbol{x_k} + \boldsymbol{C}(\boldsymbol{A}^{n-1} \boldsymbol{B u_{k|k}} + \boldsymbol{A}^{n-2} \boldsymbol{B u_{k+1|k}} + ... \\ + \boldsymbol{A B u_{k+n-2|k}} + \boldsymbol{B u_{k+n-1|k}}) \end{aligned} \tag{4.8}$$

We notice that the expressions 4.7 and 4.8 contain term $\boldsymbol{x_k}$ without double subscript indicating it is a measurement, not a prediction. In order to make our equations further more compact, we introduce another notation 4.9 for our predictions. It represents a vector (possibly of vectors - MIMO systems) with first predicted sample comprised in the index and with arrow indicating past, or in our case future prediction. There is no information regarding number of future samples being computed, we choose the horizon completely independently as a certain degree of freedom of our prediction.

$$\underrightarrow{\boldsymbol{x_{k+1}}} = \begin{bmatrix} \boldsymbol{x_{k+1|k}} \\ \boldsymbol{x_{k+2|k}} \\ \vdots \\ \boldsymbol{x_{k+n|k}} \end{bmatrix} \tag{4.9}$$

Considering computing all the states up to the sample $k+n$ with the compact notation and rewriting 4.7, we obtain:

$$\underrightarrow{\boldsymbol{x_{k+1}}} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^n \end{bmatrix} \boldsymbol{x_k} + \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{n-1}\boldsymbol{B} & \boldsymbol{A}^{n-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \underrightarrow{\boldsymbol{u_k}} \tag{4.10}$$

And similarly with 4.8:

$$\underrightarrow{\boldsymbol{y_{k+1}}} = \begin{bmatrix} \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^n \end{bmatrix} \boldsymbol{x_k} + \begin{bmatrix} \boldsymbol{CB} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{CAB} & \boldsymbol{CB} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA}^{n-1}\boldsymbol{B} & \boldsymbol{CA}^{n-2}\boldsymbol{B} & \cdots & \boldsymbol{CB} \end{bmatrix} \underrightarrow{\boldsymbol{u_k}} \tag{4.11}$$

Introducing matrices $\boldsymbol{P}$ and $\boldsymbol{H}$, we abbreviate 4.11 into:

$$\underrightarrow{\boldsymbol{y_{k+1}}} = \boldsymbol{P}\boldsymbol{x_k} + \boldsymbol{H}\underrightarrow{\boldsymbol{u_k}} \tag{4.12}$$

4.3 Unbiased prediction with desired outputs

Considering obtaining the desired outputs (our reference) from our system, the state space equations without any disturbances must satisfy the following:

$$\boldsymbol{x_{ds}} = \boldsymbol{A}\boldsymbol{x_{ds}} + \boldsymbol{B}\boldsymbol{u_{ds}} \tag{4.13}$$

$$\boldsymbol{w} = \boldsymbol{y_{ds}} = \boldsymbol{C}\boldsymbol{x_{ds}} \tag{4.14}$$

where $\boldsymbol{y_{ds}}$ represent the desired system outputs at the steady state, $\boldsymbol{x_{ds}}$ corresponding states and $\boldsymbol{u_{ds}}$ appropriate inputs. The system unknowns are designated then directly by 4.15:

$$\begin{bmatrix} \boldsymbol{x_{ds}} \\ \boldsymbol{u_{ds}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{A} - \boldsymbol{I_n} & \boldsymbol{B} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{y_{ds}} \\ \boldsymbol{0} \end{bmatrix} \tag{4.15}$$

We further define unbiased (relative to the desired state) outputs, states and inputs and since we consider linear state space, which holds the superposition principle, we introduce an unbiased state space representation 4.19, 4.20 and an unbiased state space output prediction 4.21.

$$\hat{\boldsymbol{y}}_{\boldsymbol{k}} = \boldsymbol{y}_{\boldsymbol{k}} - \boldsymbol{y}_{\boldsymbol{ds}} \tag{4.16}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \boldsymbol{x}_{\boldsymbol{k}} - \boldsymbol{x}_{\boldsymbol{ds}} \tag{4.17}$$

$$\hat{\boldsymbol{u}}_{\boldsymbol{k}} = \boldsymbol{u}_{\boldsymbol{k}} - \boldsymbol{u}_{\boldsymbol{ds}} \tag{4.18}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{k+1}} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{B}\hat{\boldsymbol{u}}_{\boldsymbol{k}} \tag{4.19}$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{k}} = \boldsymbol{C}\hat{\boldsymbol{x}}_{\boldsymbol{k}} \tag{4.20}$$

$$\underrightarrow{\hat{\boldsymbol{y}}_{\boldsymbol{k+1}}} = \boldsymbol{P}\hat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{H}\underrightarrow{\hat{\boldsymbol{u}}_{\boldsymbol{k}}} \tag{4.21}$$

4.4 Objective function

Model predictive control performs an optimization at each control step regarding the state feedback and control inputs. The objective measure of the system relations contains an objective (cost) function, as introduced by 4.1. This function can generally take up different forms, however, the equation in our instance is selected utterly deliberately.

We often use the single terms in a quadratic form, since the quadratic function always has a unique minimum, thus the optimization outputs no ambiguity. When considering multiple system variables this term is a product of vector and its transposed identity. And we multiply each of them with an appropriate weighting matrix, which is capable of penalising vector elements with different coefficients. With a reference to the previous section, the purpose of controlling a system is to obtain a desired output as a steady state, therefore, it is natural to minimize tracking errors and manipulated inputs, generally unbiased values. The literature presents a suitable generic objective function accordingly:

$$J(\boldsymbol{u}) = \sum_{i=0}^{N} \left[(\boldsymbol{e}_{\boldsymbol{k+i}})^{\boldsymbol{T}} \boldsymbol{Q} (\boldsymbol{e}_{\boldsymbol{k+i}}) + (\boldsymbol{u}_{\boldsymbol{k+i}})^{\boldsymbol{T}} \boldsymbol{R} (\boldsymbol{u}_{\boldsymbol{k+i}}) + (\boldsymbol{\Delta u}_{\boldsymbol{k+i}})^{\boldsymbol{T}} \boldsymbol{R}_{\boldsymbol{\Delta}} (\boldsymbol{\Delta u}_{\boldsymbol{k+i}}) \right] \tag{4.22}$$

where $(\boldsymbol{e}_{\boldsymbol{k+i}})$ denotes the tracking errors 4.23, $(\boldsymbol{u}_{\boldsymbol{k+i}})$ the input deviations 4.24 and $(\boldsymbol{\Delta u}_{\boldsymbol{k+i}})$ the rate of input change 4.25.

$$\boldsymbol{e}_{\boldsymbol{k+i}} = \boldsymbol{y}_{\boldsymbol{k+i+1|k}} - \boldsymbol{w}_{\boldsymbol{k+i+1|k}} \tag{4.23}$$

$$\boldsymbol{u_{k+i}} = \boldsymbol{u_{k+i|k}} - \boldsymbol{u_{k+i|k(ds)}} \tag{4.24}$$

$$\boldsymbol{\Delta u_{k+i}} = \boldsymbol{u_{k+i|k}} - \boldsymbol{u_{k+i-1|k}} \tag{4.25}$$

We simplify the objective function for our case by neglecting the rate of input change and adjust the notation to match our statements, namely 4.21 (considering summation over i indicating the horizon).

$$J(\underrightarrow{\boldsymbol{\hat{u}_{k+i}}}) = \sum_{i=0}^{N} \left[(\underrightarrow{\boldsymbol{\hat{y}_{k+i+1}}})^{\boldsymbol{T}} \boldsymbol{Q} (\underrightarrow{\boldsymbol{\hat{y}_{k+i+1}}}) + (\underrightarrow{\boldsymbol{\hat{u}_{k+i}}})^{\boldsymbol{T}} \boldsymbol{R} (\underrightarrow{\boldsymbol{\hat{u}_{k+i}}}) \right] \tag{4.26}$$

4.5 Generalised predictive control

Generalised predictive control (GPC) is an MPC control method, which has all the above mentioned principles applicable for MPC. The main difference is that it provides an analytical control law if no constraints are considered. GPC algorithm was proposed originally in 1987 by Clarke, Mohtadi and Tuffs.

