The Collinear Mecanum Drive

Matthew Thomas Watson

August 2019

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

The University Of Sheffield.

Department of Electronic and Electrical Engineering
Faculty of Engineering
The University of Sheffield
UK

To Lucy

Abstract

This thesis focuses on the modelling, analysis, and control of a novel robot locomotion system known as the Collinear Mecanum Drive (CMD). The CMD utilises three or more collinear Mecanum wheels to generate omnidirectional motion, whilst simultaneously dynamically balancing. By dynamically balancing, the ground footprint of a robot utilising a CMD can be designed to be arbitrarily thin, only lower bounded by choice of wheel diameter. The omnidirectional manoeuvrability of the CMD in combination with this narrow footprint allows for the navigation of much smaller gaps between obstacles than existing omnidirectional locomotion methods, achieved by translating directly along the common wheel rotation axis. This provides improved manoeuvrability in confined or cluttered environments. Being a dynamically balanced system, the height of the center of mass of a robot driven by a CMD can be increased without requiring a proportional increase in the width of the ground footprint so as to avoid toppling during acceleration or disturbance, as would be the case for existing statically stable omnidirectional locomotion methods. The CMD therefore promises to enable the creation of tall, space-efficient robots of more human-like form factors, that are better able to navigate the confined and cluttered environments commonly encountered in the home, office, and retail personal robotics sectors, and in the manufacturing and warehousing mobile industrial robotics sectors. This thesis derives and analyses the CMD's kinematics and dynamics models, explores a variety of control approaches, and develops the trajectory planning methods necessary for the autonomous navigation of an environment. It is hoped that this locomotion technology will see application across a range of existing and future mobile robotics sectors.

Acknowledgements

Firstly, I would like to thank my supervisor Dr Daniel Gladwin for his unwavering support and belief in me throughout the past six years of my education, for the freedom with which he has provided me during this PhD, and for his inspirational work ethic. Dr Gladwin has gone far beyond his expected supervisory duties to facilitate my development as both a researcher and a professional. I would also like to express my gratitude to Consequential Robotics and Sebastian Conran for supporting this project, and for exposing me to further interesting work during this PhD. I hope we can continue to work together to apply this research. Thank you to the University of Sheffield's Department of Electronic and Electrical Engineering, and in particular the Electrical Machines and Drives group, for their support throughout the past eight years of my education. Thank you to Tony Prescott, and to the staff and researchers of Sheffield Robotics for their support and companionship, and in particular thank you to Michael Port, whose machining and manufacturing experience greatly improved the quality of the prototype developed during this PhD. Finally, I would like to thank my wonderful partner Lucy for her boundless love and encouragement over the past six years, and I hope many more to come!

Contents

List of Figures

List of Tables

Nomenclature

α_i	Roller angle of rotation of wheel i relative to the wheel rotation axis and about a vertical axis running through the center of the wheel
B	Local body frame coordinate system
E	Global inertial frame coordinate system
$\hat{b}_x$	A unit vector along the x axis of the coordinate system B, or similar.
$\mathcal{S}_{MAS}$	Maximal admissible set
$\mathcal{L}$	Lagrangian
$\mathcal{X}_e$	The set of a dynamic system's equilibrium states
W_i	Local wheel frame coordinate system of wheel i
$\mathcal{S}_{MCAS}$	Maximal constrained admissible set
Ω_i	Angular position of the ground-contacting roller of wheel i about $\hat{r}_x$
$\overline{x}$	The maximum constrained value of x in the constraint $\lvert x\rvert \leq \overline{x}$
ϕ	CMD rotation about the vertical
$\mathbb{R}$	The set of real numbers
P	Local pendulum frame coordinate system
τ	A vector of wheel torques
θ_i	Angular position of wheel i relative to vertical about $\hat{w}_x$
θ_p	System lean angle relative to vertical

θ_{ei} Angular position of wheel i relative to the pendulum body, measured by an encoder.

$\vec{\Omega}$ A vector of angular rates about the three pendulum axes $\vec{\Omega} = \begin{bmatrix}\Omega_p & \Omega_q & \Omega_r\end{bmatrix}^T$

$\vec{\Omega}_m$ A vector of angular rates about the three pendulum axes as measured by the onboard accelerometers $\vec{\Omega}_m = \begin{bmatrix}\Omega_{m,p} & \Omega_{m,q} & \Omega_{m,r}\end{bmatrix}^T$

$\vec{a}$ A vector of accelerations along the three pendulum axes $\vec{a} = \begin{bmatrix}a_x & a_y & a_z\end{bmatrix}^T$

$\vec{a}_m$ A vector of angular rates about the three pendulum axes as measured by the onboard accelerometers $\vec{a}_m = \begin{bmatrix}a_{m,p} & a_{m,q} & a_{m,r}\end{bmatrix}^T$

$\vec{v}_{EC,B}$ The velocity of C relative to E expressed in B, or similar.

B^+ The Moore–Penrose inverse of B

C^n The set of functions for which derivatives exist and are continuous up to the nth order

c_j A control perturbation at time step j

g Acceleration due to gravity ($\mathrm{m\,s^{-2}}$)

H A Hessian matrix

h_p Height of pendulum center of mass along $\hat{p}_z$ and relative to B

h_{cm} Height of overall system center of mass along $\hat{p}_z$ and relative to B

I_{pbx} Pendulum inertia about $\hat{b}_x$

I_{pby} Pendulum inertia about $\hat{b}_y$

I_{pbz} Pendulum inertia about $\hat{b}_z$

I_{px} Pendulum inertia about $\hat{p}_x$

I_{py} Pendulum inertia about $\hat{p}_y$

I_{pz} Pendulum inertia about $\hat{p}_z$

J A cost function

K_t	Motor torque constant
k_{rw}	Coefficient of rolling friction between wheel and ground
k_{vr}	Coefficient of roller bearing viscous friction
k_{vw}	Coefficient of wheel bearing viscous friction
l_i	Location of wheel i along $\hat{b}_x$, measured from the platform's center.
m_p	Pendulum body mass
n_c	Control horizon
n_i	Independent constraint softening horizon
n_r	Reference previewing horizon
n_s	Slack variable quantity
n_w	Number of waypoints
n_w	Number of wheels
n_{con}	Constraint horizon
P	Power
p	Reduced vector of generalised coordinates
$p,\ p(t)$	A polynomial in t
Q	Quadratic state cost
Q	Q in a QR decomposition
q	Vector of generalised coordinates
R	Quadratic input cost
R	R in a QR decomposition
r	A vector of target output references
r_r	Roller radius, measured at widest point.

R_w	Motor terminal resistance
r_w	Wheel radius
R_{bp}	The rotation matrix that rotates a vector from coordinate frame P into B, or similar.
S_n	Flat output n
t	Time in seconds
T_i	Duration for which polynomial $p_i(t)$ defines a trajectory segment
T_s	Sampling period
v	Vector of system local body frame velocities
v_x	System local body frame velocity along $\hat{b}_x$
v_y	System local body frame velocity along $\hat{b}_y$
x	Component of Cartesian position in inertial frame
x	State space model state vector
y	Component of Cartesian position in inertial frame
y	State space model output
Z_k	Autonomous model state vector at timestep k

Chapter 1

Introduction

This chapter introduces the broad robotics problem that is to be addressed by the work in this thesis, and gives an outline of the structure of the thesis and the content of the chapters to follow. It concludes with a statement of the novel contributions to the literature made by the work presented in this thesis, and a list of publications.

It is a widely accepted belief that robots will continue to see ever increasing application across a wide variety of industries, ranging from industrial warehousing and manufacturing applications to personal robotics in the home and office. Whilst in previous decades this growth has been largely centered around fixed-base manipulator robots, over the past number of years there has been significant growth in the field of mobile robotics, robots which are able to freely navigate their environments whilst performing inspection, manipulation, and human-interaction tasks.

The environments in which mobile robots have to operate can be divided into two categories: structured, man-made environments, such as that found within indoor spaces, and the unstructured environments encountered in the natural world, with each possessing significantly different properties and challenges from a mobile robotics perspective. Indoor environments are typically designed around their human users; ground surfaces are usually smooth and solid, with the exception of stairs, and obstacles such as walls, doors, and furniture are sized and located as to be comfortably navigated and worked around by humans. This sets a lower bound on the minimum width of free space left between obstacles that a mobile robot must navigate. Compared to the height of a human this may be

a relatively small gap, as the smallest dimension of a standing human's projection onto the ground plane will typically be around a sixth of their height [1], a relatively large ratio. These indoor environments are in contrast to the unconstrained environments encountered in the natural world, which feature rough and discontinuous terrain, compliant ground materials, and three-dimensional obstacles, with no bounds on the required minimum navigable gap.

The structured nature of indoor environments also extends to the task space in which a mobile robot's manipulators, sensors, and interfaces must operate. Again, these spaces often desired around humans; sensor-obscuring furniture extends up to heights of over a meter, equipment interfaces are located as to be operated by a sitting or standing user, and items to be reached by a robot may be stored anywhere from ground level to high shelving. Similarly, in applications where a robot is to interact with a user, such as in personal robotics or retail customer service, this interaction may need to be performed in a face-to-face manner with a standing human.

Unsurprisingly, considering the optimal indoor robot form factor and locomotion characteristics as that which best fit the above criteria points to an optimal form factor of around human-height, with a ground footprint similar to that of a human, and with the ability to instantaneously move in any direction. Whilst locomotion methods have been demonstrated which can meet these criteria, all suffer from one or more drawbacks that have prevented their mainstream adoption.

The work in this thesis aims to address this gap: presenting, analysing, and controlling a novel form of omnidirectional locomotion that is free of the faults of these existing solutions.

1.1 Thesis Outline

This thesis is split into two introductory chapters, five novel contribution chapters, and one concluding chapter.

Chapter 2 aims to familiarise the reader with some of the key concepts explored in this thesis, and gives an overview of the generalised methods used to approach these problems. Chapter 3 introduces the invention that is the focus of this thesis, the Collinear Mecanum Drive (CMD). Generalised CMD kinematic

and dynamic models are derived, and are analysed with respect to the CMD's underactuation and nonholonomy properties, nonlinear controllability, and the size of the maximum feedback linearisable subsystem. Chapter 4 describes the reasoning and methodology behind the design and development of a CMD experimental prototype, which is later used for control validation throughout this thesis. Experimental methods for the measurement of model parameter are presented, and an online full-state estimator is derived and demonstrated. Chapter 5 partially feedback linearises the CMD, and then demonstrates three nonlinear controllers for the regulation of local frame body translational velocities, inertial frame translational velocities, and global inertial frame positions, all with Lyapunov stability guarantees. Chapter 6 addresses the problem of wheel traction and the generation of toward minimum-time trajectories by the development and implementation of a constrained dual-mode model predictive controller, capable of enforcing velocity, lean angle, and wheel torque constraints. Finally, Chapter 7 shows how the dynamics model of a CMD can be made to be differentially flat, allowing the generation of approximately dynamically feasible state and input trajectories from sufficiently smooth geometric Cartesian paths. These methods are demonstrated using existing polynomial trajectory planning techniques, allowing for the fast online generation of trajectories through multiple successive Cartesian waypoints. Finally, a novel approach to this polynomial optimisation problem is demonstrated, in which sum-of-squares programming is used to enforce trajectory constraints in continuous time, yielding a significant reduction in computation time over the state-of-the-art.

1.2 Novel Contributions

The main novel contributions of this thesis are summarised below:

1. **Dynamics Modelling and Analysis of the Collinear Mecanum Drive - Chapter 3**

   This work is the literature's first derivation of the kinematics and dynamics models of the CMD. These are then analysed in terms of nonholonomy and underactuation properties, the size of the largest feedback linearisable subsystem is determined, and controllability and accessibility of the CMD are proven. An analysis of the effect of parameter choices on the CMD char-

acteristics is performed, providing generalised guidelines for the optimal selection of these parameters.

2. **Partial Feedback Linearisation of the CMD and Nonlinear Control - Chapter 5**

   Chapter 5 shows how the CMD can be partially feedback linearised by a change of input and coordinate transformation, which is then used in the derivation of three nonlinear controllers using backstepping techniques, providing Lyapunov stability and convergence guarantees. These three controllers regulate local frame body velocities, required when operating the CMD as a vehicle or teleoperated platform, and both inertial frame velocities and positions, required when the CMD is required to operate autonomously. These are all experimentally demonstrated on the CMD prototype.

3. **Constrained Model Predictive Control of the CMD - Chapter 6**

   Chapter 6 details the derivation and implementation of an online CMD model predictive controller. This incorporates optimal satisfaction of velocity, lean angle, and wheel torque constraints. A dual-mode topology is used, providing a guarantee of stability and convergence for the linearised model. This is successfully implemented in real-time, and represents both the first implementation of a model predictive controller for the position control of a CMD, as well as the literature's first demonstration of the model predictive position control of a dynamically balancing mobile system of any wheel configuration.

4. **Dynamically Feasible Trajectory Generation for the CMD using Differential Flatness - Chapter 7**

   A method is demonstrated by which the CMD dynamics model can be made to be differentially flat, achieved through a combination of suitable fictitious output selection and model simplification. This allows the derivation of dynamically feasible state trajectories from sufficiently smooth polynomials in the flat outputs, allowing for fast and dynamically feasible online trajectory planning.

5. **Constrained Trajectory Planning using Sum-of-Squares Programming - Chapter 7**

   This contribution is a novel constrained trajectory planning method, in which sum-of-squares programming is used to enforce velocity inequality constraints in continuous time, rather than in a discrete manner at a recursively grown list of points as in the existing state-of-the-art. This yields around a 40% reduction in execution time over the existing state-of-the-art.

6. **Invention of the Reaction Wheel Collinear Mecanum Drive - Chapter 8**

   This invention was proposed toward the end of this PhD, resulting in UK Patent application number 1901297.0. By the incorporation of a reaction wheel attached to the CMD body a further degree of control is introduced, which can be controlled as to make the CMD fully actuated with respect to the CMD generalised coordinates. In combination with the CMD's omnidirectional manouevrability this renders the RWCMD holonomic, yielding a system with exactly the same locomotion characteristics as existing statically stable omnidirectional platforms, whilst requiring a fraction of the ground footprint. No research was performed around this invention, as this was deemed to fall outside of the scope of this thesis. Proposed research to explore this concept is outlined in Chapter 8.

1.3 Publications

- M. T. Watson, D. T. Gladwin, and T. J. Prescott, "The Collinear Mecanum Drive: Modelling, Analysis, Partial Feedback Linearization, and Nonlinear Control," *IEEE Transactions on Robotics*, *(preprint)*, 2020. `doi.org/10.1109/TRO.2020.2977878`.

- M. T. Watson, D. T. Gladwin, T. J. Prescott, and S. O. Conran, "Dual-Mode Model Predictive Control of an Omnidirectional Wheeled Inverted Pendulum," *IEEE/ASME Transactions on Mechatronics*, vol. 24, no. 6, pp. 2964-2975, Dec. 2019. `doi.org/10.1109/TMECH.2019.2943708`.

- M. T. Watson, D. T. Gladwin, T. J. Prescott, and S. O. Conran. "Velocity Constrained Trajectory Generation for a Collinear Mecanum Wheeled

Robot" in *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, 2019, pp. 4444–4450. doi.org/10.1109/ICRA.2019.8794019.

- M. T. Watson, D. T. Gladwin, T. J. Prescott, and S. O. Conran. "Design and Control of a Novel Omnidirectional Dynamically Balancing Platform for Remote Inspection of Confined and Cluttered Environments" in *IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES)*, Hamilton, New Zealand, 2018, pp. 473–478. doi.org/10.1109/IESES.2018.8349923.
- M. T. Watson, UK Patent Application No. 1901297.0, filed 30/01/2019 by The University of Sheffield and Consequential Robotics Ltd.

Chapter 2

Wheeled Robot Locomotion

This chapter aims to introduce and frame some of the key concepts used in this thesis. It then introduces some alternative existing solutions to the problem outlined in Chapter 1, along with a brief review of some common approaches to their control and trajectory planning.

2.1 Related Concepts

A number of key concepts are now introduced to provide background and context for the rest of the thesis.

2.1.1 Manoeuvrability

The manoeuvrability of a wheeled mobile robot is defined by its kinematic constraints. These are constraints on the velocities of a system imposed by the configuration of its wheels and their prevention of any sliding motion of the wheel surface against the ground, with the assumption of constant traction. They are linear in the system velocities, and can be represented in the Pfaffian matrix form

$$A^T(q)\dot{q} = 0 \tag{2.1.1}$$

in which each row represents a single Pfaffian constraint $a_i^T(q)\dot{q} = 0$, where each a_i^T is smooth and independent.

A Pfaffian constraint is referred to as a nonholonomic constraint if it is nonintegrable, i.e. if it cannot be integrated to an equivalent set of configuration constraints $h(q) = c$ where $\frac{\partial h(q)}{\partial q} = \lambda(q)a^T(q)$, with some integrating factor $\lambda(q) \neq 0$

and integration constant c. Systems subject to one or more nonholonomic kinematic constraints are referred to as nonholonomic systems, whereas those with only holonomic or no kinematic constraints are referred to as either omnidirectional or holonomic systems. While a nonholonomic system's velocities are restricted to evolving on the null space of $A^T(q)$, unlike in systems with holonomic constraints there is no loss of accessibility in the system's configuration space. A car is a common example of a nonholonomic system due to its use of standard wheels, in which these constraints prevent direct translation in a direction orthogonal to the car's heading, with such a translation instead only achievable using a parallel parking manoeuvre. Despite these constraints, a car is still able to reach any given (x, y, ϕ) configuration in the absence of obstacles by combinations of forward motion and a varying steering angle, demonstrating the nonintegrability of its nonholonomic constraints and the retention of full configuration space accessibility. Despite this full accessibility, i.e. despite being fully controllable, Brockett's theorem shows that systems with nonholonomic kinematic constraints cannot be asymptotically stabilised to an equilibrium by smooth time invariant feedback of the form $u = u(q)$ [2]. This complicates control design, and prevents high bandwidth Cartesian position control as achieved by humans and other omnidirectional systems, yielding systems with poor navigational ability in confined environments and inefficient movement when performing precise position tracking tasks.

While in the context of kinematics it appears that all omnidirectional systems are holonomic systems, referring to a system as being holonomic implies the absence of nonholonomic constraints of any order, whereas the concepts of manoeuvrability and omnidirectionality only consider first-order kinematic constraints. These two terms are therefore not equivalent: holonomic systems form a subset of omnidirectional systems.

Systems with omnidirectional manoeuvrability typically utilize either omniwheels or Mecanum wheels[1] [3], both exampled in Figure 2.1. While omnidirectional motion can be also achieved by the use of two or more independently

[1] Mecanum wheels, invented by Swedish engineer Bengt Erland Ilon in 1972, are named after the Swedish company Mecanum AB who originally patented the concept. This patent was bought by the US Navy for logistics applications on aircraft carriers, and was later sold to KUKA for use on warehouse vehicles and robots. Mecanum wheels are also commonly referred to as Ilon or Swedish wheels.

(a) A Mecanum wheel, $\alpha = \pm\pi/4$.

(b) A Mecanum wheel, side view.

(c) An omniwheel, $\alpha = \pi/2$.

Figure 2.1: Typical omnidirectional wheels

actuated and steered regular wheels, this requires a delay before motion can be commenced whilst the wheels are aligned with the desired direction of motion. Omnidirectional wheels provide intentional directional slip orthogonal to their direction of traction by the use of a ring of unactuated rollers affixed about the circumference of the wheel. For an omniwheel these rollers have axes of rotation offset from that of the parent wheel by a rotation of 90° about an axis orthogonal to that of the parent wheel's axis of rotation, whereas for a Mecanum wheel an angle of 45° is used[2]. The rollers on both wheel types are shaped such that the projection of the wheel onto a plane normal to its rotation axis forms a perfect circle, which for omniwheels necessitates the use two separate rows of rollers, visible in Figure 2.1c. Gfrerrer derives the parametric equations required to define the desired shape of an omnidirectional wheel's rollers [4]. Ferriere compares Mecanum and omniwheels of varying construction with regards to horizontal and vertical vibration, load capacity, and step climbing ability [5], arguing that Mecanum wheels advantageously offer a continuous roller contact transition and a reduction in vertical vibration over omniwheels, but introduce a parasitic yaw torque due to the discontinuous movement of the ground contact point along each roller during wheel rotation. Dickerson shows that frictionless Mecanum wheels achieve half the traction of regular wheels [6], as half of the net ground force generated by each wheel acts along the direction of intentional slip and can therefore do no work. Systems using Mecanum wheels therefore require greater

[2]While roller angles other than 45° or 90° can also be used, no examples of wheels of any alternative angles could be found, and no name appears to be ascribed such wheels.

drive torques to achieve the same body accelerations as a similar system with regular wheels. This is exacerbated by the discontinuous ground contact point during wheel rotation, as each transition from one roller to the next provides an opportunity for the wheel to slip.

Any deviation of a Mecanum wheel's rotation axis from parallelism with the ground results in a non-circular projection, meaning they are best operated on hard flat surfaces[3], and suspended such that their rotation axis always remains parallel to the ground. Whilst the degree of imperfection in ground flatness that can be tolerated is wheel, application, and specification specific, in general any variation in wheel geometry or ground surface would manifest as a vertical vibration and a variation in wheel diameter, preventing smooth rolling. Omnidirectional wheels must be used in sets of three or more, appropriately oriented to define a unique inverse kinematic mapping. Mecanum wheels are often used in sets in which roller angles of both $\pi/4$ and $-\pi/4$ are used, allowing the definition of a unique inverse kinematic mapping whilst keeping all wheel rotation axes parallel. Provided a unique kinematic mapping and thus full controllability, any omnidirectional acceleration can be instantaneously achieved by appropriate actuation of the wheels, without requiring any prior steering to align with the desired direction of motion. This allows for omnidirectional motion using only three actuators, compared to a steered system with regular wheels that requires at least four actuators and a delay whilst its wheels are steered to their new required heading. For solutions using any kind of omnidirectional wheel, care must be taken to ensure that all of the roller rotation axes do not intersect at a single point, as this allows for uncontrolled rotation about the intersection point.

2.1.2 Stability and System Footprint

In the context of this section consider the stability of a robot to be a measure of its ability to handle disturbances that act to perturb its attitude relative to the vertical, without this causing a reduction in controllability, i.e. without non-redundant wheels losing contact with the ground. In typical vehicles this stability is attained statically by the possession of three or more ground contact points, forming the vertices of a polygon beneath the vehicle. Assuming an initially stationary system the vehicle will always return to its single statically

[3] as opposed to rough terrain such as that encountered outdoors

stable equilibrium from any attitude under the force of gravity alone[4], provided a vertical line drawn through its center of mass (CoM) intersects this ground contact polygon. This means tall systems with small footprints are more easily toppled than short, wide systems, and so there exists a minimum ratio of CoM height to minimum footprint dimension for a system to possess a given degree of disturbance rejection in all directions. If such a system is to operate on sloped surfaces or carry objects at a horizontal offset from the system's center of mass as in Figure 2.2 this ratio must be increased further, as both of these tasks act to move the system's CoM towards the edge of its footprint, reducing the margin that is left to ensure stability under disturbance. This upper bounds the degree of slope angle and mass of payload that can be tolerated without falling, even in the absence of disturbance. Similarly, this approach to attaining stability limits the maximum accelerations that can be generated by the system changing its direction of motion, requiring tall systems to always move at a sufficiently slow speed if they are required to quickly react to dynamic obstacles without falling.

Static stability can be increased by using more than three wheels to increase the minimum footprint width, though this comes at a cost of requiring the inclusion of suspension if all wheels are to maintain simultaneous ground contact. This, along with inevitable flex in the robot's structure present even in three-wheeled systems, introduces further coupled dynamics into the system. This is evident when a tall statically stable robot drives quickly over an imperfect surface, as the top of the robot tends to shift and oscillate due to excitation by perturbations from the ground. While static stability can actually be achieved using only two wheels in a differential drive configuration with a CoM below the wheel axle, this requires impractically large wheel diameters for robots of any significant height, and so are not a viable locomotion solution for human form factored indoor robotics.

Dynamically stable systems attain stability by a degeneration of this ground contact polygon into a line or point, achieved either by a collinear arrangement of contact points, or by reducing the number of contact points to two or one. Dynamically stable systems instead use active actuation to apply forces to the ground that allow them to dynamically balance about what is now an unstable

[4]Though depending on the exact shape of the footprint the CoM may still overshoot the opposite side of the footprint provided insufficient damping, so this is a necessary but not sufficient condition for static stability.

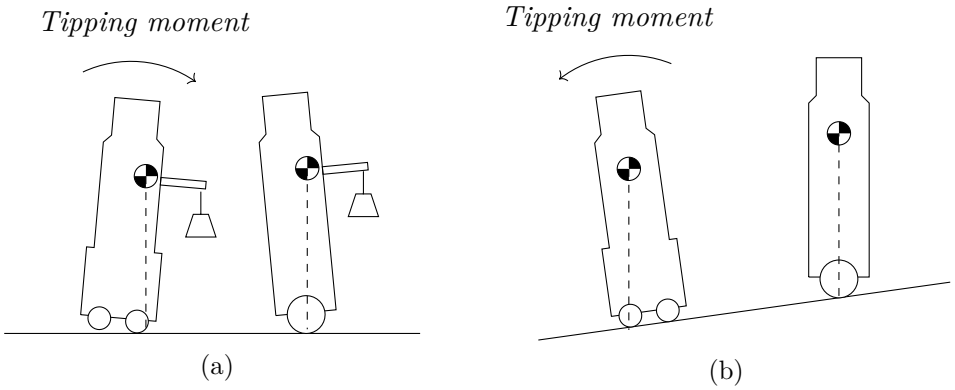


Figure 2.2: A demonstration of how statically stable systems become less stable when carrying off-axis loads or operating on sloped surfaces.

upright equilibrium. Such systems are often referred to as mobile inverted pendulums, with a common example of this being two-wheeled inverted pendulums (TWIPs) such as the Segway[5]. This elimination of a footprint dimension allows the robot to be designed to be arbitrarily thin in this dimension, allowing the creation of robots with the same height and resilience to disturbance as a statically stable robot, whilst possessing a much thinner form factor and subsequently more lightweight construction. As a dynamically balancing system is able to control the position of its CoM relative to its contact point(s) it can adapt to changes in its CoM position due to the carrying of additional loads, without the same associated reduction in stability that is experienced in statically stable systems. Similarly, as a dynamically stable system has a one or two dimensional ground contact polygon it can climb ramped surfaces without an associated change in its attitude, provided in the 2D case that the ground contact line is kept sufficiently perpendicular to the direction of slope. These two operations are exampled in Figure 2.2.

As a dynamically balanced robot must be sensitive to changes in its attitude in order to balance, intentional attitude perturbations can be used as a user input from which to control the robot. For example, a balancing robot that is controlled to regulate its velocities to zero can be guided to a new location by a gentle continuous push from a user. This is potentially a more natural way of

[5]Though technically a Segway requires a rider to become dynamically balanced, as the system alone has a CoM below its wheel axle and is therefore statically stable.

positioning or teaching a robot, as opposed to a statically stable robot that would require a joystick or strategically placed force sensors to achieve a similar effect. This concept is well demonstrated by Nagarajan [7], in which a ball-balancing robot is taught desired movement trajectories by guidance from a human using a single finger.

2.1.3 The Ideal Indoor Mobile Robot

From Chapter 1 it is known that the optimal mobile indoor robot form factor and locomotion characteristics are similar to that attained by a human. This requires a larger ratio of height to footprint size than attainable using statically stable locomotion methods, necessitating the use of a dynamically balanced solution. This is to be achieved whilst also possessing omnidirectional manoeuvrability, allowing this reduced footprint to be used to achieve improved navigation of confined environments, better position regulation during manipulation and environment interaction tasks, and increased gracefulness of motion. To maintain low complexity and cost this is to be achieved using a wheeled solution, rather than a bipedal legged approach.

2.1.4 Dynamics of Mechanical Systems

The dynamics of an open-chain of rigid bodies can be expressed in the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Q_f \tag{2.1.2}$$

where $q \in \mathbb{R}^n$ denotes a vector of generalised coordinates, $Q_f \in \mathbb{R}^n$ a vector of forces or torques acting on the generalised coordinates, $M(q) \in \mathbb{R}^{n\times n}$ a symmetric positive-definite mass matrix, $C(q,\dot{q})$ a matrix containing Coriolis and centripetal force terms, and $G(q)$ a vector capturing forces acting on the generalised coordinates due to the system's pose relative to gravity. These equations are linear in $\ddot{q}$, quadratic in $\dot{q}$, and trigonometric in q, and are not unique for a given system, allowing some derivation methods to yield 'simpler' or more convenient models than others.

Three commonly used methods exist for the derivation of these equations for a given system: the classical Newton-Euler and Lagrangian formulations, and the more recent Kane's method. While these three methods all yield equivalent

models, some models may contain fewer terms or fewer uses of transcendental functions. Comparisons of these methods typically show Kane's method yields the most numerically simple models, shortly followed by the Lagrangian method, and with the Newton-Euler method yielding a significant increase in complexity [8, 9]. The results of these studies, however, ignore the potential benefits of compiler-level optimisation, and so may not be representative of real-time implementations.

The incorporation of nonholonomic constraints into this derivation is challenging for the Newton-Euler formulation, requiring the explicit calculation of the associated constraint forces, whereas for the Lagrangian and Kane's methods more straightforward approaches exist [10, 11]. In this thesis the Lagrangian method is used, as this features more commonly in the literature than Kane's method, whilst allowing for the systematic incorporation of nonholonomic constraints.