If we use previously derived terms and substitute 4.21 into 4.26 we obtain:

$$J(\underrightarrow{\boldsymbol{\hat{u}_{k+i}}}) = \sum_{i=0}^{N} \left[(\boldsymbol{P\hat{x}_k} + \boldsymbol{H}\underrightarrow{\boldsymbol{\hat{u}_{k+i}}})^{\boldsymbol{T}} \boldsymbol{Q} (\boldsymbol{P\hat{x}_k} + \boldsymbol{H}\underrightarrow{\boldsymbol{\hat{u}_{k+i}}}) + (\underrightarrow{\boldsymbol{\hat{u}_{k+i}}})^{\boldsymbol{T}} \boldsymbol{R} (\underrightarrow{\boldsymbol{\hat{u}_{k+i}}}) \right] \tag{4.27}$$

We find the objective function minimum by setting its gradient with respect to the future input vectors to zero.

$$\frac{\partial J(\underrightarrow{\boldsymbol{\hat{u}_{k+i}}})}{\partial \underrightarrow{\boldsymbol{\hat{u}_{k+i}}}} = \boldsymbol{0} \tag{4.28}$$

Since we assume matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ to be diagonal, then $\boldsymbol{Q} = \boldsymbol{Q^T}$ and $\boldsymbol{R} = \boldsymbol{R^T}$, which $\forall i$ yields:

$$2(\boldsymbol{H^T Q H} + \boldsymbol{R})\underrightarrow{\boldsymbol{\hat{u}_{k+i}}} + 2\boldsymbol{H^T Q P \hat{x}_k} = \boldsymbol{0} \tag{4.29}$$

Hence for every future considered input sample i at time instance k applies the explicit control law the expression 4.30

$$\underrightarrow{\boldsymbol{\hat{u}_{k+i}}} = -(\boldsymbol{H^T Q H} + \boldsymbol{R})^{-1} \boldsymbol{H^T Q P \hat{x}_k} \tag{4.30}$$

which represents a proposed control input deviation for samples $i = 0, 1, ...N$. However, the algorithm is based on the *receding horizon* and thus updates the future control inputs at every step. Therefore, we apply only the first input sample to the control while neglecting the remaining computed proposed inputs at the same step. This is demonstrated by 4.31

$$\underrightarrow{\boldsymbol{\hat{u}_{k+i}}} = -\boldsymbol{I_1}(\boldsymbol{H^T Q H} + \boldsymbol{R})^{-1} \boldsymbol{H^T Q P \hat{x}_k} \tag{4.31}$$

where I_1 is a zero matrix with only unit diagonal elements corresponding to the input vector at first future sample.

4.6 Quadratic programming and constrained predictive control

Since all systems naturally work properly only in certain conditions, we often must ensure that the inputs comply with specified input ranges, or on the contrary we must guarantee the output limits. In those cases we need to include constraints into our predictive control algorithm design, while still optimizing the system performance.

For this purpose we introduce a quadratic programming method (QP), which can optimize a quadratic function of several variables while complying with linear constraints, as follows. We define the quadratic function $\mathcal{F}(\boldsymbol{x})$:

$$\mathcal{F}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{G}\boldsymbol{x} + \boldsymbol{f}^{\boldsymbol{T}}\boldsymbol{x} + c \tag{4.32}$$

with a symmetric positive definite matrix $\boldsymbol{G} \in \mathbb{R}^{n\times n}$, a vector $\boldsymbol{f} \in \mathbb{R}^{n}$ and a constant $c \in \mathbb{R}$. Since we want to find the function optimum while determining $\boldsymbol{x}$, the solution is completely independent of constant c and the problem can be formulated accordingly.

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{G}\boldsymbol{x} + \boldsymbol{f}^{\boldsymbol{T}}\boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{C_b}\boldsymbol{x} \leq \boldsymbol{d} \end{aligned} \tag{4.33}$$

By transferring the predictive control cost function 4.27 into the formulation 4.33 and expressing the input constraints generally, we obtain:

$$\begin{aligned} \underset{\underrightarrow{\hat{\boldsymbol{u}}_{k+i}}}{\text{minimize}} \quad & \sum_{i=0}^{N}\left[(\boldsymbol{P}\hat{\boldsymbol{x}}_k + \boldsymbol{H}\underrightarrow{\hat{\boldsymbol{u}}_{k+i}})^{\boldsymbol{T}}\boldsymbol{Q}(\boldsymbol{P}\hat{\boldsymbol{x}}_k + \boldsymbol{H}\underrightarrow{\hat{\boldsymbol{u}}_{k+i}}) + (\underrightarrow{\hat{\boldsymbol{u}}_{k+i}})^{\boldsymbol{T}}\boldsymbol{R}(\underrightarrow{\hat{\boldsymbol{u}}_{k+i}})\right] \\ \text{subject to} \quad & \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & -\boldsymbol{I} \end{bmatrix} \underrightarrow{\boldsymbol{u}_{k+i}} \leq \begin{bmatrix} \overline{\boldsymbol{u}} \\ \vdots \\ \overline{\boldsymbol{u}} \\ \underline{\boldsymbol{u}} \\ \vdots \\ \underline{\boldsymbol{u}} \end{bmatrix} \end{aligned} \tag{4.34}$$

where $\boldsymbol{I}$ denotes the identity matrix for p system inputs, $\overline{\boldsymbol{u}}$ designates the vector of corresponding fixed upper input limits and $\underline{\boldsymbol{u}}$ the vector of fixed lower limits. However, the cost function minimizes the differential values $\underrightarrow{\hat{\boldsymbol{u}}_{k+i}}$, though the limits has to be determined for the absolute input values. If we rewrite $\underrightarrow{\boldsymbol{u}_{k+i}}$ in terms of $\underrightarrow{\hat{\boldsymbol{u}}_{k+i}}$ by assuming 4.35, we alter the vector of limits by substracting the time-varying unoptimized current known system input $\boldsymbol{u_k}$, which yields 4.36 constraint.

$$\underrightarrow{\boldsymbol{u}_{k+i}} = \boldsymbol{u}_k + \underrightarrow{\hat{\boldsymbol{u}}_{k+i}} \tag{4.35}$$

By multiplying the cost function expression and rewriting it in terms of 4.33 we form 4.36, and by utilizing an appropriate solver we perform the optimization with input constraints.

$$\begin{aligned}
&\underset{\underrightarrow{\hat{\boldsymbol{u}}_{k+i}}}{\text{minimize}} \quad \sum_{i=0}^{N}\left[\frac{1}{2}(\underrightarrow{\hat{\boldsymbol{u}}_{k+i}})^T(2\boldsymbol{H}^T\boldsymbol{Q}\boldsymbol{H}+2\boldsymbol{R})(\underrightarrow{\hat{\boldsymbol{u}}_{k+i}})+(2\hat{\boldsymbol{x}}_k^T\boldsymbol{P}^T\boldsymbol{Q}\boldsymbol{H})^T(\underrightarrow{\hat{\boldsymbol{u}}_{k+i}})\right]\\
&\text{subject to} \quad \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I}\\ -\boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{I} & \cdots & \boldsymbol{0}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{0} & \boldsymbol{0} & \cdots & -\boldsymbol{I} \end{bmatrix} \underrightarrow{\hat{\boldsymbol{u}}_{k+i}} \leq \begin{bmatrix} \overline{\boldsymbol{u}}-\boldsymbol{u}_k\\ \vdots\\ \overline{\boldsymbol{u}}-\boldsymbol{u}_k\\ \underline{\boldsymbol{u}}+\boldsymbol{u}_k\\ \vdots\\ \underline{\boldsymbol{u}}+\boldsymbol{u}_k \end{bmatrix}
\end{aligned} \tag{4.36}$$

Similarly, output constraints derivation can be indicated:

$$\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I}\\ -\boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{I} & \cdots & \boldsymbol{0}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{0} & \boldsymbol{0} & \cdots & -\boldsymbol{I} \end{bmatrix} \underrightarrow{\boldsymbol{y}_{k+i+1}} \leq \begin{bmatrix} \overline{\boldsymbol{y}}\\ \vdots\\ \overline{\boldsymbol{y}}\\ \underline{\boldsymbol{y}}\\ \vdots\\ \underline{\boldsymbol{y}} \end{bmatrix} \tag{4.37}$$

where $\overline{\boldsymbol{y}}$ denotes the vector of upper output limits and $\underline{\boldsymbol{y}}$ the vector of lower limits. We abbreviate the left-side matrix with $\boldsymbol{C_y}$, the right-side vector with $\boldsymbol{d_y}$ and express the outputs in terms of input measures 4.38.

$$\boldsymbol{C_y}(\boldsymbol{P}\boldsymbol{x}_k+\boldsymbol{H}\underrightarrow{\boldsymbol{u}_{k+i}}) \leq \boldsymbol{d_y} \tag{4.38}$$

By substituting 4.35 into equation above and by separating known and unknown terms, we obtain:

$$\boldsymbol{C_y}\boldsymbol{H}\underrightarrow{\hat{\boldsymbol{u}}_{k+i}} \leq \boldsymbol{d_y}-\boldsymbol{C_y}\boldsymbol{P}\boldsymbol{x}_k-\boldsymbol{C_y}\boldsymbol{H}\boldsymbol{u}_k \tag{4.39}$$

which can be added up to the input constraints by concatenation.

4.7 Nonlinear Model Predictive Control

Predictive control strategies have been evolving continuously since the development of optimal control and various algorithm approaches have been researched so far. Although they were not being applied in their entire complexity in the early phase, nowadays, due to the sufficient hardware computation capability, very advanced predictive control techniques in combination with sophisticated optimal problem solving tools are being deployed and successfully implemented. As a further logical step from the previously presented method, a Nonlinear Model Predictive Control (NMPC) is introduced.