The Lagrangian is defined as the difference of the system's total kinetic energy $\mathcal{K}(q, \dot{q})$ and potential energy $\mathcal{U}(q)$ as

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q) \tag{2.1.3}$$

from which the equations of motions can be expressed as

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = Q_f \tag{2.1.4}$$

Nonholonomic constraints can be incorporated by the use of the Lagrange equations in the form

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = Q_f + A(q)^T\lambda \tag{2.1.5}$$

where $A(q)^T$ denotes the matrix defining the k Pfaffian constraints in (2.1.1), and $\lambda \in \mathbb{R}^k$ a vector of Lagrange multipliers.

These multipliers can be eliminated by multiplication with the transpose of a matrix $\Phi(q)$ representing the null space of $A(q)$, such that $\Phi^T(q)A^T(q) = 0$, allowing the derivation of the new dynamic model

$$M(p)\ddot{p} + C(p, \dot{p})\dot{p} + G(p) = B(p)\tau \tag{2.1.6}$$

defined in a new minimal coordinate vector p, where now $p \in \mathbb{R}^{n-k}$, and $\dot{q} = \Phi\dot{p}$ [10].

2.1.5 Underactuated Mechanical Systems

Underactuated mechanical systems are rigid body systems of the form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = B(q)u \tag{2.1.7}$$

where $m = \text{rank}(B(q))$, $n = \dim(q)$, and $m < n$, i.e. systems which have fewer independent control inputs than degrees of freedom [12]. This means the input u cannot be used to independently generate forces on all q, and the system cannot be exactly linearised by feedback. Assuming by suitable choice of input and coordinates a system can be described in the form (2.1.7) in which $B(q) = \begin{bmatrix} I_m, & 0_{n-m} \end{bmatrix}^T$, the first m equations can be referred to as the fully actuated subsystem, and the latter $n - m$ equations can be referred to as the unactuated subsystem. Spong showed that the fully actuated subsystem can always be linearised by a suitable change of coordinates and control [13], termed partial feedback linearisation. However, the unactuated system remains nonlinear and coupled to the new control, still necessitating the use of nonlinear control techniques, albeit now to control a system of dimension $n - m$ rather than n. The stabilisation of such systems has been well studied [14, 15]. Olfati-Saber introduces methods for finding explicit changes of coordinates and control that transform a large range of underactuated mechanical systems into one of three cascade normal forms [16], along with a classification system for determining the required normal form based on some model properties. Generalised control techniques are then presented for these normal forms.

Balancing systems such as the inverted pendulum are classic examples of underactuated mechanical systems, in which the base of the pendulum must be actuated to translate the pendulum to some new position, whilst also regulating the pendulum's lean angle as to come to rest at the upright unstable equilibrium and remaining bounded to some neighbourhood of the upright position. Underactuated balancing systems have unstable zero dynamics, so are also referred to as non-minimum phase systems. In practice, such systems have states that when tracking a step response must first move in a direction away from that of

the step, with the position state of the base of an inverted pendulum forming a classic example. These systems cannot track arbitrary trajectories in configuration space, as their dynamics must evolve according to one or more second order nonholonomic constraints. This is in contrast to holonomic systems, which can track arbitrary trajectories in configuration space, greatly simplifying their control and trajectory planning. A number of techniques for the approximation of a desired trajectory by a trajectory that satisfies dynamic constraints and can thus be exactly tracked using an inversion of the system dynamics have been shown [17–19]. This non-minimum phase property limits the bandwidth of any controller, and prevents perfect tracking of dynamically feasible trajectories in the presence of disturbance. This greatly limits the control performance achievable for such systems.

2.1.6 Control of Dynamically Stable Mobile Robots

Feedback control is required to asymptotically drive a robot's states towards some time-varying reference trajectories by suitable actuation of its inputs. Depending on the application, these reference trajectories may be C^∞ smooth, or they may be discontinuous. This is to be achieved in the presence of measurement noise, external disturbance, unmodelled dynamics, trajectory dynamic infeasibility, and model parameter uncertainty. The degree of trajectory dynamic infeasibility that must be tolerated by the controller varies depending on the properties of the overarching trajectory planner; some planners may yield fully dynamically feasible trajectories, whereas a human user may demand step changes in reference. This control may have to be performed whilst enforcing constraints on wheel torques so as to avoid wheel slip, and whilst also enforcing constraints on system states to maintain safety and/or stability, such as limiting the lean angle of the pendulum body and translational velocities.

This control problem also varies depending on the type of state references that are to be tracked. In this thesis two distinct use cases are considered: the control of local frame body velocities, with the system's Cartesian positions states allowed to grow unboundedly, and the control of global inertial frame positions.

A local frame body velocity controller for a dynamically balancing mobile robot aims to iteratively generate input trajectories such that the system's local frame body velocities converge to a time varying reference trajectory, whilst bounding

deviations of the system's lean angle from the unstable equilibrium and safely handling actuator saturation. This is useful in applications where a human user is expected to generate the reference velocity trajectory, such as when used as a personal vehicle or teleoperated platform. The integration of local frame body velocity trajectories to inertial frame position trajectories is easily performed by the human user rather than the controller, providing a more intuitive input to the user and simplifying control design.

An inertial frame position controller instead aims to iteratively generate input trajectories to yield asymptotic tracking of a time-varying Cartesian position reference, allowing the robot to be commanded to move to a particular location in an obstacle-free environment. This must be performed while enforcing lean angle and actuator constraints as above, but must also enforce constraints on system velocities so as to generate a safe evolution toward the target position. This form of controller is required for autonomous navigation tasks, such as moving through a series of waypoints.

2.1.7 Trajectory Planning

Trajectory planning is the process of calculating system state and input trajectories that yield a desired transition from an initial to terminal state, potentially passing through a number of intermediate states. This allows entire complex state trajectories to be planned at once, as opposed to achieving such a transition using just closed-loop control, in which the resulting time evolution of the system states are instead defined by the dynamics of the closed-loop system. Planning entire trajectories at once allows the minimisation of some cost function along the entire trajectory, whilst potentially also enforcing constraints on system states and inputs. The planning of trajectories is typically a more computationally intensive process than the calculation of a controller feedback, meaning the resulting trajectories can only respond to large disturbances, so small deviations from a planned trajectory must be corrected by a low-level tracking controller instead.

For some systems the planning of trajectories can be broken down into two parts: planning a geometric path $q(s),\ s \in [0, 1]$ from an initial configuration $q(0)$ to a new configuration $q(1)$, and defining a smooth monotonic timing law $s(t),\ \dot{s}(t) \geq 0$ that describes the rate at which the system is to progress along the

path by the mapping of s to $t \in [0, T]$, $s : [0, T] \rightarrow [0, 1]$, see [10]. This splitting of trajectory planning allows velocity constraint satisfaction by uniformly slowing the trajectory through a linear scaling of the timing law. It is also possible to plan paths directly in the time domain to define trajectories $q(t)$, though this prevents the same linear time scaling of trajectories. For holonomic systems it is sufficient for $q(s)$ to possess bounded first order derivatives to represent dynamically feasible configuration trajectories, whereas for systems with nonholonomic kinematic constraints $q(s)$ must also have bounded second order derivatives to avoid instantaneous changes in direction. For systems with second-order nonholonomic constraints the resulting time domain trajectories $q(s(t))$ or $q(t)$ must satisfy these dynamic constraints in order to represent feasible configuration trajectories. Depending on the system this could require bounded derivatives up to much higher orders, tightly coupling the planning of the path and timing laws, meaning these are often derived simultaneously. These differing requirements necessitate the use of significantly different planning techniques depending on the kinematic and dynamic constraints acting on the system.

For some systems the mapping of Cartesian and heading trajectories to the dynamically feasible state space trajectories required for the exact tracking of a sufficiently smooth Cartesian path can be performed using the concept of differential flatness [20, 21]. This is a property exhibited by a number of common mobile systems, including holonomic vehicles [10] and quadrotors [22, 23]. A nonlinear input-affine system $\dot{x} = f(x) + G(x)u$ is differentially flat if there exists a set of outputs $y = h(x)$ such that all system states x and inputs u can be defined algebraically in terms of the flat outputs and their time derivatives up to a finite order r. For the systems listed above the Cartesian position and heading states are found to be flat outputs, allowing dynamically feasible state trajectories to be calculated by an algebraic function of trajectories in the Cartesian position and heading states, provided they are continuously differentiable up to the necessary order r. The required degree of smoothness r is entirely dependent on dynamic model structure, with $r = 2$ for holonomic systems and $r = 4$ for quadrotors.

For some systems this differential flatness property can also exist partially [24]. This technique partitions the system state vector as $x = \begin{bmatrix} x_s^T & x_r^T \end{bmatrix}^T$, in which the x_s states and input can be algebraically defined in terms of some flat outputs, allowing trajectories in these states to be algebraically derived from a sufficiently

smooth output trajectory. The remaining x_r state trajectories must then be derived by numerical integration of the system dynamics from initial conditions, though this integration now only need be performed on a lower dimensional system. Nonlinear trajectory optimisation techniques can then be used to optimise trajectories in the flat outputs to yield some optimal evolution of both the x_s and x_r states, potentially a less complex problem than optimising trajectories in the system's inputs. As the correct choice of output to yield partial or full differential flatness is not always obvious for some systems, methods for numerically searching for these outputs have been demonstrated [25].

The complexity of the path planning problem varies greatly depending on the inclusion or exclusion of obstacles, and the presence of any other state or input constraints. A number of methods exist for planning dynamically unconstrained paths in the presence of obstacles, such as RRT-Connect [26], A* [27], PRM [28], and artificial potential fields [29]. Methods also exist for the planning of trajectories in the presence of obstacles that are subject to kinematic and dynamic constraints, such as kinodynamic RRT* [30] and non-convex nonlinear optimisation techniques such as various shooting and transcription methods [31], all forming searches of higher-dimensional spaces. However, through this high-dimensionality and the potential presence of non-convex constraints due to obstacles, these problems can be hard to solve in real-time for real-world navigation tasks, and are strongly susceptible to local minima. The planning of dynamically feasible trajectories through 2D or 3D occupancy grids is therefore usually approached by splitting the problem into two parts: first finding the shortest continuous path to a goal through the occupancy grid using a 2D or 3D search, and then convexifying the obstacle constraint set along this path or sampling the path into an appropriate set of waypoints and performing a second optimisation to find dynamically feasible trajectories through this constraint tube or set of waypoints.

As the 2D path planning part of this process is relatively system agnostic, this thesis instead focuses on the latter part of this problem, the task of finding dynamically feasible state trajectories that pass through two or more waypoints in configuration space, whilst also satisfying state and input constraints.

2.2 Existing Solutions

Existing wheeled locomotion solutions can be grouped into four categories based on their manoeuvrability and stability: nonholonomic statically stable systems, holonomic systems, nonholonomic dynamically stable systems, and omnidirectional dynamically stable systems. These are described in this section, and a review is performed of methods commonly used in their control and trajectory planning.

2.2.1 Nonholonomic Statically Stable Systems

This class includes a variety of configurations of fixed and steered standard wheels that allow for motion subject to nonholonomic kinematic constraints, with the car being a classic example of this class. Research into the control of these systems is typically concerned with finding solutions to the parallel parking problem in mobile robotics, as again from Brockett's theorem a system with nonholonomic kinematic constraints cannot be stabilised to a given equilibrium by smooth time invariant feedback [2, 32]. Furthermore, as nonholonomic statically stable mobile robots are not subject to any constraints on their dynamics, the paths of their trajectories can be planned independently of the path timing law, allowing the separation and simplification of these tasks. The bulk of this research therefore has limited relevance to the system explored in this thesis, and that which is relevant is also applicable to omnidirectional statically stable systems; these relevant topics are therefore reviewed in the next section.

2.2.2 Omnidirectional Statically Stable Systems

Three or more omnidirectional wheels located on the vertices of a polygon can be used to achieve static stability and omnidirectional motion. A number of omnidirectional statically stable robots are commercially available, exampled in Figure 2.3, which typically either use a triangular arrangement of omniwheels with intersecting rotation axes, or a rectangular arrangement of Mecanum wheels with parallel rotation axes and alternating handedness. These systems have been well studied in the literature, with the main topics of interest being that of planning trajectories to navigate environments in the presence of obstacles, and the creation of controllers for the asymptotic tracking of these trajectories.

As through their inherent static stability these systems are not subject to any second-order nonholonomic constraints, and as through the use of omnidirectional wheels also do not possess kinematic constraints, these systems are said to be holonomic, reflecting the lack of nonholonomic constraints of any order acting on the system. In practice, holonomic systems are able to instantaneously generate sustained acceleration in any direction, yielding a maximally controllable system. Through this lack of kinematic constraints these systems can be asymptotically stabilised in configuration space by a linear feedback of the form $u = -Kx$, with suitable gain K, allowing for simple and high-bandwidth control.

Kinematic and dynamics modelling of holonomic systems is straightforward [10, 33–35], and various approaches to modelling and identifying friction effects in the omnidirectional wheels used in these systems have been demonstrated [36]. Han introduces a calibration method to account for dead reckoning error introduced by roller slippage, bearing play, and friction [37], showing a 50% improvement in dead reckoning accuracy during lateral translation. Purwin models and analyses the effects of weight transfer on traction during acceleration [38].

Trajectory planning for holonomic systems is an easier task than for systems with kinematic or dynamic constraints, as arbitrary continuous paths can be followed, requiring no bounded derivatives. As with kinematically nonholonomic systems, arbitrary smooth and monotonic timing laws may be chosen, again allowing the separation of these two planning tasks. This path planning is typically achieved by defining paths using low order splines [39, 40], upon which heuristic based timing laws can be defined to satisfy acceleration and velocity constraints. This has been extended to shape-aware planning around obstacles for robots of complex 2D geometries [41].

2.2.3 Dynamically Stable Systems with Nonholonomic Kinematic Constraints

The most common configuration of robot that dynamically balances whilst being subject to nonholonomic kinematic constraints is the two-wheeled inverted pendulum (TWIP). This configuration utilises two differential drive wheels attached to the robot body, with a center of mass located above the wheel axle, dynamically balancing about the wheel rotation axis. Just as in a statically stable differential drive robot, the full configuration space is accessed by combinations

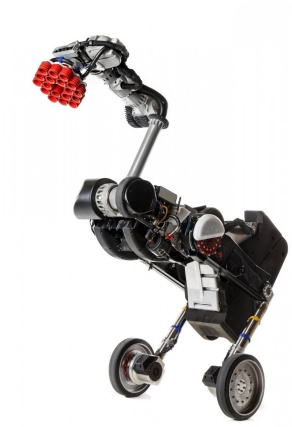
(a) A KUKA YouBot, an omnidirectional robot utilising four Mecanum wheels.

(b) Pepper [42], a small omnidirectional humanoid robot utilising three omniwheels.

Figure 2.3: Typical omnidirectional statically stable mobile robots, also referred to as holonomic robots.

of forward motion, controlled by manipulation of the system's lean angle, and rotation about a vertical axis running through the center of a line drawn between the two wheels. The TWIP is very well studied in the literature, and is closely related to the standard inverted pendulum, a classic testbed system for control development. A number of review articles [43, 44] and books [45] compare different modelling and control methods for both the regular and mobile inverted pendulum. Modelling is typically performed from first principles by the Newton-Euler [46–49], Euler-Lagrange [45, 50, 51], or Kane's methods [50, 52, 53], all of which yield equivalent models. A number of authors only consider one-dimensional models that ignore yaw dynamics, or include yaw dynamics but ignore cross-coupling effects by using two independent models. These modelling approaches are not suitable for the coupled system considered in this thesis, as these would likely fail to capture significant cross-coupled dynamics. Black-box model estimation approaches have also been applied [54], though without demonstrating any improvement in model accuracy over modelling from first principles. Controllability of TWIPs has been proven and their maximum relative degree has been derived [45, 51]. Partial feedback linearisation of the TWIP has been performed, with the TWIP's forward velocity and position states forming unstable

(a) EPFL's JOE, an early mobile inverted pendulum [46].



(b) Boston Dynamic's Handle, a two-wheeled inverted pendulum robot for package handling [67].

Figure 2.4: Notable two-wheeled mobile inverted pendulums

zero dynamics [45, 51, 55, 56].

Being a classic control test bed, a huge variety of control approaches have been applied to the TWIP, such as linear state space control [46, 57], sliding mode control [58], model predictive control [59–61], and a variety of model-free approaches [62–65]. Traction modelling for TWIPs has been studied, along with the development of a traction-aware model predictive controller [66].

As with statically stable nonholonomic systems, part of the related literature is concerned with the stabilisation of the TWIP to some Cartesian position and pose in the presence of nonholonomic kinematic constraints, and the planning of kinematically feasible paths. Again this is a substantially different task for these systems compared to that studied in this thesis, so this portion of the TWIP literature is of limited relevance and is not reviewed.

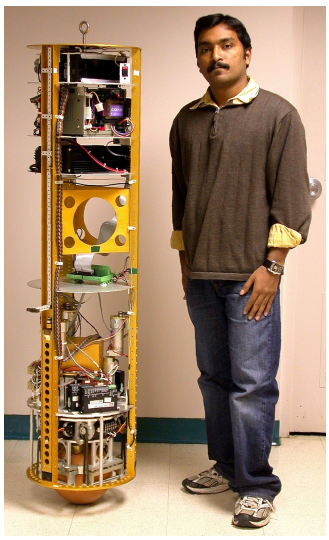
2.2.4 Omnidirectional Dynamically Stable Systems

Omnidirectional dynamically balancing systems currently rely on just two distinct wheel configurations: ball-drive actuators, and omniwheels with actuated rollers. Both of these systems balance in two dimensions simultaneously, and represent the most similar class of mobile robot to that explored in this thesis.

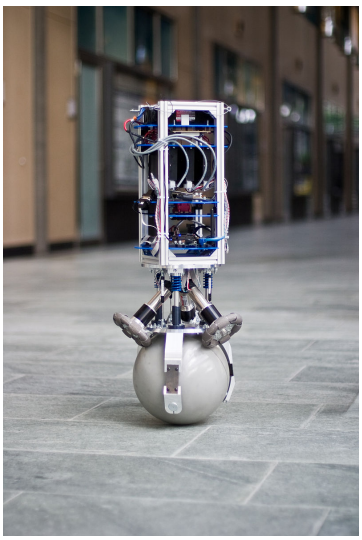
Ball-balancing robots are single-wheeled robots that dynamically balance on top of a ball [68]. Two notable ball-balancing research platforms exist in the control literature: ETH Zurich's Rezero [69], and Carnegie Mellon University's Ballbot [70], both shown in Figure 2.5. Multiple methods exist for actuation of the ball. The most elegant solution contacts the ball with three individually actuated omnidirectional wheels [69], allowing the robot to apply a net torque to the ball about any arbitrary axis. A similar drive mechanism has been created using rollers with linear slip [71], though no advantage over omniwheels is shown. Alternatively a mouse-ball configuration can be used, in which two horizontal rollers press on the sides of the ball to control its pitch and roll, with yaw control achieved by a revolute joint in the body [70]. Promising results have been shown by a spherical induction motor, achieving actuation of the ball using no actuated moving parts other than the ball [72]. By manipulation of the ball such a robot is able to control its attitude, which in turn allows it to control its translational acceleration. Such a robot is therefore able to navigate an environment whilst remaining balanced about the upright unstable equilibrium. However, in possessing only a single ground contact point the maximum yaw torque that can be applied to the ball is restricted to that which can be achieved without the ball twisting against the ground as static friction is overcome. This limits the ability of ball-balancing robots to interact with their environment or resist disturbance in this dimension. Despite a number of efforts no ball-balancing robot has ever been successfully commercialised, due to vibration issues caused by the use of omniwheels on the non-flat ball surface, and effects similar to static friction due to deformation of the ball under the roller and restraint contact points.

Decoupled planar models of a ball-balancing robot have been developed using the Newtonian [73] and Euler-Lagrange [69, 70, 74] methods, and fully coupled dynamics modelling of ball-balancing robots has been performed using the Newtonian [75, 76] and Euler-Lagrange [69, 77] methods. Full state estimation based on inertial and odometry data has been demonstrated using an extended Kalman Filter [78], achieving more than sufficient performance and dead reckoning accuracy for the evaluation of different control approaches.

Closed-loop trajectory tracking control of ball-balancing robots has been performed using gain scheduling to provide a nonlinear controller with a wide operating region [69]. Differential flatness [20, 21] and polynomial optimisation

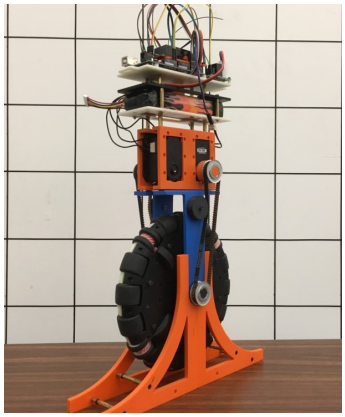


(a) CMU's Ballbot [70]



(b) ETH Zurich's Rezero [69]

Figure 2.5: Notable ball-balancing robots



(a) OmnUR robot from UCLA's RoMeLa [79]

(b) Honda's UNI-CUB personal mobility device [80]

Figure 2.6: Omnidirectional balancing systems utilising omniwheels with actuated rollers

techniques have been used to plan dynamically feasible trajectories through multiple fixed state waypoints [73, 81] using a decoupled 2D dynamics model. This optimisation can be performed very quickly, planning long trajectories in $< 1\,\text{ms}$, but does require some model simplification in order to achieve differential flatness, making planned trajectories less dynamically feasible with deviation from the vertical. This model simplification can be avoided by utilizing partial differential flatness, in which a sufficiently smooth parametrised trajectory in the fully differentially flat outputs is optimised to yield a desired trajectory in the non-differentially flat states [24]. This method is able to plan very aggressive trajectories that remain dynamically feasible for arbitrary lean angles, however, the trajectories of non-differentially flat states must still be determined by integration of the system dynamics from initial conditions, so this remains a slow and computationally expensive method. Nagarajan [74] plans trajectories in configuration space by describing trajectories in the lean angle states of the robot in each planar axis using a series of parametrised hyperbolic secant functions, with parameters optimised subject to the system's planar dynamic constraints. These trajectories are then tracked using a zero moment point (ZMP) technique and PID controller. Nagarajan [82] also generates dynamically feasible trajectories through state waypoints by performing a nonlinear optimisation subject to the nonholonomic dynamic constraint imposed by the system's underactuation to optimise a linear map between some sufficiently smooth desired trajectory and a subset of the system dynamics, performed using a full 3D coupled model. This optimisation takes in the region of 1 s for typical trajectories, and can therefore feasibly be used for some real-time planning applications. Pardo implements direct transcription to generate optimal trajectories in the presence of simple obstacles using nonlinear programming [83]. While producing optimal trajectories, this technique is too computationally expensive to be implemented in real-time for complex trajectories or multiple obstacles. Neunert [84] demonstrates effective unconstrained optimal model predictive control for planning trajectories through multiple temporally fixed waypoints. This is performed by minimising a quadratic input and state error cost function over a receding horizon by optimisation of a linear time-varying feedback and feedforward controller, using the full nonlinear coupled model for prediction. This optimisation can be performed very quickly, taking 20 ms to plan trajectories of 4 s duration, and with solve time

scaling linearly with trajectory length.

A handful of balancing robots utilising a single omniwheel with actively driven rollers have been demonstrated, exampled in Figure 2.6. These omniwheels require either complex mechanical design to transfer torque from a single central actuator to the rollers, or incorporate an individual actuator within every roller. By possessing a single contact point these systems suffer from the same lack of significant yaw control as ball-balancing robots. This is addressed in Figure 2.6b by the inclusion of a second omniwheel with passive rollers. Limited research has been undertaken into the control of and trajectory planning for this wheel configuration.

Chapter 3

The Collinear Mecanum Drive

This chapter introduces the Collinear Mecanum Drive (CMD), the locomotion system that is to be studied in this thesis. It begins with the derivation of the kinematic and dynamic models of the CMD, and then analyses the underactuation properties of the dynamics model, its controllability, and the size of the maximum feedback linearisable subsystem. An analysis of parameter selection is then performed to provide insight into the optimal CMD design.

The Collinear Mecanum Drive (CMD) utilizes three or more collinear Mecanum wheels to enable omnidirectional locomotion, whilst simultaneously dynamically balancing about the wheel rotation axis. By dynamically balancing, the ground footprint of a robot utilising a CMD can be designed to be arbitrarily thin, only lower bounded by choice of wheel diameter. The omnidirectional manoeuvrability of the CMD in combination with this narrow footprint allows for the navigation of much smaller gaps between obstacles than existing omnidirectional locomotion methods, achieved by translating directly along the common wheel rotation axis. This is in contrast to two-wheeled inverted pendulum systems, which would have to instead perform a parallel-parking style manoeuvre to achieve the same translation. Being a dynamically balanced system, the height of the center of mass of a robot driven by a CMD can be increased without requiring a proportional increase in the width of the ground footprint so as to avoid toppling during acceleration or disturbance, as would be the case for existing statically stable omnidirectional locomotion methods.

Omnidirectional dynamically balanced motion has previously only been demonstrated using ball-balancing robots, which possess a number of disadvantages

compared to the CMD. By only balancing about a single axis, as opposed to the two dimensional balancing required in a ball-balancing robot, the CMD retains greater control authority over the statically stable subsystem, and less energy is required to maintain balance. The CMD also retains significant control authority over its rotational dynamics about the vertical by virtue of multiple ground contact points, allowing for greater control performance and environment interaction.

This new locomotion system allows for the creation of omnidirectional systems of the same height as existing statically stable omnidirectional platforms, whilst requiring a fraction of the ground footprint and overall system size, and with a much smaller minimum navigable gap. This enables the creation of tall and slender robots that are better able to navigate cluttered environments such as those encountered in personal robotics in the home, retail, and office sectors. This is achieved with a fraction of the complexity of existing ball-balancing or legged solutions, requiring only three actuators and three moving parts[1], allowing for increased reliability and reduced unit cost.

Only a single example of a CMD existed in the literature prior to this research, in which a three wheeled CMD demonstrated the possibility of omnidirectional movement whilst simultaneously dynamically balancing [85]. However, this CMD demonstrates poor performance, taking 60 s to perform a 10 cm translation along its wheel axis, and no attempt is made to model or analyse the system dynamics.

3.1 Kinematic Model

In order to derive the inverse kinematics and dynamics models of the proposed platform, the nonholonomic constraints imposed by the Mecanum wheels must first be derived.

Consider the proposed CMD platform depicted in Figure 3.1 on a flat plane, where $\{\mathrm{E}, \hat{e}_x, \hat{e}_y, \hat{e}_z\}$ denotes the fixed inertial reference frame. The body attached frame $\{\mathrm{B}, \hat{b}_x, \hat{b}_y, \hat{b}_z\}$ is obtained by a rotation of E about $\hat{e_z}$ by ϕ, followed by a translation of $x\hat{e}_x + y\hat{e}_y$, with B located on the wheel rotation axis in the center of the platform. The pendulum attached frame $\{\mathrm{P}, \hat{p}_x, \hat{p}_y, \hat{p}_z\}$ is obtained by a translation of B by h_p along $\hat{b}_z$, followed by a rotation of θ_p about $\hat{b}_x$, where h_p

[1] This is excluding the unactuated Mecanum wheel rollers, as compared to a typical moving part within a robot these are very simple and low cost.

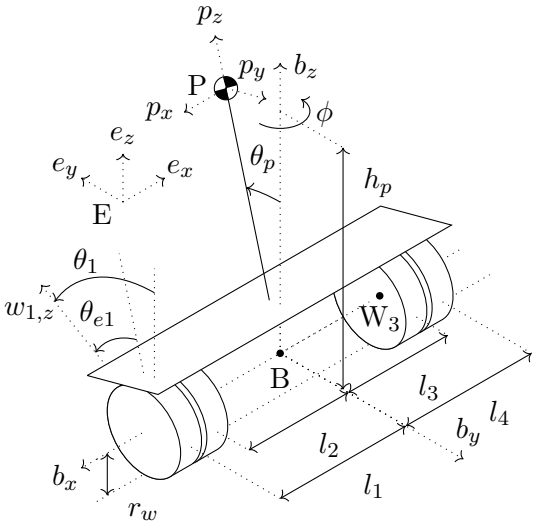


Figure 3.1: Collinear Mecanum Drive coordinates and parameters for the experimental prototype shown in Figure 4.3

represents the height of the pendulum center of mass along $\hat{b}_z$ relative to B, with associated mass m_p and inertia tensor $I_p = \texttt{diag}(\begin{bmatrix} I_{px} & I_{py} & I_{pz} \end{bmatrix})$. The i wheel coordinate frames $\{W_i, \hat{w}_{i,x}, \hat{w}_{i,y}, \hat{w}_{i,z}\}$ are obtained by a rotation of B about $\hat{b}_x$ by θ_i and a translation of $\hat{b}_x l_i$, and have identical masses m_w and identical inertia tensors $I_w = \texttt{diag}(\begin{bmatrix} I_{wx} & I_{wyz} & I_{wyz} \end{bmatrix})$ attached at W_i. Only one roller is considered per wheel, and it is assumed to always be positioned directly under the center of the wheel along the $\hat{w}_{i,z}$ axes, with the contact point between this and the ground assumed to be fixed under the center of the roller. This is a simplification, as in reality as a Mecanum wheel rotates this contact point transitions from one side of the roller to the other, before discontinuously jumping back to the start of the next roller as the next roller contacts the ground. Incorporating this phenomena yields a discontinuous model, greatly complicating simulation and model-based control design. The exact contact location is also sensitive to small variations in ground flatness, and is hard to exactly determine in a real-world system. For these reasons this simplification is justified, and is expected to manifest as a cyclic disturbance acting as a torque about $\hat{b}_z$ as each l_i varies over rotation of wheel i. The roller axis of rotation $\hat{r}_i$ is defined as a rotation of $\hat{b}_x$ by α_i about $\hat{b}_z$ where $\sin(\alpha_i) \neq 0$ and $\cos(\alpha_i) \neq 0$, with roller angular position given as a rotation about $\hat{r}_i$ by Ω_i. Due to their small size the rollers are assumed to

be massless and inertialess for sake of model simplicity.