The structure of NMPC algorithm can be mostly interpreted analogously to the linear MPC. The nonlinear system can be generally described by:

$$\boldsymbol{x}^{+} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{4.40}$$

where $f : \boldsymbol{X} \times \boldsymbol{U} \rightarrow \boldsymbol{X}$ provides a nonlinear mapping from current states and control inputs to the next sample states. The predicted output trajectory can be calculated iteratively, as follows, up to the Nth sample, starting at $\boldsymbol{x(0)} = \boldsymbol{x_0}$ and yields the dependence upon the future control inputs up to the $(N-1)th$ sample.

$$\underrightarrow{\boldsymbol{x}^{+}_{\boldsymbol{k+1}}} = f(\underrightarrow{\boldsymbol{x_k}}, \underrightarrow{\boldsymbol{u_k}}) \quad k = 0, ..., N-1. \tag{4.41}$$

We solve at each time step the optimal control problem given by:

$$\begin{aligned} \underset{\underrightarrow{\boldsymbol{u_k}}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \mathcal{L}(\underrightarrow{\boldsymbol{x_k(x_0)}}, \underrightarrow{\boldsymbol{u_k}}) \\ \text{subject to} \quad & \underrightarrow{\boldsymbol{x_{k+1}(x_0)}} = f(\underrightarrow{\boldsymbol{x_k(x_0)}}, \underrightarrow{\boldsymbol{u_k}}) \end{aligned} \tag{4.42}$$

If we denote $\underrightarrow{\boldsymbol{u}^{\star}_{\boldsymbol{k}}} \in \mathbb{R}^N(\boldsymbol{x_0})$ as an optimal computed solution up to the $(N-1)th$ sample, we use the first future optimal input $\boldsymbol{u}^{\star}_{\boldsymbol{k}}$ as the system control input for the next step with rest of calculated values being neglected. However, as we want the optimization to converge as fast as possible and the optimal problem solvers work iteratively, we can conveniently use these values for initial estimate for the next sample calculation.

The optimal control problem 4.42 presents the basic NMPC algorithm generally without any applied constraints. While NMPC poses greater requirements onto the complexity, optimization techniques and computational demands, it is being increasingly used even in automotive and aerospace applications.

Chapter 5

Implementing predictive control algorithm

5.1 Control synthesis without MPC

In order to implement the torque vectoring predictive control algorithm properly, we need to clarify the structure of the entire control system with considering driver inputs. As depicted in fig. A.3, which we take as our reference, we assume the driver to have a full control over longitudinal velocity by observing the actual output velocity from the vehicle model and comparing it with his desired velocity. The driver applies the accelerator pedal accordingly, possibly until he reaches maximum, which is represented by a simple discrete proportional regulator and a saturation limit. The resulting torque is applied and distributed to each wheel equally.

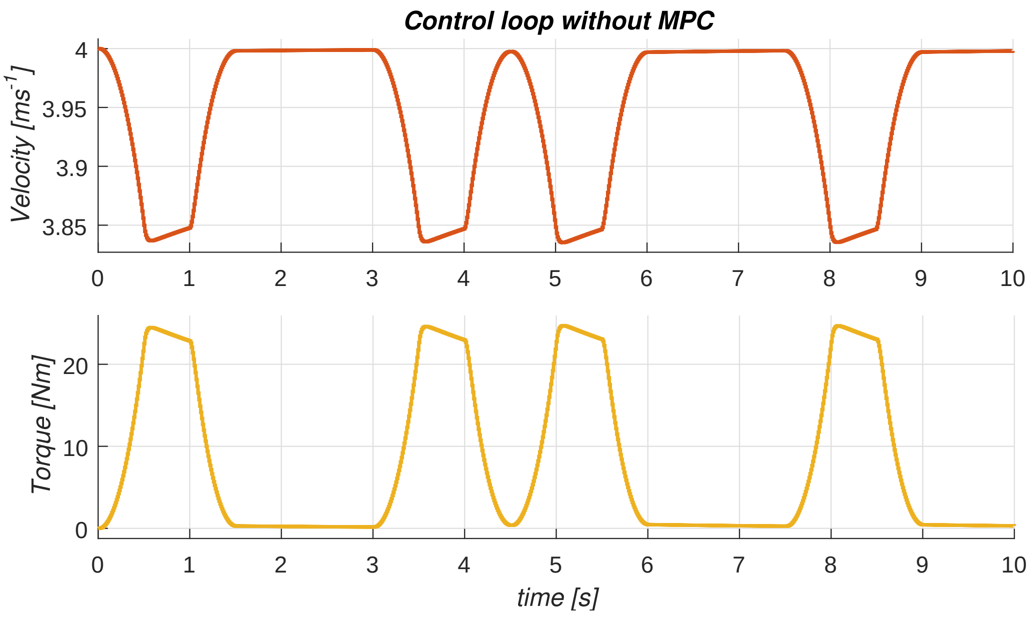

Figure 5.1: Control algorithm without MPC optimization for lane change maneuver

Assuming the vehicle has a small enough speed, by which the driver does not need to brake and wants to maintain the speed throughout the maneuver, then by a proper adjustment of the proportional regulator (K_p coefficient reflects the driver's experience), we obtain fig. 5.1. It is important to state at this point, that the driver inputs do not contain any delayed responses or imperfections.

5.2 Predictive control matrices

As we derived the predictive control matrices $\boldsymbol{P}$ and $\boldsymbol{H}$ in the expression 4.12, we precompute them before applying to the control law since their form is stationary.

Listing 5.1: Predictive control matrices

```matlab
% initializing discrete state space system
conss; clearvars -except A B m rw
Ts = 1e-4; [A, B] = conti2disc(A,B,Ts); B(:,5) = [];

% computing predictive control matrices
n = 4; % horizon parameter

    % P array
[rA, cA] = size(A);
P = zeros(rA*n, cA); % pre-allocating
subP = A;
for i = 1:n
  for j = 1:rA
    P(j+(i-1)*rA,:) = subP(j,:);
  end
  subP = subP*A; % A^n
end

    % H matrix
[rB, cB] = size(B);
H = zeros(rB*n, cB*n); % pre-allocating;
subH = B;
for l = 1:n
  for i = (1:n+1-l)
    for j = 1:rB
      for k = 1:cB
        H(j+(i-1)*rB+(l-1)*rB,k+(i-1)*cB) = subH(j,k);
      end
    end
  end
  subH = A*subH;
end

% creating system control structure
clearvars -except P H Ts m rw n
sys = struct('P',P,'H',H,'Ts',Ts,'m',m,'rw',rw);
sysBus = Simulink.Bus.createObject(sys);
```

The script 5.1 initializes discrete system and input matrices and fills them into $\boldsymbol{P}$ and $\boldsymbol{H}$ accordingly. Since the outputs of our system are the same as its states and thus matrix $\boldsymbol{C}$ is and identity matrix, we do not need to consider it in our calculations. We create a structure of the system parameters and a corresponding bus, which we feed into MPC function block A.4 as fixed parameters.

As we mentioned earlier, there is a logical mismatch between the system inputs and the inputs, which we want to control. We derived previously the generic model considering four torques and the steering angle as the model inputs. However, as we do not intend to control how the driver shall steer in order to track the desired states, as the first approximation we define the future steering angle input to be zero, just for purposes of control synthesis. This step is equivalent to erasing the appropriate column in the input matrix and thus neglecting the input sensitivity or input direction with respect to the outputs and is performed in the script (listing) 5.1.

Listing 5.2: Defining weighting matrices

```
% defining Q and R weighting matrices
r = ones(4*n,1); R = diag(r);
q = 5e6*ones(3*n,1); Q = diag(q);

% creating control structure
wgt = struct('R',R,'Q',Q);
busWgt = Simulink.Bus.createObject(wgt);
```

According to our preferences, we can arbitrarily choose weights corresponding to individual states and input vector elements, which illustrates the script (listing) 5.2. This parameter setting has a crucial impact on the control performance and behaviour. Both scripts are executed as an initialization in Simulink *InitFcn*.

5.3 Analytical control law

In order to deploy the predictive control computation, we define our desired states and calculate the corresponding deviations regarding the actual dynamic model states. We may arbitrarily set any desired state value, however, with respect to behavior and performance of the vehicle, the choice shall be based on a proper estimation of the achievable dynamic responses. We estimate the longitudinal velocity as a current measured vehicle velocity and additionally, by taking into account the actual torque demand, accordingly 5.1.

$$v_{x_{des}} = v_x + \int \frac{u}{mr_w} dt \approx v_x + \frac{u}{mr_w} kT_s \tag{5.1}$$

The purpose of the torque vectoring algorithm is to make the vehicle dynamics more efficient, namely reduce the local slip angle and thus the lateral velocity.

$$v_{y_{des}} = 0 \tag{5.2}$$

Eventually, the most abstract state to choose is the angular velocity, which determines vehicle behavior the most. We theoretically set it according to 5.3, which reflects the

Listing 5.3: Control law computation

```
function [u1,u2,u3,u4] = mpc(sys,wgt,u,v_sc,d,vx,vy,omega)

% deviation from desired state
vx_des = vx + v_sc/(sys.m*sys.rw);
vy_des = 0;
omega_des = vx*tan(d);
x_k = [vx - vx_des; vy - vy_des; omega - omega_des];

% control law
H = sys.H; P = sys.P; Q = wgt.Q; R = wgt.R;
u_k = -(H'*Q*H + R)\(H'*Q*P*x_k);
u1 = u+u_k(1);
u2 = u+u_k(2);
u3 = u+u_k(3);
u4 = u+u_k(4);
```

obvious dependence on the longitudinal velocity and the steering angle.

$$\omega_{des} = v_x \tan\delta \tag{5.3}$$

Nevertheless, it is appropriate to estimate this state more precisely on a real vehicle, based on experience of how the vehicle shall respond and adjust the control suitably. The precalculated stationary control matrices are then deployed into the analytical control law, which outputs predicted control deviation inputs. We assume to add the driver torque command as the bias to ensure the control reflects the torque proportionality of the accelerator pedal. The figure 5.2 reproduces the results of an example analytical control law computation utilizing Generalised Predictive Control algorithm.