Considering a single Mecanum wheel, let $\hat{\mu}_p$ represent the unit vector running parallel to $\hat{r}_i$ through the ground contact point, expressed in the local body attached frame, let W represent the wheel's centre, and let the roller contact the ground directly under W at C as $C = W - r_w\hat{b}_z$, where r_w denotes the wheel radius measured to the roller contact point and perpendicular to the wheel rotation axis.

For no slip to occur, the component of the roller's velocity at the contact point along $\hat{\mu}_p$ must always be zero, so

$$\vec{v}_{EC,B} \cdot \hat{\mu}_p = 0 \tag{3.1.1}$$

in which $\vec{v}_{EC,B}$ represents the velocity of C relative to E expressed in the local body frame B, and where $\cdot$ denotes the dot product.

$\vec{v}_{EC,B}$ can be expressed as the body frame velocity of the wheel at W relative to E summed with the tangental velocity due to wheel angular velocity $\dot{\theta}_i$ as

$$\vec{v}_{EC,B} = \vec{v}_{EW,B} - r_w\hat{b}_y\dot{\theta}_i \tag{3.1.2}$$

Similarly, $\vec{v}_{EW,B}$ can be defined in terms of the body frame velocity of B relative to E as

$$\vec{v}_{EW,B} = \vec{v}_{EB,B} + \dot{\phi}l_i\hat{b}_y \tag{3.1.3}$$

Finally, $\vec{v}_{EB,B}$ can be expressed in the inertial frame as

$$\vec{v}_{EB,B} = R_{EB}^T\vec{v}_{EB,E} \tag{3.1.4}$$

Combining (3.1.1)-(3.1.4) and splitting $\vec{v}_{EB,E}$ into its components along $\hat{e}_x$ and $\hat{e}_y$, denoted x and y, yields the nonholonomic no-slip constraint

$$\dot{x}\cos(\alpha_i - \phi) - \dot{y}\sin(\alpha_i - \phi) - \dot{\phi}l_i\sin(\alpha_i) + \dot{\theta}_i r_w\sin(\alpha_i) = 0 \tag{3.1.5}$$

Similarly, the angular velocity of the roller $\dot{\Omega}_i$ is proportional to its velocity along the vector $\hat{\mu}_t$, where $\hat{\mu}_t$ is perpendicular to $\hat{\mu}_p$ and parallel to the ground, so

$$\vec{v}_{EC} \cdot \hat{\mu}_t = r_r\dot{\Omega}_i \tag{3.1.6}$$

which by substitution with (3.1.2)-(3.1.3) yields the nonholonomic rolling constraint

$$\dot{x}\sin(\alpha_i - \phi) + \dot{y}\cos(\alpha_i - \phi) + \dot{\phi} l_i \cos(\alpha_i) - \dot{\theta}_i r_w \cos(\alpha_i) = \dot{\Omega}_i r_r \tag{3.1.7}$$

Equation (3.1.5) can be applied to wheels 1 through n_w and rewritten in matrix form to define the platform's inverse kinematic mapping $f^{-1} : (\dot{x}, \dot{y}, \dot{\phi}) \rightarrow \dot{\theta}_i$

$$\dot{\theta}_i = \begin{bmatrix} \frac{-\cos(\alpha_i - \phi)}{r_w \sin(\alpha_i)} & \frac{\sin(\alpha_i - \phi)}{r_w \sin(\alpha_i)} & \frac{l_i}{r_w} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} \qquad \text{for } i = [1 \,..\, n_w] \tag{3.1.8}$$

Remark 1. *Minimum wheel quantity*

As the row vector on the left of (3.1.8) *is clearly of rank 1 and dimension 3, a minimum of three wheels, with* (α_i, l_i) *chosen so that the rows of the matrix composed by stacking the row vectors in* (3.1.8) *are independent, are required to create a unique forward kinematic mapping* $f : \dot{\theta} \rightarrow (\dot{x}, \dot{y}, \dot{\phi})$ *where* $\theta = \begin{bmatrix} \theta_1 & .. & \theta_{n_w} \end{bmatrix}^T$, $n_w \geq 3$.

3.2 Dynamics Model

Here the generalised CMD dynamics model is derived using the Lagrangian method, chosen for its systematic incorporation of nonholonomic constraints and straightforward derivation. This is derived both in terms of the generalised coordinates and their derivatives, and in terms of generalised positions and local body frame velocities, a more useful formulation for control development later in this thesis.

There exist two methods of deriving a dynamics model subject to these nonholonomic constraints using the Lagrangian method; Lagrange multipliers can be used to directly incorporate the nonholonomic constraints, or the constraints can be approximately 'holonomised' using the psuedo-inverse of the inverse kinematic transformation matrix. Zimmerman showed both methods to be equivalent in the context of Mecanum wheeled vehicles [86]. Here the former approach is taken.

The system's dynamics equations are derived by use of the Euler-Lagrange equation in terms of the Lagrangian $\mathcal{L}(q, \dot{q})$, generalised coordinates q, generalised

forces Q, Lagrange multipliers λ, and Pfaffian constraint matrix $A^T(q)$, defined as

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = Q + A^T(q)\lambda \tag{3.2.1}$$

where $A(q)$ follows the Pfaffian constraint form $A^T(q)\dot{q} = 0$.

The generalised coordinates q are selected as

$$q = \begin{bmatrix} x & y & \phi & \theta_p & \theta_1 & \dots & \theta_{n_w} & \Omega_1 & \dots & \Omega_{n_w} \end{bmatrix}^T \tag{3.2.2}$$

and from (3.1.5) and (3.1.7) $A^T(q)$ is defined as

$$A^T(q) = \begin{bmatrix} \cos(\alpha_1 - \phi) & -\sin(\alpha_1 - \phi) & -l_1 \sin(\alpha_1) & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \cos(\alpha_{n_w} - \phi) & -\sin(\alpha_{n_w} - \phi) & -l_{n_w} \sin(\alpha_{n_w}) & 0 \\ \sin(\alpha_1 - \phi) & \cos(\alpha_1 - \phi) & l_1 \sin(\alpha_1) & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \sin(\alpha_{n_w} - \phi) & \cos(\alpha_{n_w} - \phi) & l_{n_w} \sin(\alpha_{n_w}) & 0 \end{bmatrix}$$
$$\begin{bmatrix} r_w \sin(\alpha_1) & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & r_w \sin(\alpha_{n_w}) & 0 & \dots & 0 \\ -r_w \cos(\alpha_1) & \dots & 0 & r_r & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -r_w \cos(\alpha_{n_w}) & 0 & \dots & r_r \end{bmatrix} \tag{3.2.3}$$

The Lagrangian $\mathcal{L}(q, \dot{q})$ is found as the difference of kinetic and potential energy in the system $\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q)$, where $\mathcal{K}(q, \dot{q})$ represents the sum of translational and rotational kinetic energy, and $\mathcal{U}(q)$ the total potential energy. The rotational kinetic energy of the system is defined as the sum of rotational energy of the pendulum mass and four wheel masses. As wheel torques act about the $\hat{b}_x$ axis, pendulum inertia I_p must be redefined about $\mathrm{P} - h_p \hat{p}_z$ as $I_{p,b}$, achieved using the parallel axis theorem with translation vector $r_p = \begin{bmatrix} 0 & 0 & -h_p \end{bmatrix}^T$ as

$$I_{p,b} = I_p + m_p \left[(r_p \cdot r_p) I_{3\times 3} - r_p \otimes r_p \right] \tag{3.2.4}$$

where $\otimes$ denotes the outer product. The wheel rotation axes $\hat{w}_{x,i}$ are already aligned with τ_i, so I_w remains unchanged.

This allows rotational kinetic energy K_r to be defined as

$$\mathcal{K}_r(\dot{q}) = \frac{1}{2}\vec{\omega}_p^T I_{p,b}\vec{\omega}_p + \frac{1}{2}\sum_{i=1}^{n_w}\vec{\omega}_{w_i}^T I_w \vec{\omega}_{w_i} \tag{3.2.5}$$

where

$$\vec{\omega}_b = \dot{\phi}\hat{b}_z \tag{3.2.6}$$

$$\vec{\omega}_p = R_{bp}^T\vec{\omega}_b + \dot{\theta}_p\hat{p}_x \tag{3.2.7}$$

$$\vec{\omega}_{w,i} = R_{bw_i}^T\vec{\omega}_b + \dot{\theta}_i\hat{w}_x \tag{3.2.8}$$

Similarly, translational kinetic energy is defined as the sum of that of the pendulum and four wheel masses as

$$\mathcal{K}_t(\dot{q}) = \frac{1}{2}\vec{v}_p^T m_p \vec{v}_p + \frac{1}{2}m_w\sum_{i=1}^{n_w}\vec{v}_{w,i}^T\vec{v}_{w,i} \tag{3.2.9}$$

where

$$\vec{v}_p = R_{bp}^T R_{eb}^T\begin{bmatrix}\dot{x} & \dot{y} & 0\end{bmatrix}^T + \vec{\omega}_p \times h\hat{p}_z \tag{3.2.10}$$

$$\vec{v}_{w,i} = R_{bw_i}^T R_{eb}^T\begin{bmatrix}\dot{x} & \dot{y} & 0\end{bmatrix}^T + \vec{\omega}_{w_i} \times l_i\hat{w}_x \tag{3.2.11}$$

Finally, potential energy is purely that due to the action of gravity on the pendulum body, defined as

$$\mathcal{U} = m_p g h_p \cos(\theta_p) \tag{3.2.12}$$

The generalised forces Q capture all non-conservative forces acting on the system, which here are motor torques Q_τ and both rolling and viscous friction forces Q_f for $Q = Q_\tau + Q_f$. The n_w motor drive torques $\tau = \begin{bmatrix}\tau_1 & \dots & \tau_{n_w}\end{bmatrix}^T$ act individually on each wheel, with each also producing an opposing counter-torque on the pendulum body. No motor torques act directly on the x, y, ϕ, or Ω_i generalised coordinates; the interactions between these and the motor torques are instead captured by the nonholonomic constraints (3.1.5) and (3.1.7). This

defines Q_τ as

$$Q_\tau = \begin{bmatrix} 0_{3\times 1} \\ \sum_{i=1}^{n_w}(-\tau_i) \\ \tau \\ 0_{n_w\times 1} \end{bmatrix} \tag{3.2.13}$$

Viscous friction is modelled at two interfaces for this system: at the body-to-wheel revolute joints, with coefficient k_{vw}, and at the wheel-to-roller revolute joints, with coefficient k_{vr}. It is assumed that k_{vw} is also able to approximate the various motor phenomena that sum to yield a non-zero no-load current. Linear rolling friction is modelled at the roller-to-ground interface as a torque about $\hat{w}_x$ proportional to wheel angular velocity $\dot{\theta}_i$, with coefficient k_{rw}. While there will also exist a rolling friction force acting along $\hat{b}_x$, there does not exist a simple experimental approach to allow the independent measurement of this coefficient and k_{vr}, so it is assumed that this can be sufficiently captured by the existing k_{vr} coefficient. Tractive friction forces between the roller and ground are already assumed to be infinite in the definition of the nonholonomic constraints in (3.1.5) and (3.1.7). It is assumed that kinetic friction in the wheel bearings can be fully compensated by application of a discontinuous torque offset to the wheel actuators, allowing its exclusion from the dynamics model, and it is assumed that static friction is negligible for model simplicity. Kinetic friction in the roller bearings cannot be compensated in such a manner, and cannot easily be modelled without introducing a discontinuity, so is therefore treated as a external disturbance. Again, static friction in this interface is also assumed to be negligible for model simplicity.

Viscous friction in the wheel-to body-revolute joint acts proportionally to the difference between each wheel angular velocity $\dot{\theta}_i$ and the pendulum's angular velocity $\dot{\theta}_p$, applying a torque of $k_{vw}(\dot{\theta}_p - \dot{\theta}_i)$ to each θ_i generalised coordinate and a torque of $\sum_{i=1}^{n_w}(k_{vw}(\dot{\theta}_i - \dot{\theta}_p))$ to the pendulum body θ_p. Viscous friction in the wheel-to-roller revolute joint acts proportionally to $-\dot{\Omega}_i$, applying a torque of $-k_{vr}\dot{\Omega}_i$ to each Ω_i generalised coordinate. The counter-torque from this friction force acts about two axes on the wheel. That about the $\hat{w}_x$ axis acts to rotate the wheel, applying a torque of $k_{vr}\hat{b}_x \cdot R_{br}\begin{bmatrix}\dot{\Omega}_i & 0 & 0\end{bmatrix}^T$ to each of θ_i. That orthogonal to $\hat{w}_x$ and parallel to the ground imparts an axial load on the wheel,

which is transmitted through the wheel mounting to directly apply a force on the pendulum body along the $\hat{b}_x$ axis. This is equivalent to a force acting on the $[x \;\; y]^T$ generalised coordinates of

$$\begin{bmatrix} I_{2\times 2} & 0_{2\times 1} \end{bmatrix} R_{eb} \begin{bmatrix} \frac{1}{-r_w+r_r} \\ 0 \\ 0 \end{bmatrix} \left(\hat{b}_y \cdot \sum_{i=1}^{n_w} R_{br,i} \begin{bmatrix} k_{vr}\dot{\Omega}_i \\ 0 \\ 0 \end{bmatrix} \right) \tag{3.2.14}$$

Rolling friction acting about b_x is proportional to wheel angular velocity $\dot{\theta}_i$.

This defines Q_v as

$$Q_f = \begin{bmatrix} \begin{bmatrix} I_{2\times 2} & 0_{2\times 1} \end{bmatrix} R_{eb} \begin{bmatrix} \frac{1}{-r_w+r_r} \\ 0 \\ 0 \end{bmatrix} \left(\hat{b}_y \cdot \sum_{i=1}^{n_w} R_{br,i} \begin{bmatrix} k_{vr}\dot{\Omega}_i \\ 0 \\ 0 \end{bmatrix} \right) \\ 0 \\ \sum_{i=1}^{n_w} (k_{vw}(\dot{\theta}_i - \dot{\theta}_p)) \\ -\dot{\theta}_1 k_{rw} + k_{vw}(\theta_p - \dot{\theta}_1) + k_{vr}\hat{b}_x \cdot R_{wr} \begin{bmatrix} \dot{\Omega}_1 & 0 & 0 \end{bmatrix}^T \\ \vdots \\ -\dot{\theta}_{n_w} k_{rw} + k_{vw}(\theta_p - \dot{\theta}_{n_w}) + k_{vr}\hat{b}_x \cdot R_{wr} \begin{bmatrix} \dot{\Omega}_{n_w} & 0 & 0 \end{bmatrix}^T \\ -k_{vr}\dot{\Omega}_1 \\ \vdots \\ -k_{vr}\dot{\Omega}_{n_w} \end{bmatrix} \tag{3.2.15}$$

Introducing $2n_w$ Lagrange multipliers $\lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_{2n_w} \end{bmatrix}^T$ allows the solution of (3.2), giving a system of $4 + 2n_w$ ODEs. These can be arranged into the matrix form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = A^T(q)\lambda + F\dot{q} + B\tau \tag{3.2.16}$$

with symmetric positive semidefinite[2] inertia matrix $M(q)$, Coriolis and centripetal matrix $C(q,\dot{q})$, derived using the Christoffel symbols of $M(q)$ such that

[2]Usually for Lagrangian systems $M(q)$ is positive definite, however, in choosing to model the wheel roller as being massless and inertialess eigenvalues of zero are introduced into $M(q)$.

$\dot{M}(q) - 2C(q,\dot{q})$ is skew symmetric as

$$c_{i,j} = \frac{1}{2}\sum_{k=1}^{4+2n_w}\left(\frac{\partial M_{ij}(q)}{\partial \dot{q}_k} + \frac{\partial M_{ik}(q)}{\partial \dot{q}_j} + \frac{\partial M_{jk}(q)}{\partial \dot{q}_i}\right)\dot{q}_k \tag{3.2.17}$$

and with gravity matrix $G(q)$, viscous and rolling friction matrix F, and input matrix B. These matrices are defined in full in Appendix A.1.

Provided the conditions set out in Remark 1 are met, examining $\text{rank}(A) = 2n_w$ indicates that $2n_w$ of the model's $4+2n_w$ degrees of freedom are fully constrained by A, meaning $2n_w$ generalised coordinates can be made redundant by elimination of the Lagrange multipliers. Defining the nullspace of A as Φ, such that $A\Phi = 0$ and therefore $\Phi^T A^T = 0$, it is evident that λ can be eliminated from (3.2.16) by premultiplication with Φ^T.

As the choice of Φ must satisfy $A^T\dot{q} = 0$, there exists a minimal vector of generalised velocities $\dot{p}$ that map back to $\dot{q}$ as $\dot{q} = \Phi\dot{p}$. As there are infinite solutions for Φ and therefore choices of $\dot{p}$, it is possible to choose Φ such that the rows mapping $\dot{p}$ to $\begin{bmatrix}\dot{x} & \dot{y} & \dot{\phi} & \dot{\theta}_p\end{bmatrix}^T$ in $\dot{q}$ form the rows of an identity matrix, allowing Φ to provide a mapping from the full generalised coordinates vector $\dot{q}$ to a convenient minimal generalised coordinate vector $\dot{p} = \begin{bmatrix}\dot{x} & \dot{y} & \dot{\phi} & \dot{\theta}_p\end{bmatrix}^T$, giving Φ as

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-\cos(\alpha_1+\phi)}{r_w\sin(\alpha_1)} & \frac{-\sin(\alpha_1+\phi)}{r_w\sin(\alpha_1)} & -\frac{l_1}{r_w} & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-\cos(\alpha_{n_w}+\phi)}{r_w\sin(\alpha_{n_w})} & \frac{-\sin(\alpha_{n_w}+\phi)}{r_w\sin(\alpha_{n_w})} & -\frac{l_{n_w}}{r_w} & 0 \\ \frac{2\cos(\phi)\cos(\alpha_1)}{r_r} & \frac{2\sin(\phi)\cos(\alpha_1)}{r_r} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{2\cos(\phi)\cos(\alpha_{n_w})}{r_r} & \frac{2\sin(\phi)\cos(\alpha_{n_w})}{r_r} & 0 & 0 \end{bmatrix} \tag{3.2.18}$$

Multiplying by Φ^T, substituting $\dot{q} = \Phi\dot{p}$ and $\ddot{q} = \Phi\ddot{p} + \dot{\Phi}\dot{p}$, and recognising that all matrices that depend on q in (3.2.16) depend only upon elements that

also exist within p, allows (3.2.16) to be rewritten in the reduced generalised coordinates p as

$$\Phi^T M(q)\Phi\ddot{p} + \Phi^T M(q)\dot{\Phi}\dot{p} + \Phi^T C(q,\dot{q})\Phi\dot{p} + \Phi^T G(q) = \Phi^T F\Phi\dot{p} + \Phi^T B\tau_i \quad (3.2.19)$$

which can be written as the new set of model matrices

$$M_p(p) = \Phi^T M(q)\Phi \quad (3.2.20)$$

$$C_p(p,\dot{p}) = \Phi^T M(q)\dot{\Phi} + \Phi^T C(q,\dot{q})\Phi \quad (3.2.21)$$

$$G_p(p) = \Phi^T G(q) \quad (3.2.22)$$

$$F_p(p) = \Phi^T F\Phi \quad (3.2.23)$$

$$B_p(p) = \Phi^T B \quad (3.2.24)$$

in which $M_p(p)$ is now both symmetric and positive definite, and $\dot{M}_p(p) - 2C_p(p,\dot{p})$ remains skew symmetric. These matrices are defined in full in Appendix A.2.

As $\det(M_p(p)) \neq 0 \; \forall \; p \in \mathbb{R}^4$ for sensical parameter choices $M_p(p)$ is invertible, allowing (3.2.19) to be solved for $\ddot{p}$ as

$$\ddot{p} = M_p(p)^{-1}(F_p(p)\dot{p} + B_p(p)\tau - C_p(p,\dot{p})\dot{p} - G_p(p)) \quad (3.2.25)$$

thus allowing integration of the system dynamics over time from an initial state $(p_0, \dot{p}_0)$ and with input trajectory $u(t)$.

As $\text{rank}(B_p) < \dim(\tau)$ when $n_w > 3$, as in the experimental prototype in this thesis, the input τ does not represent a linearly independent set of inputs. This means there exists a linear map $\Lambda : \tau \to u$ that maps τ onto a new minimal set of independent inputs u, written in matrix form as $u = \Lambda\tau$, in which there exist infinite choices for u. Separating $B_p(p)$ into its nonlinear and linear components

$$B_p(p) = B_{p,nl}(p)B_{p,l} \quad (3.2.26)$$

$$= \begin{bmatrix} R_{eb}(p) & 0_{2\times 2} \\ 0_{2\times 2} & I_{2\times 2} \end{bmatrix} \begin{bmatrix} -\frac{\cot\alpha_1}{r_w} & -\frac{\cot\alpha_2}{r_w} & \dots & -\frac{\cot\alpha_{n_w}}{r_w} \\ -\frac{1}{r_w} & -\frac{1}{r_w} & \dots & -\frac{1}{r_w} \\ -\frac{l_1}{r_w} & -\frac{l_2}{r_w} & \dots & -\frac{l_{n_w}}{r_w} \\ -1 & -1 & \dots & -1 \end{bmatrix} \quad (3.2.27)$$

and defining $\hat{B}_{p,l}$ as a basis for the column space of $B_{p,l}$, one suitable map can

be found as $\Lambda = \hat{B}_{p,l}^{+} B_{p,l}$, where

$$\hat{B}_{p,l} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & r_w & 0 \end{bmatrix} \quad \Lambda = -\tfrac{1}{r_w} \begin{bmatrix} \cot\alpha_1 & \cot\alpha_2 & \dots & \cot\alpha_{n_w} \\ 1 & 1 & \dots & 1 \\ l_1 & l_2 & \dots & l_{n_w} \end{bmatrix} \tag{3.2.28}$$

giving the overall input matrix $B_p(p) = B_{p,nl}(p)\hat{B}_{p,l}\Lambda\tau$. Replacing $B_p(p)$ in 3.2.19 with $\hat{B}_p(p) = B_{p,nl}(p)\hat{B}_{p,l}$ and using $u = \Lambda\tau$ as the new input yields the new system

$$M_p(p)\ddot{p} + C_p(p,\dot{p})\dot{p} + G_p(p) = F_p(p)\dot{p} + \hat{B}_p(p)u \tag{3.2.29}$$

in which $\dim(u) = \operatorname{rank}(\hat{B}_p)$. Intuitively, the elements of this new input represent force on the body parallel to $\hat{e}_x$, force on the body parallel to $\hat{e}_y$, and torque on the body about $\hat{e}_z$.

A known input u can be mapped back to a choice of τ in which $\sum_{i=1}^{n_w} \tau_i^2$ is minimised as $\tau = \Lambda^{+}u$. If wheel torques are to be constrained this can be enforced by the solution of the constrained least-squares minimisation

$$\min_{\tau} \|\tau\|_2^2 \quad s.t. \quad \Lambda\tau = u, \quad -\overline{\tau} \le \tau_i \le \overline{\tau} \;\forall\; i \in [1 \,..\, n_w] \tag{3.2.30}$$

solvable as a quadratic program for feasible combinations of u and $\overline{\tau}$.

In some cases it is more useful to express (3.2.19) in terms of inertial frame positions, but local body frame velocities and accelerations. This can be achieved by defining (3.2.18) such that the first two rows of Φ equal

$$\Phi_{1:2} = \begin{bmatrix} R_{eb} & 0_{2\times 2} \end{bmatrix} \tag{3.2.31}$$

to give the new set of velocity states $v = \Phi_v^{-1}\dot{q}$, where $v = \begin{bmatrix} v_x & v_y & \dot{\phi} & \dot{\theta}_p \end{bmatrix}^T$. This therefore also defines the kinematic model

$$\dot{p} = \Psi v, \qquad \Psi = \begin{bmatrix} R_{eb} & 0_{2\times 2} \\ 0_{2\times 2} & I_{2\times 2} \end{bmatrix} \tag{3.2.32}$$

which can be differentiated to allow the substitution

$$\ddot{p} = \Psi\dot{v} + \dot{\Psi}v \tag{3.2.33}$$

This allows the definition of the two models

$$M_v(p)\dot{v} + C_v(p,v)v + G_v(p) = F_v v + B_v \tau \tag{3.2.34}$$

and

$$M_v(p)\dot{v} + C_v(p,v)v + G_v(p) = F_v v + \hat{B}_v u \tag{3.2.35}$$

in which again $M_v(p)$ is symmetric positive definite, $\dot{M}_v(p) - 2C_v(p,v)$ is skew symmetric, and matrices F_v, B_v, and $\hat{B}_v$ no longer depend on p. Again, $\det(M_p) \neq 0 \; \forall \; p \in \mathbb{R}^4$ for sensical parameter choices. These matrices are defined in full in Appendix A.3.

As in (3.2.25), the model (3.2.34) can be solved for $\dot{v}$ as

$$\dot{v} = M_v(p)^{-1} \left(F_v v + B_v \tau - C_v(p,v)v - G_v(p) \right) \tag{3.2.36}$$

It is this model which is to be used for simulation of CMD throughout the rest of this thesis, with numerically integration performed using MATLAB's `ode45`, an explicit Runge-Kutta (4,5) solver [87], with absolute and relative tolerances of 1×10^{-9}.

3.2.1 Underactuation of Dynamics

As $m = \text{rank}(\hat{B}_v) = 3$, $n = \dim(p) = 4$, $m < n$, the system is underactuated, meaning there are more configuration variables than independent control inputs and therefore no general inverse of $\hat{B}_p$ or $\hat{B}_v$ exists. From [16] underactuated systems can be analysed by splitting their configurations into *external variables* and *shape variables*, where shape variables p_s are defined as those appearing in the inertia matrix $M(p)$, and external variables p_x are defined as those not appearing in $M(p)$. This means that external variables do not appear in the kinetic energy expression $K(p,v) = \frac{1}{2} v^T M(p) v$, so $\frac{\partial K(p,v)}{\partial p_x} = 0$. If this is the case the Lagrangian is said to possess kinetic symmetry w.r.t. the external variables. From a seminal work by Saber [16] underactuated systems can be classified based the properties of the shape variables. The system (3.2.29) possesses the following properties:

1. The shape and external variables are $p_s = (\phi, \theta_p)$ and $p_x = (x, y)$ respectively.

2. Both the shape and external variables are actuated, in that all rows of B are nonzero, meaning the shape and external variables are coupled.

3. Partitioning $M_p(p)$ into shape and external variables as

$$M_p(p) = \begin{bmatrix} m_{xx}(p) & m_{xs}(p) \\ m_{sx}(p) & m_{ss}(p) \end{bmatrix} \tag{3.2.37}$$

the normalised momentums conjugate to p_x take the form

$$\pi_x = \dot{p}_x + m_{xx}^{-1}(p_s) m_{xs}(p_s) \dot{p}_s \tag{3.2.38}$$

$$= \begin{bmatrix} \dot{x} - a\cos(\phi)\dot{\phi} + b\left(\cos(\phi)\sin(\theta_p)\dot{\phi} + \cos(\theta_p)\sin(\phi)\dot{\theta}_p\right) \\ \dot{y} - a\sin(\phi)\dot{\phi} + b\left(\sin(\phi)\sin(\theta_p)\dot{\phi} - \cos(\theta_p)\cos(\phi)\dot{\theta}_p\right) \end{bmatrix} \tag{3.2.39}$$

with known positive constants a and b, which clearly by being globally smooth are integrable functions.

However, the normalised momentums conjugate to p_s

$$\pi_s = \dot{p}_x + m_{sx}^{-1}(p_s) m_{ss}(p_s) \dot{p}_s \tag{3.2.40}$$

which for brevities sake are not shown, are not integrable due to the existence of singularities, that is $\nexists\, h(p_x, p_s) : \dot{h} = \pi_s$.

4. There are equal numbers of shape and external variables, so $\dim(p_s) = \dim(p_x)$.

5. As $\text{rank}(m_{xs}(p)) = n - m \;\forall\; \{p \in \mathbb{R} \mid \theta_p \neq \pi/2\}$ the system is said to be strongly locally inertially coupled [12].

Surprisingly, these properties make this system unclassified by the system set out by Saber [16], as this author states never to have found a system with this combination of having actuated shape variables, coupled inputs, and nonintegrable momentums, therefore this combination is not considered nor analysed. This means the literature at present provides no expected normal form for which it should be possible for the system to be rearranged into.