In the previously derived predictive control matrices, we neglected the steering input sensitivity. Though it still has an impact upon the vehicle behavior. Even though we cannot directly affect the future steering angle input, it might be generally rather better to estimate the future steering value as a current value in the entire prediction horizon than set the prediction to zero. However, the control law computation does not change its form, since it has been derived for deviation variables.

Proof. If we adjust the state equation of the system to meet the requirement as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}_{tq}\boldsymbol{u}_{k_{tq}} + \boldsymbol{B}_{st}\boldsymbol{u}_{k_{st}} \tag{5.4}$$

where $\boldsymbol{B}_{tq}$ consists only of individual torque input sensitivities and $\boldsymbol{u}_{k_{tq}}$ represents merely torque vector and similarly with $\boldsymbol{B}_{st}$ and $\boldsymbol{u}_{k_{st}}$ for steering input, then we can write for the output predictions the expression 5.5, with $\boldsymbol{S}_{st}$ and $\boldsymbol{H}_{tq}$ having similar composition as $\boldsymbol{H}$ and $\boldsymbol{u}_{k_{st}}$ being a vector of future steering input values considered as the current known value for the entire horizon. By a proper cost function minimizing as explained previously, we obtain 5.6.

$$\underrightarrow{\boldsymbol{y}_{k+1}} = \boldsymbol{P}\boldsymbol{x}_k + \boldsymbol{H}_{tq}\underrightarrow{\boldsymbol{u}_{k_{tq}}} + \boldsymbol{S}_{st}\boldsymbol{u}_{k_{st}} \tag{5.5}$$

$$\underrightarrow{\hat{\boldsymbol{u}}_{(k+i)_{tq}}} = -(\boldsymbol{H}_{tq}^T \boldsymbol{Q} \boldsymbol{H}_{tq} + \boldsymbol{R})^{-1} \boldsymbol{H}_{tq}^T \boldsymbol{Q} (\boldsymbol{P} \hat{\boldsymbol{x}}_k + \boldsymbol{S}_{st} \hat{\boldsymbol{u}}_{k_{st}}) \tag{5.6}$$

Since the predicted steering angle input stays the same over the predicted horizon, the differences equal to zero vector and thus 5.8.

$$\hat{\boldsymbol{u}}_{k_{st}} = \boldsymbol{0} \tag{5.7}$$

$$\underrightarrow{\hat{\boldsymbol{u}}_{(k+i)_{tq}}} = -(\boldsymbol{H}_{tq}^T \boldsymbol{Q} \boldsymbol{H}_{tq} + \boldsymbol{R})^{-1} \boldsymbol{H}_{tq}^T \boldsymbol{Q} \boldsymbol{P} \hat{\boldsymbol{x}}_k \tag{5.8}$$

■

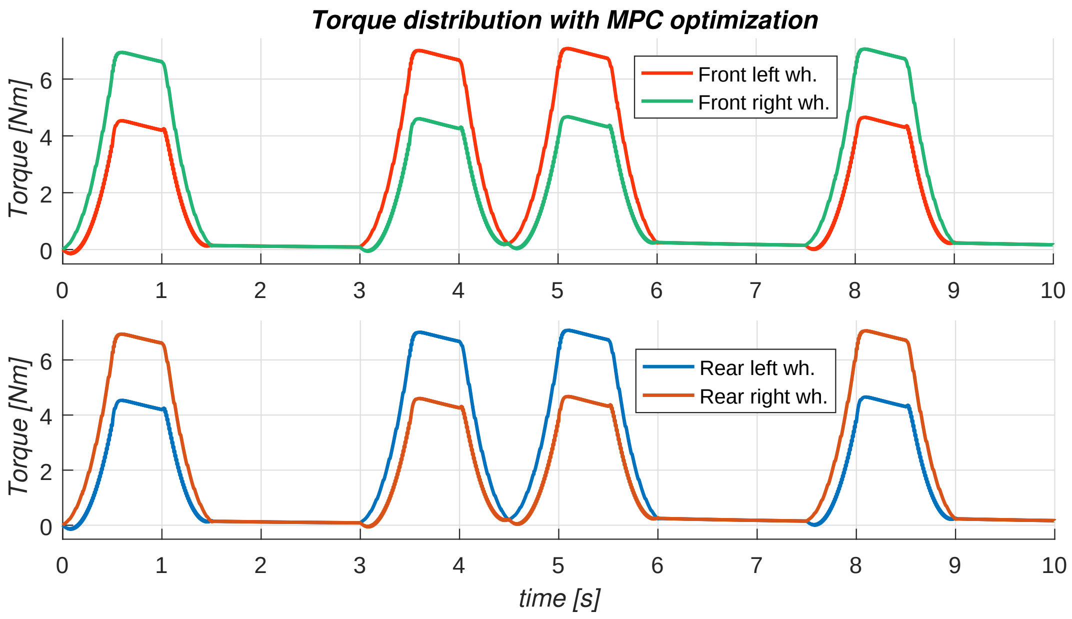

Figure 5.2: MPC algorithm example results for lane change maneuver

The significant impact upon the vehicle behavior has a choice of weighting coefficients penalising either system states or system inputs. If we assume $\boldsymbol{Q}$ and $\boldsymbol{R}$ to be identity matrices scaled by prescalers q_{pr} and r_{pr}, then by varying the ratio of the prescalers we affect the vehicle performance and it is thus adjustable, as the fig. 5.3 shows. However, the weights have to be carefully chosen, properly calibrated and tested on a real vehicle, in order to avoid possible instability.

The fig. 5.4 illustrates the advantages of Torque Vectoring and improvements of handling and vehicle stability. We track the vehicle states and inputs while cornering with the prescribed input steering angle (1st graph) both with and without Torque Vectoring System (further used as TVS). This figure (as well as fig. 5.2 for lane change maneuver) depicts the optimized torque distribution (TVS) among all wheels (2nd and 3rd graph) and indicates the higher torque deployment onto the outer wheels, which is typical for all TVS implementations and enhances the vehicle performance. In the 5th graph we observe the actual longitudinal velocity with the optimized torques of TVS and graphs 4 and 6 reflect yaw and longitudinal velocity differences, respectively, of an

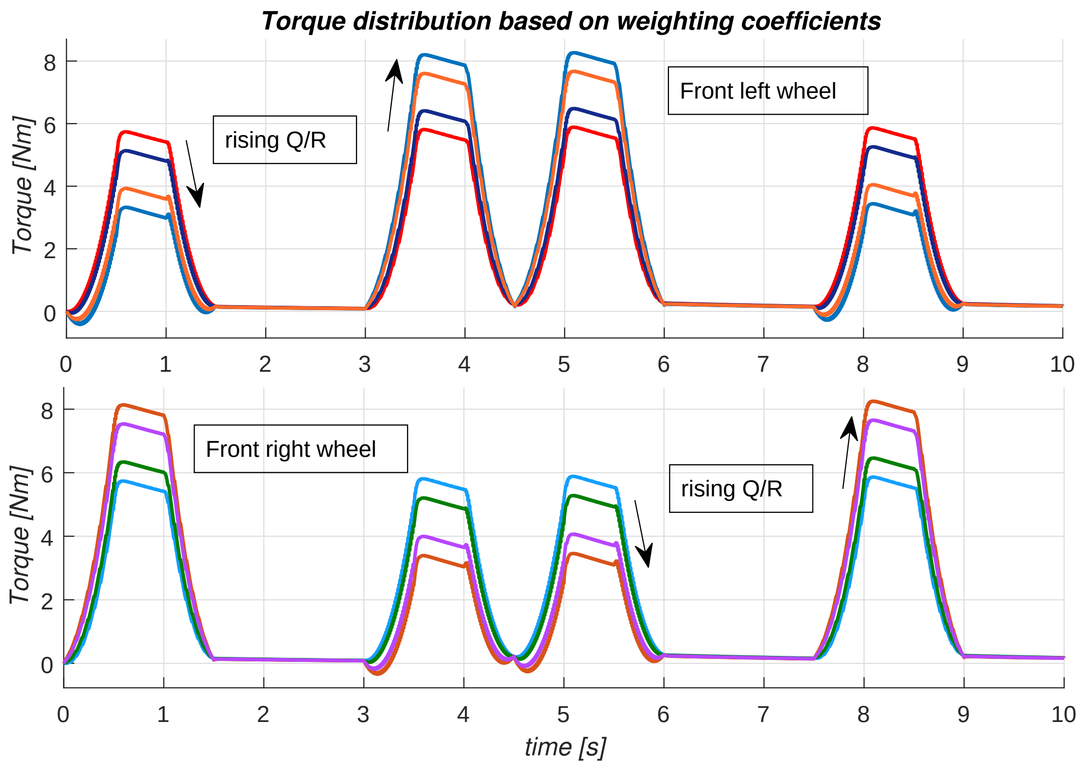

Figure 5.3: Weighting coefficients influencing torque distribution

optimized (TVS) and common solution at each time step. Applying more torque onto the outer wheels generates an additional torque onto the whole body frame and results in increasing yaw velocity (graph 4), which can be considered as a main target of this optimization, since the vehicle better follows trajectory corresponding to the defined steering angle and tries to compensate the inertial forces acting onto the body frame.