3.2.2 Controllability

The controllability of a system describes its ability to move from any initial point in its state space $x_0 \in \mathbb{R}^n$ to any other point $x_T \in \mathbb{R}^n$ within finite time $T < \infty$ by manipulation of its inputs $u \in \mathbb{R}^m$. The global controllability of linear systems in the form $\dot{x} = Ax + Bu,\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m$ is easily proven by determining if the Kalman controllability matrix C_o is of full rank, i.e. $\text{rank}(C_o) = n$, where

$$C_o = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \tag{3.2.41}$$

Linear systems satisfying this condition can always be globally stabilised to the origin by a feedback of the form $u = -Kx$.

Such a proof does not exist for nonlinear systems. A weaker form of this proof is to instead show that a nonlinear system is small-time locally controllable (STLC), and a further weaker form is to show that a nonlinear system is small-time locally accessible (STLA).

Letting W represent an infinitely small region in state space centered around x_0, i.e. the local neighbourhood of x_0, define $\mathcal{R}^W$ as the set of configurations x_T that can be achieved by manipulation of u in an infinitely small time T without leaving W. A STLC system will be able to use sequences of control input to affect change in x_0 in all directions in W, meaning x_0 will be an interior point within $\mathcal{R}_W$, $p_0 \in \text{int}(\mathcal{R}^W)$, and therefore $\mathcal{R}^W = W$.

A STLA system, whilst still able to locally access a space with the same dimension as W, is restricted to accessing a subset $\mathcal{R}_W \subset W$, in which p_0 is on the boundary of $\mathcal{R}_W$ and so $p_0 \notin \text{int}(\mathcal{R}^W)$ [88].

Theorem 1. *The CMD is STLC from its equilibrium states for sensical model parameters.*

The set of equilibrium states $\mathcal{X}_e$ is defined as the set of states x with constant input $u = 0_{m\times 1}$ at which $\dot{x} = 0_{n\times 1}$, given for the state space $x = \begin{bmatrix} p & v \end{bmatrix}^T$ as

$$\begin{aligned} \mathcal{X}_e &= \{x \mid \dot{x} = f(x,u) = 0, \quad u = 0\} \\ &= \Big\{ \begin{bmatrix} x_1 & x_2 & \dots & x_8 \end{bmatrix}^T \mid (x_1, x_2, x_3) \in \mathbb{R}^3, \\ &\qquad x_4 = \pi k, \quad k \in \mathbb{Z}, \quad x_j = 0,\ j = [5\,..\,8] \Big\} \end{aligned} \tag{3.2.42}$$

As $0_{n\times 1} \in \mathcal{X}_e$, (3.2.35) can be linearised about the stationary upright equilibrium at the origin, yielding a system $\dot{x} = Ax + Bu$, where A and B take the form

$$A = \begin{bmatrix} & 0_{4\times 4} & & & & I_{4\times 4} & & \\ 0 & 0 & 0 & 0 & a & 0 & 0 & 0 \\ 0 & 0 & 0 & b & 0 & c & 0 & d \\ 0 & 0 & 0 & 0 & 0 & 0 & e & 0 \\ 0 & 0 & 0 & f & 0 & g & 0 & h \end{bmatrix} \quad B = \begin{bmatrix} & 0_{4\times 3} & \\ i & 0 & 0 \\ 0 & j & 0 \\ 0 & 0 & k \\ 0 & l & 0 \end{bmatrix} \tag{3.2.43}$$

in which A possesses some positive eigenvalues, meaning the upright equilibrium is unstable. Likewise, linearising about any $x \in \{\mathcal{X}_e : x_4 = \pi\}$ yields a negative semidefinite A, meaning the lowest pendulum position is a stable equilibrium, as would be expected.

Examining the Kalman controllability rank condition yields $\text{rank}(C_o) = 8 = n$, indicating controllability of the linearised model at the equilibrium states.

Given the complexity of A and B for the fully parametrised model it is not possible to directly examine their structure to ascertain the effect of parameter choice on controllability, so the effect of exceptional parameter values is examined instead. Physically impossible parameter choices, such as $r_w = 0$, $l_1 = l_2$ etc. are not considered. The conditions on α_i and l_i set out in Remark 1 must be met for the model to remain defined. Setting $h_p = 0$ yields $\text{rank}(C_o) = n - 1$ and thus a reduction in controllability, meaning the pendulum CoM must not lie inline with the wheel axis.

A nonlinear system that is controllable when linearised at its equilibrium states is STLC from the equilibrium states for the full nonlinear system [89, p. 74-75], meaning the CMD is STLC for $x \in \mathcal{X}_e$ given sensical parameter choices.

For comparison a two-wheeled inverted pendulum moving on a 2D plane yields $\text{rank}(C_o) = 6$, as the nonholonomic constraints imposed by the use of regular wheels prevent translation parallel to the wheel axis. A TWIP on a 2D plane therefore does not satisfy the KCRC, and is therefore not STLC, though a number of authors claim the TWIP to be STLC by analysis of the TWIP's model in joint space [90], which ignores a dimension of the configuration space required to uniquely locate the TWIP on a 2D plane.

Theorem 2. *The CMD is STLA* $\forall\ x \in \mathbb{R}^8$.

Arranging (3.2.35) in the nonlinear input-affine form

$$\dot{x} = f(x) + \sum_{j=1}^{3} g_j(x) u_j \tag{3.2.44}$$

where $x = \begin{bmatrix} p & v \end{bmatrix}^T$, the drift vector field $f(x)$ and input vector fields $g_j(x)$ take the form

$$f(x) = \begin{bmatrix} x_5 \cos(x_3) - x_6 \sin(x_3) \\ x_5 \sin(x_3) + x_6 \cos(x_3) \\ x_7 \\ x_8 \\ f_5(x_4, x_5, x_6, x_7, x_8) \\ f_6(x_4, x_5, x_6, x_7, x_8) \\ f_7(x_4, x_5, x_6, x_7, x_8) \\ f_8(x_4, x_5, x_6, x_7, x_8) \end{bmatrix} \tag{3.2.45}$$

$$\begin{bmatrix} g_1(x) \\ g_2(x) \\ g_3(x) \end{bmatrix}^T = \begin{bmatrix} & 0_{4\times 3} & \\ g_{51}(x_4) & 0 & g_{53}(x_4) \\ 0 & g_{62}(x_4) & 0 \\ g_{71}(x_4) & 0 & g_{73}(x_4) \\ 0 & g_{82}(x_4) & 0 \end{bmatrix} \tag{3.2.46}$$

in which $g_{53}(x_4) \equiv g_{71}(x_4)$.

The distribution spanned by the vector fields f and $g_j,\ j = [1\,..\,3]$ is defined as $\Delta = \text{span}\{f, g_1, g_2, g_3\}$, or in bracket notation $\Delta = \langle f, g_1, g_2, g_3 \rangle$, in which Δ is nonsingular, as assuming sensical parameters $\dim(\Delta) = 4\ \forall\ x \in \mathbb{R}^8$. The accessibility algebra $\mathcal{A}$ is defined as the involutive closure of Δ, written as $\overline{\Delta} = \Delta_{\mathcal{A}}$. A distribution is involutive if $[f, g] \in \Delta\ \forall\ (f, g) \in \Delta$, where $[\ ,\ ]$ denotes the Lie bracket operator, defined as

$$[f_1, f_2](x) = \frac{\partial f_2}{\partial x} f_1(x) - \frac{\partial f_1}{\partial x} f_2(x)$$

The involutive closure of a distribution Δ can be calculated as the distribution spanned by all possible combinations of Lie brackets calculable from its vector

fields, which can be derived iteratively as

$$\Delta_1 = \Delta, \quad \Delta_i = \langle \{\Delta_{i-1},\ \{[X,Y] \mid X \in \Delta_1, Y \in \Delta_{i-1}\}\} \rangle, \quad i \geq 2 \tag{3.2.47}$$

This procedure terminates when $\Delta_{i+1} = \Delta_i = \Delta_{\mathcal{A}}$, with the terminal value of i required to define this distribution referred to as the nonholonomy degree of the system, with an upper bound of $i \leq n - m$ [10].

For the system (3.2.44), clearly $\dim(\Delta_1) = 4$. Δ_2 is calculable as

$$\Delta_2 = \langle \{\Delta_1,\ [\Delta_1, \Delta_1]\} \rangle \tag{3.2.48}$$

which in knowing $[f, f] = 0$, $[G, G] = 0\ \forall\ g_j \in G$, where $G = \{g_1, g_2, g_3\}$, can be simplified to

$$\Delta_2 = \langle \{\Delta_1,\ [f, G]\} \rangle = \langle \{f, g_1, g_2, g_3, [f, g_1], [f, g_2], [f, g_3]\} \rangle \tag{3.2.49}$$

yielding $\dim(\Delta_2) = 7$. Δ_3 is calculable as

$$\Delta_3 = \langle \{\Delta_2,\ [\Delta_1, \Delta_2]\} \rangle \tag{3.2.50}$$

in which a single additional Lie bracket is required to yield the distribution

$$\mathcal{D} = \langle \{f, g_1, g_2, g_3, [f, g_1], [f, g_2], [f, g_3], [f, [f, g_1]]\} \rangle \tag{3.2.51}$$

that is of full rank $\dim(\mathcal{D}) = n$, meaning $\mathcal{D} = \overline{\mathcal{D}}$, and therefore $\Delta_3 \equiv \mathcal{D} \equiv \Delta_{\mathcal{A}}$. This indicates a nonholonomy degree of 3, the same as a TWIP [51, 91]. Unlike a TWIP, it is found that $\dim(\Delta_3) = n$ even for the frictionless $h_p = 0$ case, indicating STLA even when the pendulum mass generates no force on the body due to gravity. This is to be expected, as the full Cartesian state space can be accessed by combinations of rotation about $\hat{b}_z$ and translation along $\hat{b}_x$, whilst using the rotational dynamics about $\hat{b}_x$ to purely control the θ_p subsystem. As $\dim(\Delta_{\mathcal{A}}) = n$ this proves that the CMD is STLA $\forall\ x \in \mathbb{R}^n$.

Theorem 3. *The CMD is kinematically holonomic*

From (3.2.32) *the kinematic model of a CMD can be expressed as a sum of*

vector fields as

$$\dot{p} = \Psi v = \sum_{j=1}^{4} g_j(p) v \tag{3.2.52}$$

As the accessibility distribution formed by these vector fields $\Delta_{\mathcal{A}} = \langle \overline{\Psi} \rangle$ *is found to have full rank* $\dim(\Delta_{\mathcal{A}}) = \dim(p)$*, the individually nonholonomic constraints* (3.1.5) *and* (3.1.7) *are together completely integrable, meaning as in conventional statically stable Mecanum wheeled vehicles the kinematic model* (3.2.32) *is holonomic [10, p. 477]. The CMD is therefore a kinematically holonomic system.*

3.2.3 The Largest Feedback Linearisable Subsystem

Using the adjoint representation of the Lie bracket $[f, g] = \mathrm{ad}_f g$, successive Lie brackets of the vector fields f and g up to j iterations can be defined as

$$\mathrm{ad}_f^j g = \mathrm{ad}_f(\mathrm{ad}_f^{j-1} g), \quad e.g. \quad \mathrm{ad}_f^2 g = [f, [f, g]] \tag{3.2.53}$$

Following the notation of [92]

$$\begin{aligned} G &= \{g_1, g_2, g_3\} \\ G_f &= f + G = \{f + g : g \in G\} \\ \mathrm{ad}_f^j \Delta &= \left\{ \mathrm{ad}_f^j X : X \in \Delta \right\} \\ [\Delta_1, \Delta_2] &= \{[X, Y] : X \in \Delta_1, Y \in \Delta_2\} \end{aligned} \tag{3.2.54}$$

define the distributions

$$Q_0 = \langle g_1, g_2, g_3 \rangle, \quad Q_i = \left\langle \left\{ \overline{Q}_{i-1},\ \mathrm{ad}_f^i Q_0 \right\} \right\rangle \quad i \geq 1 \tag{3.2.55}$$

where $\overline{Q}_i$ denotes the involutive closure of Q_i.

Again, it is clear from the structure of (3.2.46) that $\{[g_1, g_2], [g_1, g_3], [g_2, g_3]\} = 0$, so Q_0 is involutive and therefore $Q_0 = \overline{Q}_0$. Q_1 is calculated as

$$Q_1 = \left\langle \left\{ \overline{Q}_0,\ \mathrm{ad}_f Q_0 \right\} \right\rangle = \langle \{g_1, g_2, g_3, [f, g_1], [f, g_2], [f, g_3]\} \rangle \tag{3.2.56}$$

with involutive closure

$$\overline{Q}_1 = \langle \{g_1, g_2, g_3, [f, g_1], [f, g_2], [f, g_3], [g_1, [f, g_1]], [[f, g_1], [f, g_3]]\} \rangle \tag{3.2.57}$$

which is found to be of full rank $\dim(\overline{Q}_1) = n$, meaning Q_2 must be equivalent as $Q_2 \equiv \overline{Q}_1$.

From these distributions the following sequence of non-increasing integers are computed [92, 93]

$$r_0 = \dim(Q_0) \tag{3.2.58a}$$

$$r_i = \dim(Q_i) - \dim(\overline{Q}_{i-1}), \quad i \geq 1 \tag{3.2.58b}$$

$$k_i^* = \text{card}\{r_j \geq i \mid j \geq 0\} \tag{3.2.58c}$$

in which $r_0 = 3,\ r_1 = 6 - 3 = 3,\ r_2 = 8 - 8 = 0$, giving controllability indices $k_1^* = 2,\ k_2^* = 2,\ k_3^* = 2,\ k_4^* = 0$. This indicates that the largest feedback linearisable subsystem has dimension $n_\lambda = k_1^* + k_2^* + k_3^* = 6$ [92], meaning this subsystem can be rewritten as three linear double integrators. Intuitively, this subsystem will encompass the $(\phi, \dot{\phi})$, $(\theta_p, \dot{\theta}_p)$, and $(\int v_x, v_x)$ dynamics, with the $(\int v_y, v_y)$ dynamics therefore not linearisable by static feedback and state transformation. The size of this maximum feedback linearisable subsystem is greater than that of a TWIP, which has a maximum relative degree of 4 [51].

3.2.4 Dynamics Analysis Conclusion

To summarise this section, the dynamics of the CMD have been derived, and both the complex nonholonomic constraints imposed by the use of Mecanum wheels and the friction forces acting upon their components have been successfully incorporated into a minimal dynamics model with state vector $x \in \mathbb{R}^8$. The CMD has been proven to be a kinematically holonomic and therefore omnidirectional system, which is underactuated and unstable about the upright equilibrium. It has been proven to be fully controllable from its equilibrium states, and can fully access its state space, providing the strongest possible proof of controllability for a nonlinear system. Finally, in proving that the CMD has a relative degree of 6, there exists the possibility of achieving significant model simplification through feedback linearisation, potentially only requiring the use of nonlinear control techniques to stabilise two states, with the remaining six states controllable using linear controllers. It has also been shown that a CMD of $n_w > 3$ can always be simplified to a system of input dimension $m = 3$ by simple linear algebra, simplifying the control of many-wheeled CMDs. The MATLAB script used to perform these derivations and analyses is included in Appendix B.