5.4 Applying quadratic programming

We further establish specific linear constraints for system input measures, modify the control parameters and test the quadratic programming method to solve the optimization. For testing purposes, we assume the upper input limit to be 5.9 with respect to our previous definitions, where $\boldsymbol{u_k}$ represents a saturated driver accelerator input. And we set the lower limit as 5.10, by which we completely cut negative torque values off and avoid possible system interaction between TVS and Anti-lock Brake System, which could be potentially anticipated.

$$\overline{\boldsymbol{u}} = k_s \boldsymbol{u_k} \tag{5.9}$$

$$\underline{\boldsymbol{u}} = \boldsymbol{0} \tag{5.10}$$

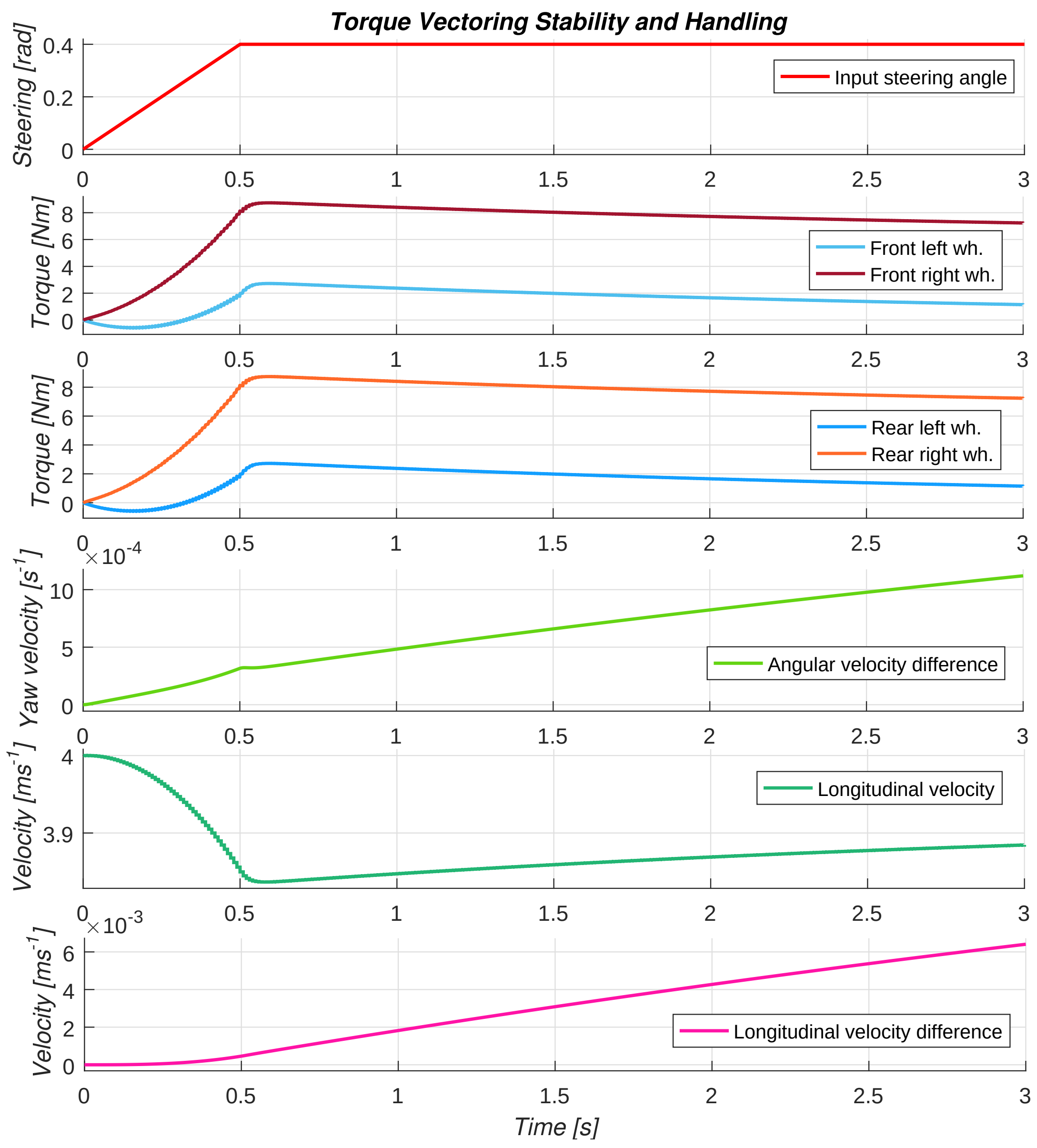

Figure 5.4: Comparison of stability and handling with and without Torque Vectoring System

The script 5.4 defines the structure for constraints with respect to number of future predicted inputs and the corresponding bus is then being deployed as depicted in A.5 into the *mpc* function 5.5. This function introduces the available quadratic programming solver *quadprog* in Matlab environment. The function *quadprog* is not supported directly for code generation, hence it must be declared as an external function in order to be called and since it returns output as type *mxArray*, the output needs to be converted by assigning to a variable of a known type.

Several example situations were simulated utilizing both analytical GPC and unconstrained MPC quadratic programming algorithm in order to compare their performance figures. Though it could not be inherently anticipated, they both yielded identical

Listing 5.4: Defining structure for constraints

```
% constraints
c = ones(4*n,1); C = diag(c);
C = vertcat(C,-C);
dupper = ones(4*n,1); dlower = zeros(4*n,1);

con = struct('C',C,'dupper',dupper,'dlower',dlower);
busCon = Simulink.Bus.createObject(con);
```

results and thus both very different methods have been verified.

Furthemore, the quadratic programming for predictive control with defined input constraints has been simulated, which shows the fig. 5.5. We observe the saturated torques by the lower input limit from constrained MPC at those points, where the GPC algorithm computes a negative torque value. The maximum applied wheel torque corresponds to k_s multiple of the given driver torque input.

Listing 5.5: Applying quadratic programming with constraints

```
function [u1,u2,u3,u4] = mpc(sys,wgt,con,u,v_sc,d,vx,vy,
   omega)

% deviation from desired state
vx_des = vx + v_sc/(sys.m*sys.rw);
vy_des = 0;
omega_des = vx*tan(d);
x_k = [vx - vx_des; vy - vy_des; omega - omega_des];

% defining constraints
dupper = con.dupper*1.5*u - u;
dlower = con.dlower + u;

% control law
H = sys.H; P = sys.P; Q = wgt.Q; R = wgt.R;
C = con.C; D = [dupper; dlower];
S = 2*H'*Q*H + 2*R;
f = (2*x_k'*P'*Q*H)';
coder.extrinsic('quadprog'); % declares external function
u_temp = quadprog(S,f,C,D);
u_k = zeros(size(dupper)); % typecasting
u_k = u_temp;
u1 = u+u_k(1);
u2 = u+u_k(2);
u3 = u+u_k(3);
u4 = u+u_k(4);
```

Similarly, the system states can be restricted, as derived in the previous chapter, which can prevent vehicle potential instability and improve performance.

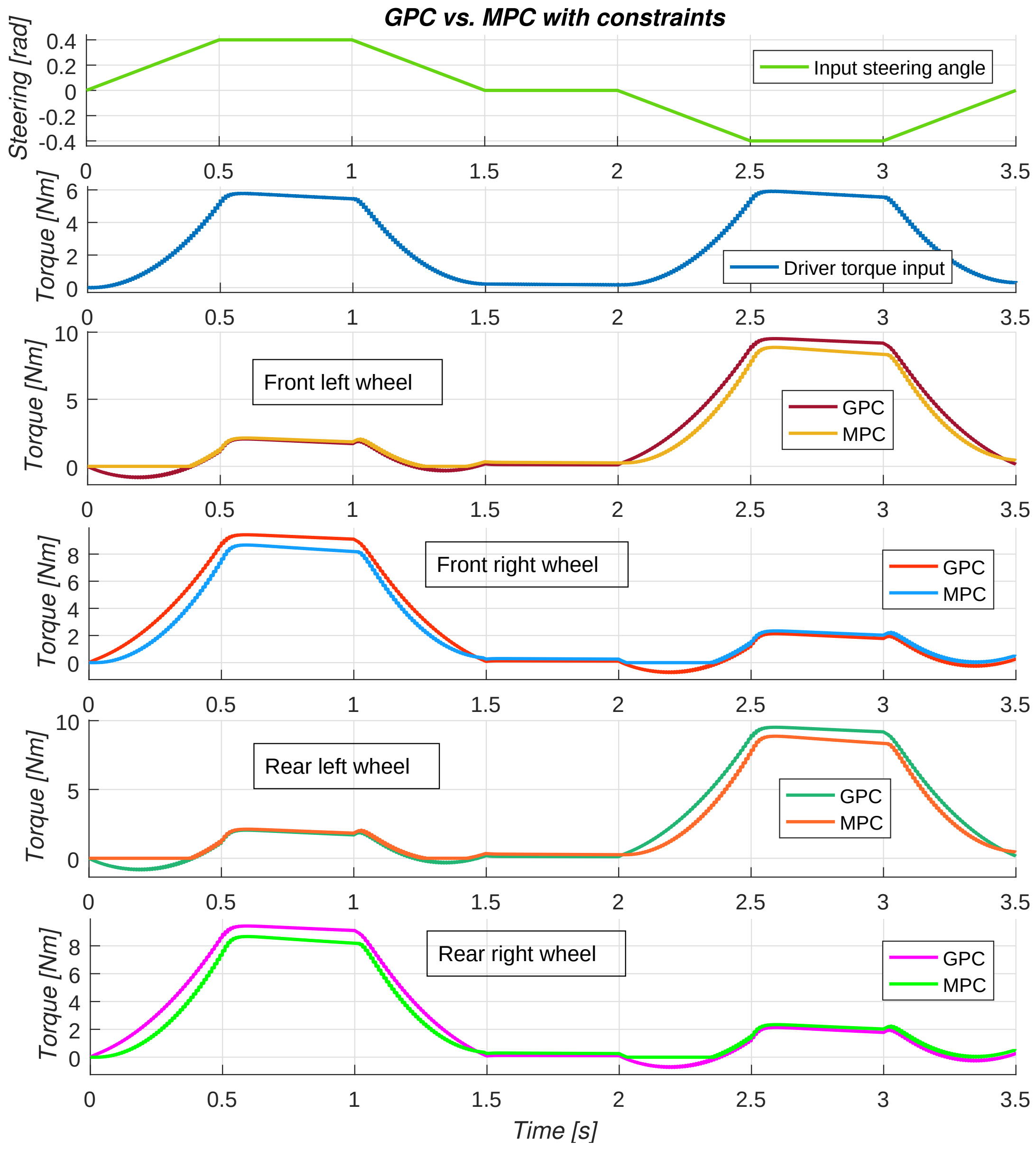

Figure 5.5: Comparing torque inputs from GPC and constrained MPC algorithm

5.5 Summary

In all previous example situations, we were utilizing the symmetrical linear state space model obtained by assuming all the inputs to be zero and thus only considering the forward vehicle motion. However, if we trim the nonlinear dynamic model around an other point, we obtain qualitatively different state space model and hence the control computation will yield different results, as the 5.6 shows. This concept introduces a potential multi-model predictive control, which can be utilized in order to improve performance of control applied to the nonlinear system, where each one of the multiple controllers operates in a different system region.

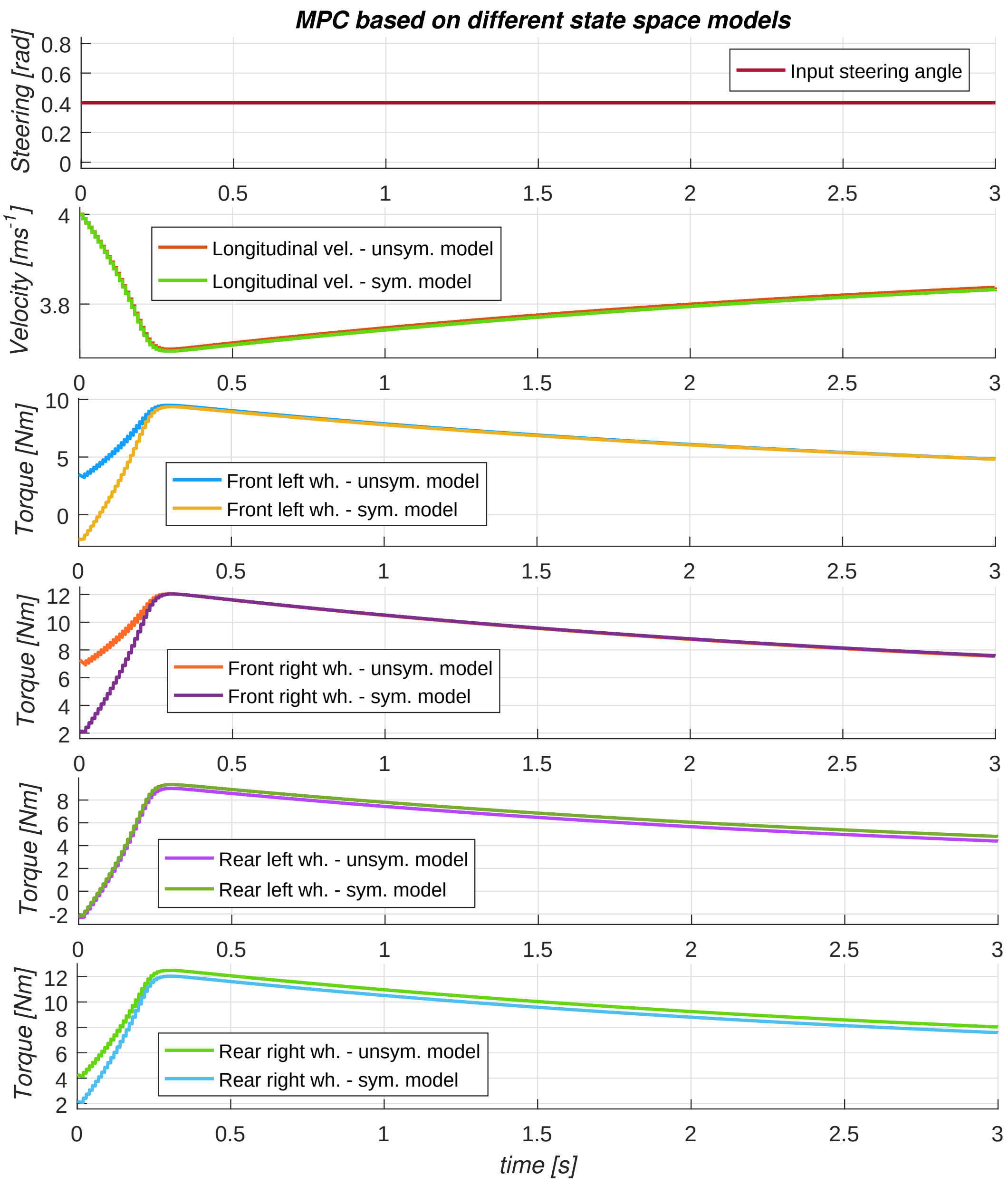

Figure 5.6: Comparison of symmetric and unsymmetric control

Chapter 6

Conclusions and future work

The content of this thesis describes a procedure and approaches for development of a torque vectoring algorithm for an all-wheel drive electric vehicle with utilizing of a simplified planar vehicle dynamic model and predictive control strategies. It presents the necessary basic dynamic motion laws, creation and validation of a nonlinear dynamic model with respect to the expected drive configuration, namely four electric motors actuating each half axle. Furthermore, it introduces the principles of trimming and linearizing of the model for linear control synthesis and by deploying the derived predictive control algorithms also the example results and analysis.

The performed simulations yielded convincing results and thus proved suitabilility of this control approach for the torque vectoring development, which has been the principal target. Apart from simulation outputs, there have been presented many outlines for further extensions and development. In order to correctly implement the system onto the real vehicle, a proper estimation of the desired behavior must be undertaken and the weighting coefficients must be reasonably chosen and calibrated, possibly even made adjustable to deliberately alter the vehicle performance. Since we can mathematically enforce the model to behave in any way, an improper set of weighting coefficients may destabilize the vehicle by applying unreasonable torque inputs, unless they are appropriately constrained, hence the constrained model predictive control for both inputs and outputs shall be deployed. As we know from practical experiences with real torque vectoring systems of the car manufacturers, the systems provide many benefits to the vehicle performance figures and will thus be implemented more frequently in the future.

The greatest impact upon computational results has the created model. Since the whole dynamic model contains nonlinearities, it is not always an easy task to find and determine the operating point around which the system shall be linearized and furthermore, by trimming the model at the specific point we neglect perhaps an important system behavior and hence the control cannot achieve the optimum, which can lead to instability. The solution may provide an extended linear MPC controller capable of switching among various local linearized state space models and thus control parameters, according to the measured states. Such a method is called multi-model predictive control and it can handle nonlinear system dynamics very efficiently with already known techniques. In order to make the algorithm more accurate, we might apply the nonlinear model predictive control, which is based upon an iterative approach of predicting the future trajectory, but the optimization conceptually remains the same. The future is most probably heading in this direction, since it is already being successfully implemented in various cases, though there are much more complex procedures of computing the control inputs and it poses much higher requirements onto the hardware performance and optimization solvers. Though for linear model predictive control the computational demands are very high as well, especially in dealing with long prediction horizons and large and extensive systems. The optimization solver or technique must be then validated and verified in order to ensure compliance of the convergence time within the control period.

As a future development, the vehicle model shall be extended with more precise subsystems, either as it has been shown in this thesis, or while considering vertical dynamics, the model shall reflect suspension tilting when cornering, and its impacts upon the planar dynamics. In order to synthesise the control properly, there should be clearly established the desired vehicle behavior, as described within this thesis, which can be estimated and tuned by performing various tests with a real vehicle. As indicated and explained in the previous paragraph, the proper control should be based upon at least multi-model predictive control, or the nonlinear model predictive control if it guaranteed more reasonable behavior. Even though the more advanced methods can deliver better results, we rather opt for a reliable and safe solution and thus perhaps simpler, than to risk passenger safety. Measure twice, cut once!

Appendices

Appendix A

Simulink models

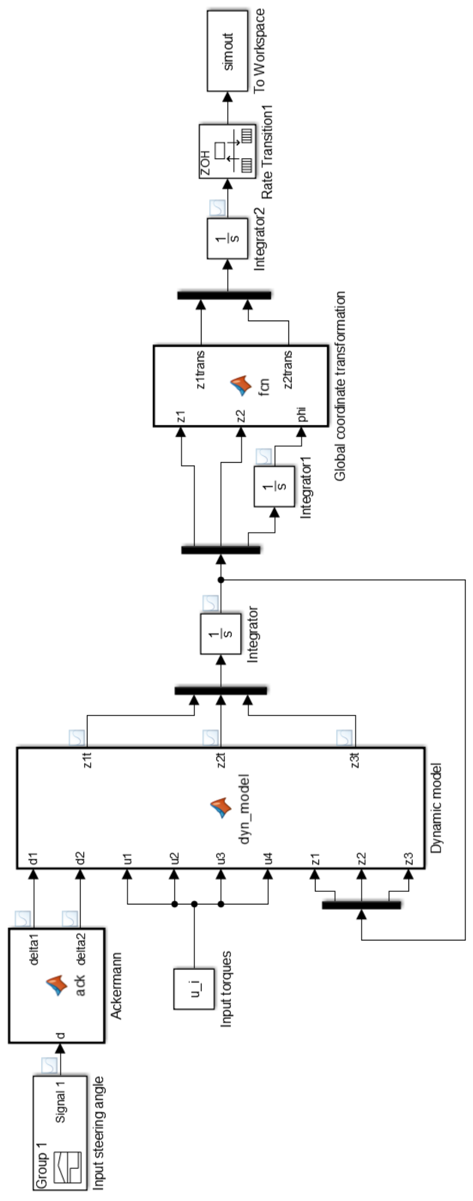

Figure A.1: Nonlinear vehicle dynamic model

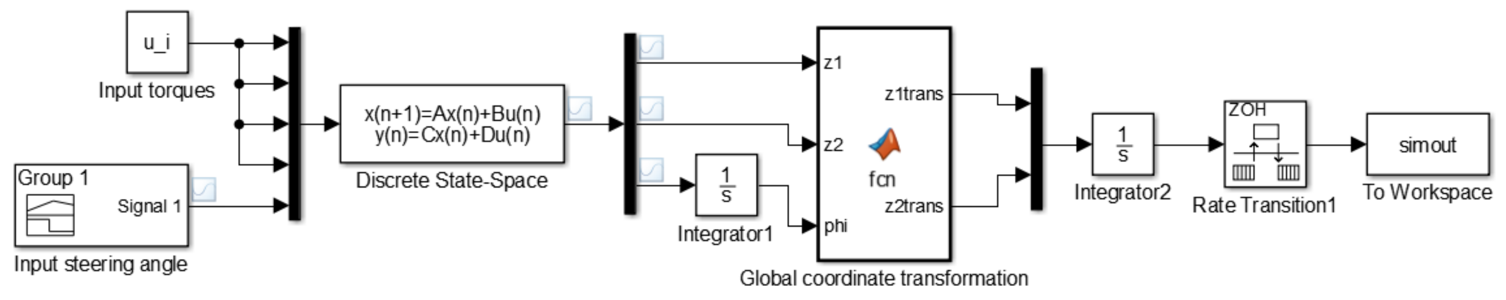

Figure A.2: Discrete linearized state space vehicle dynamic model

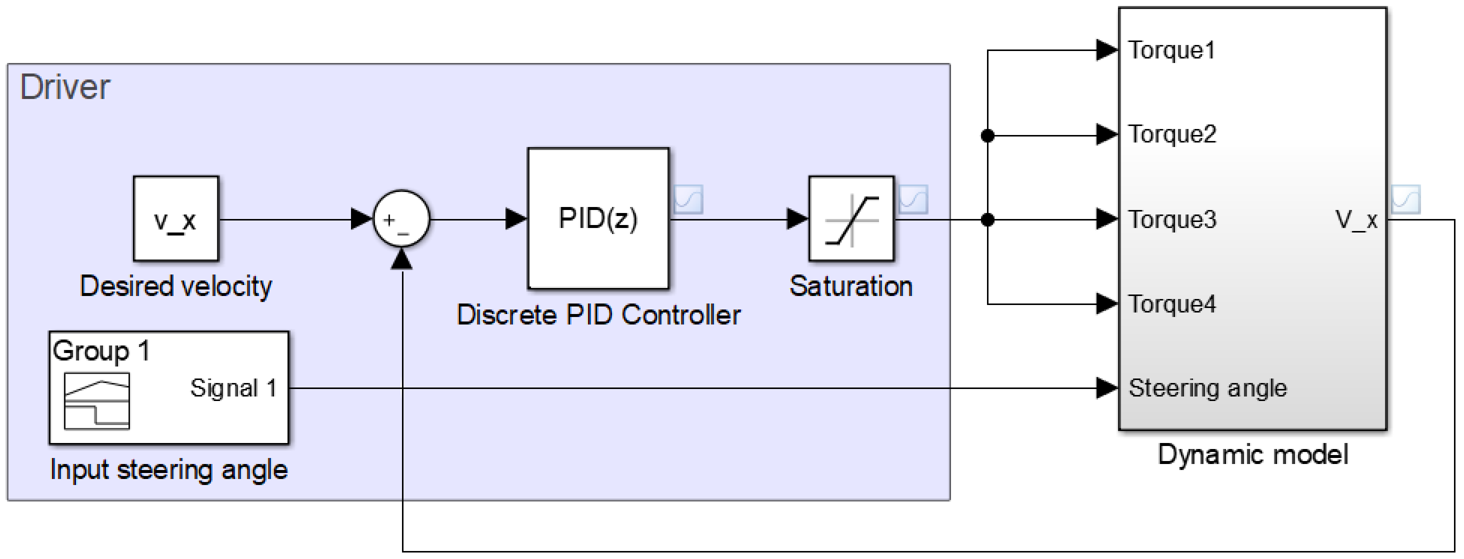

Figure A.3: Vehicle equal torque control without MPC optimization

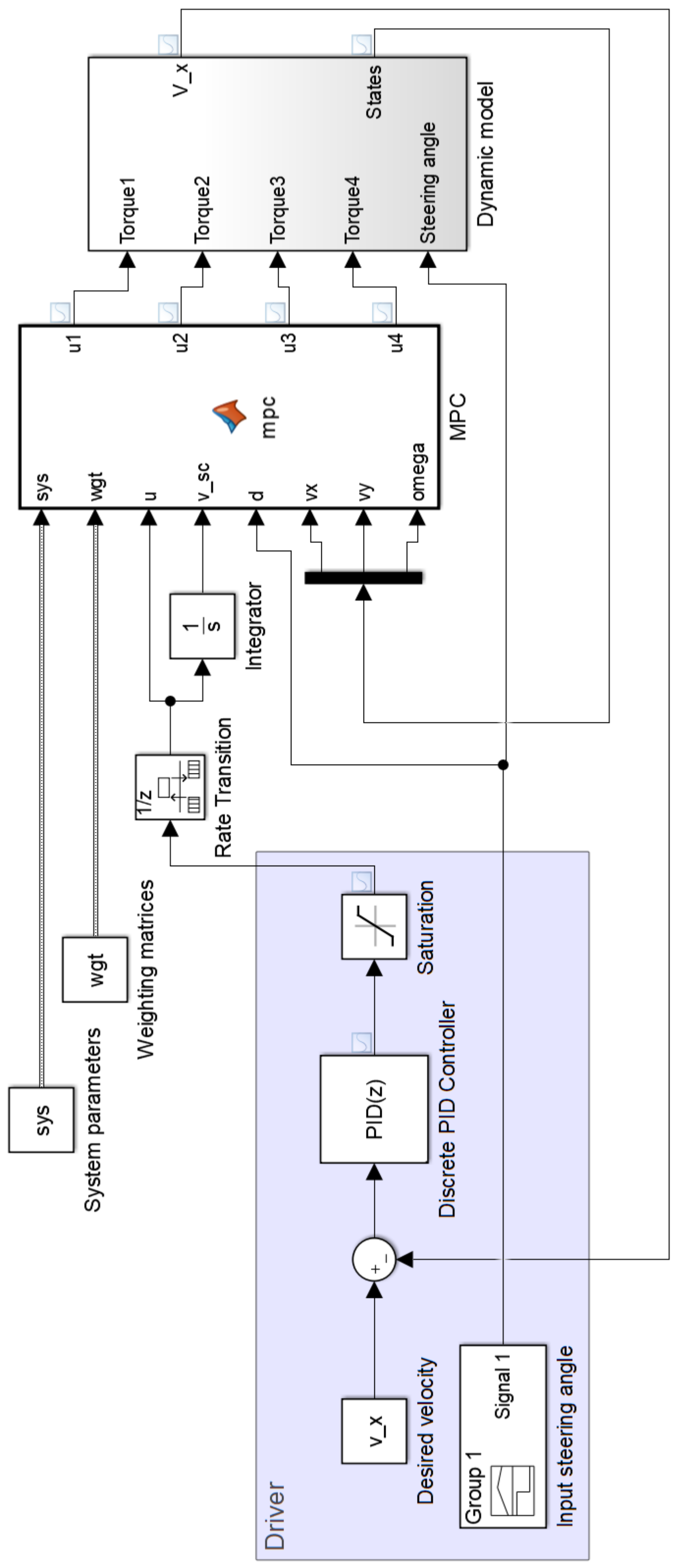

Figure A.4: Torque Vectoring MPC structure

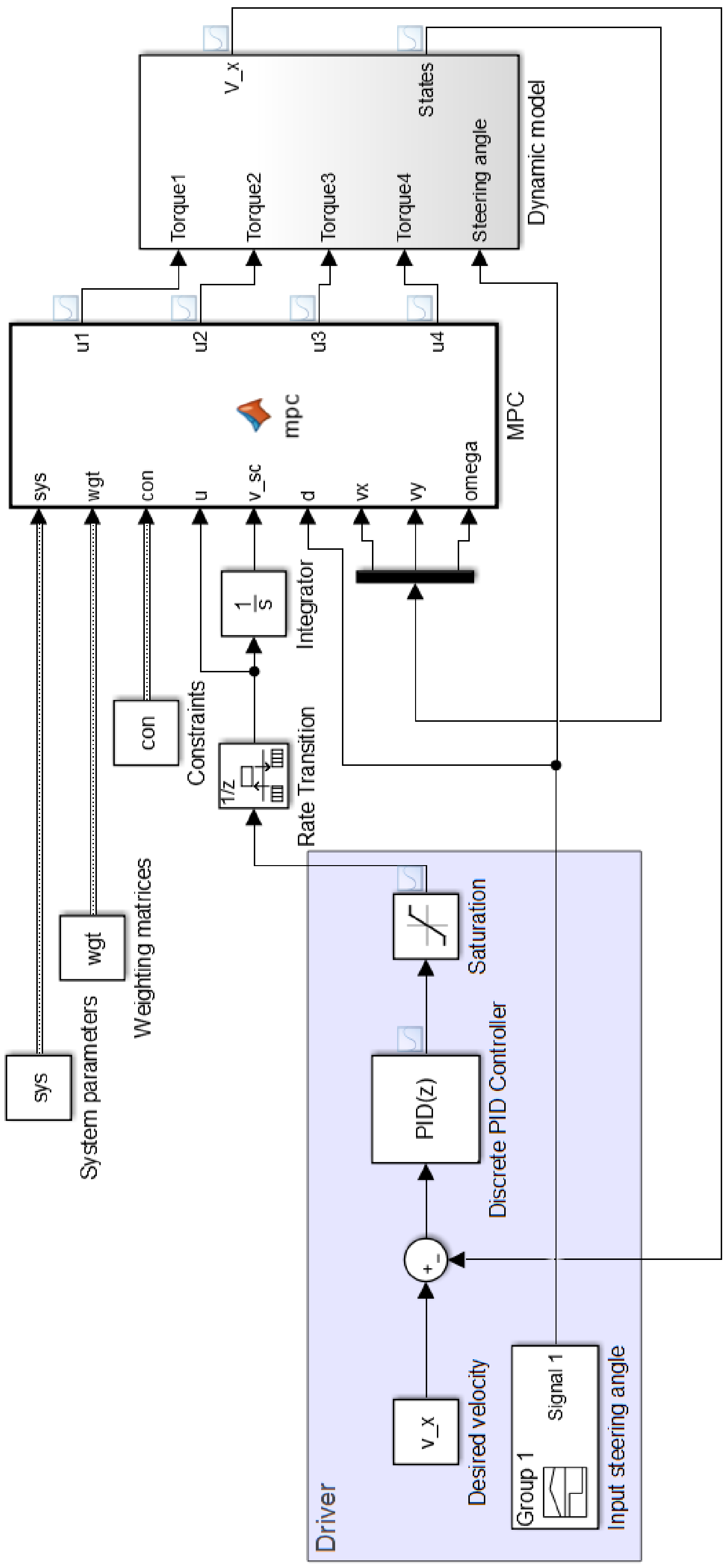

Figure A.5: Torque Vectoring MPC with constraints

Appendix B
Bibliography

[1] Reza N. Jazar. *Vehicle Dynamics: Theory and Applications*. Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA, 2008. ISBN 978-0-387-74243-4.

[2] Prokeš, Jakub. *Realtime estimation of tyre-road friction for vehicle state estimator* [online]. Master thesis. Chalmers Reproservice, Göteborg, Sweden, 2015. ISSN 1652-8557.

[3] Hans B. Pacejka. *Tire and Vehicle Dynamics*. SAE International, 2nd edition, 2005. ISBN 978-0768017021.

[4] G. Vasiljevic, S. Bogdan. *Model Predictive Control based Torque Vectoring Algorithm for Electric Car with Independent Drives* [online]. 24th Mediterranean Conference on Control and Automation (MED), Athens, Greece. IEEE 2016. ISBN 978-1-4673-8345-5.

[5] E. F. Camacho, C. Bordons. *Model Predictive control*. 2nd edition, Springer-Verlag London 2007. ISBN 978-1-85233-694-3.

[6] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. *Generalized Predictive Control. Part I. The Basic Algorithm* [online]. Automatica, 23(2):137–148, 1987.

[7] František Vlk. *Dynamika motorových vozidel*. 2nd edition, Brno, 2003. ISBN 80-239-0024-2.

[8] J. A. Rossiter. *Model-based Predictive Control. A Practical Approach*. CRC Press control series, 2004. ISBN 0-203-50396-1.

[9] Mikuláš, Ondřej. *Quadratic Programming Algorithms for Fast Model-Based Predictive Control* [online]. Bachelor thesis. Czech Technical University in Prague, Prague, 2013.

[10] L. Grüne, J. Pannek. *Nonlinear Model Predictive Control. Theory and Algorithms.* Springer-Verlag London Limited 2011. ISBN 978-0-85729-500-2.

Appendix C
Key symbols

$\boldsymbol{A}$	system matrix
$\boldsymbol{B}$	input matrix
$\boldsymbol{C}$	output matrix
$\boldsymbol{D}$	feedthrough matrix
$\boldsymbol{F}, F$	force vector, force
g	standard gravity
$\boldsymbol{I}, I$	inertia matrix, inertia
m	mass
$\boldsymbol{M}, M$	moment vector, moment
T	torque
$\boldsymbol{u}$	vector of inputs
$\boldsymbol{v}, \dot{\boldsymbol{v}}$	velocity vector, acceleration vector
$\dot{x}, \ddot{x}$	longitudinal velocity, acceleration
$\boldsymbol{x}$	vector of system states
$\dot{y}, \ddot{y}$	lateral velocity, acceleration

α	local side slip angle
β	global side slip angle
δ	steering angle
μ	rolling friction coefficient
ξ	rolling resistance coefficient
$\dot{\psi}, \ddot{\psi}$	yaw rate, acceleration
$\boldsymbol{\omega}$	angular velocity vector

Appendix D
Content of enclosed CD

```
/
├── MT_JiriTuma_2018.pdf ........................... Master's thesis in PDF
└── src ........................... compatible with Matlab R2016a and later
    ├── Modelling dynamics and linear analysis
    │   ├── nonlinear_model.slx ................... Nonlinear vehicle dynamics
    │   ├── precompute.m ........................... Symbolic precalculations
    │   ├── eqp.m ................................. Finding equilibrium point
    │   ├── con_system.m ............................ Continuous state space
    │   ├── conti2disc.m ............... Continuous to discrete conversion
    │   └── lin_model.slx ....................... Linearized vehicle dynamics
    └── Predictive control algorithm
        ├── control_unoptimized.slx ................... Control without MPC
        ├── GPC_algorithm.slx ................................ GPC algorithm
        ├── system_params.m ............................. System parameters
        ├── weights.m .................................. Weighting matrices
        ├── Constrained_MPC.slx ...................... MPC with constraints
        ├── constraints.m ........................... Constraints definition
        ├── IO_dynamic_model.slx ....... Undefined IO - prepared for trimming
        ├── trim_IO_model.m ...................... Trimming and linearizing
        ├── system_params_unsym.m ...................... Unsymmetric model
        └── MPC_unsym_model.slx ............... MPC with unsymmetric model
```