3.3 Collinear Mecanum Drive Design Considerations

This section aims to explore how the selection of a number of CMD parameters affects the overall CMD dynamics and properties, with the aim of drawing some conclusions as to help guide good CMD design practices.

3.3.1 Wheel Quantity

From Remark 1 it is known that a minimum of three wheels are required to ensure controllability of the CMD. However, this may not be the optimal number of wheels for all applications. One drawback of using only three wheels is a resulting lack of redundancy; loss of traction of a single wheel results in the loss of system controllability. Loss of traction is more likely to occur in a CMD than a conventional wheeled vehicle, as the generation of a net force along $\hat{b}_x$ or $\hat{b}_y$ requires a component of the body force generated by each wheel to be cancelled by an equal and opposite force from the other wheels. This is clear in analysis of Λ in (3.2.28), as in order to generate a net force along $\hat{b}_y$ equally signed torques must be applied to each wheel whilst ensuring $\sum_{i=1}^{n_w} \cot(\alpha_i)\tau_i = 0$, meaning the force components along $\hat{b}_x$ do no work. Similarly, generating a net force along $\hat{b}_x$ requires wheels of opposing $\text{sgn}(\alpha)$ to be rotated against one another whilst ensuring $\sum_{i=1}^{n_w} \tau_i = 0$, meaning force components along $\hat{b}_y$ due to wheel torques do no work. While doing no work, these forces still require the existence of an equal and opposite tractive force between the roller and ground, meaning the CMD is more likely to experience wheel slip than a conventional wheeled vehicle when generating the same net force on the body [5]. This is compounded by the requirement of the Mecanum wheels to present a circular projection, as this necessitates manufacture of the rollers using hard rubber compounds, and prevents the inclusion of a tread pattern as in a typical vehicle tire, reducing wheel traction. An ideal Mecanum wheel has only a single contact point, compared to a cylindrical wheel that has a contact line.

The introduction of a fourth wheel therefore helps to prevent loss of control when a wheel slips due to a transient change in wheel traction, for example when slipping on debris or when moving over small gaps or bumps between floor surfaces. However, this does not reduce incidence of loss of traction due to overly aggressive control, meaning this must be addressed in the control design.

3.3.2 Choice and Ordering of Roller Angles

While only Mecanum wheels ($\alpha = \pm\pi/4$) are currently available off-the-shelf[3], these roller angles may not necessarily be the optimal choice for use in a CMD. Multiple arguments could be made as to the definition of this optimality; this may be the sets of roller angles that allow each wheel to generate the same contribution to body accelerations $\dot{v}_x$ and $\dot{v}_y$ for a given input torque, or the optimal wheel configuration may be that which yields the most evenly distributed accessible acceleration space, or even a distribution most suited to a particular application. This optimality could also be considered in terms of translational efficiency, with the optimal configuration being that which minimises average dissipative losses for translations in every direction.

In this section these three definitions of optimality are examined for both the three and four-wheeled cases. Only two wheel configurations need to be considered for each; for the three-wheeled case these are $\alpha = [a, -a, a]$ and $\alpha = [a, a, -a]$, as the configurations $\alpha = [-a, a, a]$ and $\alpha = [-a, a, -a]$ are identical to the former following a rotation, and for the four wheeled case these are $\alpha = [a, -a, -a, a]$ and $\alpha = [a, -a, a, -a]$, with a similar argument for the other two combinations. Combinations with three wheels of one sign of a and one of the other are viewed as highly unlikely to exhibit desirable properties, and so are not considered further.

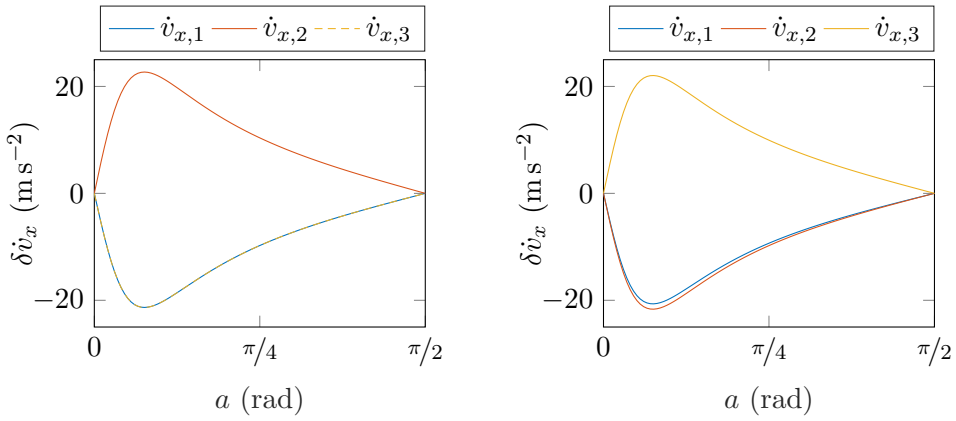
3.3.2.1 Torque Distribution

One definition of the optimal roller ordering could be that which yields equal contributions to body accelerations $\dot{v}_x$ and $\dot{v}_y$ from each wheel torque. This can be determined by analysing the first and second rows of the matrix

$$M_v(p)^{-1}B_v \tag{3.3.1}$$

that multiplies onto τ to yield contributions to $\dot{v}_x$ and $\dot{v}_y$ due to a wheel input torque in (3.2.34). As from Remark 1 $n_w \geq 3$ is required for controllability, the $n_w = 3$ case is considered first. The wheels are assumed to be evenly spaced,

[3] While omniwheels ($\alpha = \pi/2$) are also available off-the-shelf, they cannot be used in a CMD arrangement as $\cot(\pi/2) = 0$, and thus no quantity of wheels are able to generate a unique inverse kinematics mapping.



(a) $\alpha = [a, -a, a]$ (b) $\alpha = [a, a, -a]$

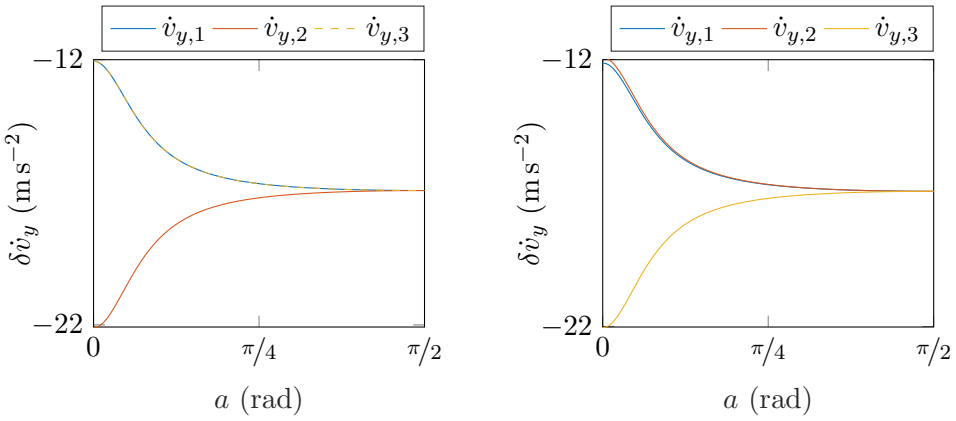
Figure 3.2: Contribution of each wheel torque τ_i to $\dot{v}_x$ for the $n_w = 3$ case in (3.3.1) over $a \in [0, \pi/2]$ with $n_w = 3$, where α takes the form $\alpha = [a, -a, a]$ (a) and $\alpha = [a, a, -a]$ (b). All others parameters are taken from Table 4.2.

placing the middle wheel directly under the CoM, i.e. $l_2 = 0$ and $l_3 = -l_1$, and roller angles are set to $\alpha = [a, -a, a]$, for which the first row of (3.3.1) is found to take the form

$$\frac{1}{\Gamma}\begin{bmatrix} -\sin(a)\left(\gamma + I_{wx} r_w \cos(a)(2m_p h_p^2 - m_p r_w h_p + 2I_{pbx})\right) \\ \sin(a)\left(\gamma + I_{wx} r_w \cos(a)(4m_p h_p^2 - m_p r_w h_p + 4I_{pbx})\right) \\ -\sin(a)\left(\gamma + I_{wx} r_w \cos(a)(2m_p h_p^2 - m_p r_w h_p + 2I_{pbx})\right) \end{bmatrix}^T \tag{3.3.2}$$

where γ and Γ are known scalar functions. This shows that the central wheel is able to generate greater acceleration in $\dot{v}_x$ than the outer two wheels. The $\alpha = [a, a, -a]$ case is too complex to show, but yields a very similar relationship, with an additional imbalance between the two wheels of identical handedness.

These two cases are shown in Figure 3.2 for $a \in [0, \pi/2]$, with all other parameters taken from Table 4.2. It is found that the $\alpha = [a, -a, a]$ case yields 3.7% more total acceleration in $\dot{v}_x$ than the $\alpha = [a, a, -a]$ case for appropriately signed wheel torques of equal magnitude. For the $\alpha = [a, -a, a]$ case it is found that the central wheel is able to generate 6.3% more $\dot{v}_x$ acceleration than the outer two for the same input torque. In both cases maximum acceleration is achieved with roller angles of $\pm 14°$, a significantly shallower angle than used in Mecanum wheels.



(a) $\alpha = [a, -a, a]$ (b) $\alpha = [a, a, -a]$

Figure 3.3: Contribution of each wheel torque τ_i to $\dot{v}_y$ in (3.3.1) over $a \in [0, \pi/2]$ with $n_w = 3$, where α takes the form $\alpha = [a, -a, a]$ (a) and $\alpha = [a, a, -a]$ (b). All others parameters are taken from Table 4.2.

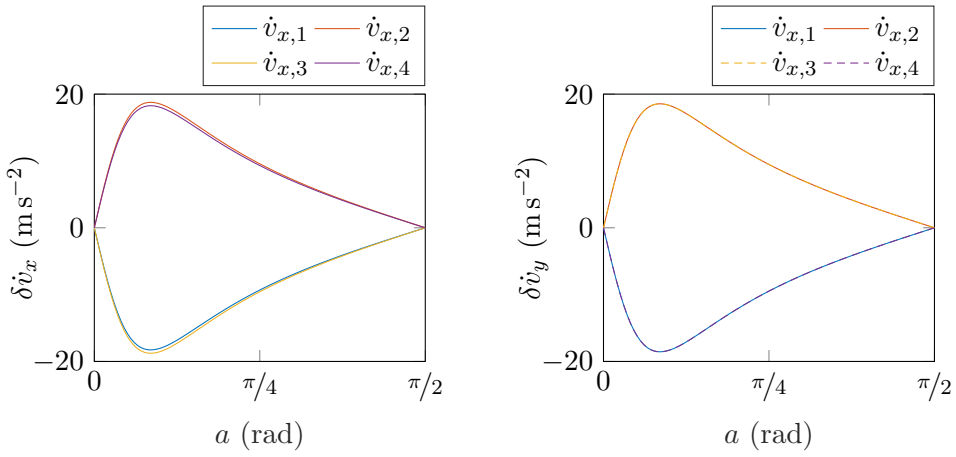
Performing the same analysis for contributions of each wheel torque to $\dot{v}_y$ yields a similar variation between roller orderings, with the $\alpha = [a, -a, a]$ case yielding two wheels with identical contributions, and with the $\alpha = [a, a, -a]$ case introducing a small variation. This also shows that in both cases the single uniquely angled wheel is again able to generate greater $\dot{v}_y$ acceleration than the two wheels of identical handedness.

For the $n_w = 4$ case where the platform is assumed to be symmetrical about the $(\hat{b}_y, \hat{b}_z)$ plane, i.e. $l_4 = -l_1$, $l_3 = -l_2$, $\alpha = [a, -a, a, -a]$, and assuming $l_1 = 2l_2$ as an example since wheels cannot be collocated, the top row of (3.3.1) is found to take the form

$$\frac{1}{\Gamma(a)} \begin{bmatrix} -\sin(a)\left(r_w^3 \cos(a)(5m_w l_1^2 + 2I_{pbz} + 8I_{wyz}) + 3I_{wx} l_1^2 r_w \cos(a)\right) \\ \sin(a)\left(r_w^3 \cos(a)(5m_w l_1^2 + 2I_{pbz} + 8I_{wyz}) + 6I_{wx} l_1^2 r_w \cos(a)\right) \\ -\sin(a)\left(r_w^3 \cos(a)(5m_w l_1^2 + 2I_{pbz} + 8I_{wyz}) + 6I_{wx} l_1^2 r_w \cos(a)\right) \\ \sin(a)\left(r_w^3 \cos(a)(5m_w l_1^2 + 2I_{pbz} + 8I_{wyz}) + 3I_{wx} l_1^2 r_w \cos(a)\right) \end{bmatrix}^T \tag{3.3.3}$$

where $\Gamma(a)$ is a large scalar function. This shows a torque bias toward the inner two wheels when accelerating along $\hat{b}_x$.

Considering the same scenario but now with rotational symmetry about $\hat{b}_z$,



(a) $\alpha = [a, -a, a, -a]$ (b) $\alpha = [a, -a, -a, a]$

Figure 3.4: Contribution of each wheel torque τ_i to $\dot{v}_x$ in (3.3.1) over $\alpha \in [0, \pi/2]$ with $n_w = 4$, where α takes the form $\alpha = [a, -a, a, -a]$ (a) and $\alpha = [a, -a, -a, a]$ (b). This shows uneven torque distribution in (a), and equal distribution in (b). All others parameters are taken from Table 4.2.

such that $\alpha = [a, -a, -a, a]$, all elements of this row become equivalent as

$$\frac{r_w \sin(2a)}{2(4I_{wx} - 4I_{wx}\sin(a)^2 + m_p r_w^2 \sin(a)^2 + 4m_w r_w^2 \sin(a)^2)} \tag{3.3.4}$$

This shows that only when α takes the form $\alpha = [a, -a, -a, a]$ do each of the input torques contribute equally to $\dot{v}_x$. This goes against the intuition that the optimal platform configuration would be symmetrical about the $(\hat{b}_y, \hat{b}_z)$ plane; the optimal platform configuration for distribution of torques when generating acceleration $\dot{v}_x$ instead possesses 2-fold rotational symmetry about $\hat{b}_z$. Obviously this analysis would be more complex with asymmetrical wheel spacings, for which this conclusion would likely not hold. The variation of these coefficients in each case is shown in Figure 3.4 for the parameters in Table 4.2 over $a \in [0, \pi/2]$. Maximum acceleration is found to occur at roller angles of $\pm 15.5°$, a slightly greater angle than the three-wheeled case.

Analysis of the second row of (3.3.1) that multiplies onto τ to yield the contribution of wheel torques to $\dot{v}_y$ shows this to be independent of a for the two considered wheel orderings, so contribution of each wheel torque to $\dot{v}_y$ is invariant in the selection and ordering of roller angles, provided all roller angles possess

the same magnitude.

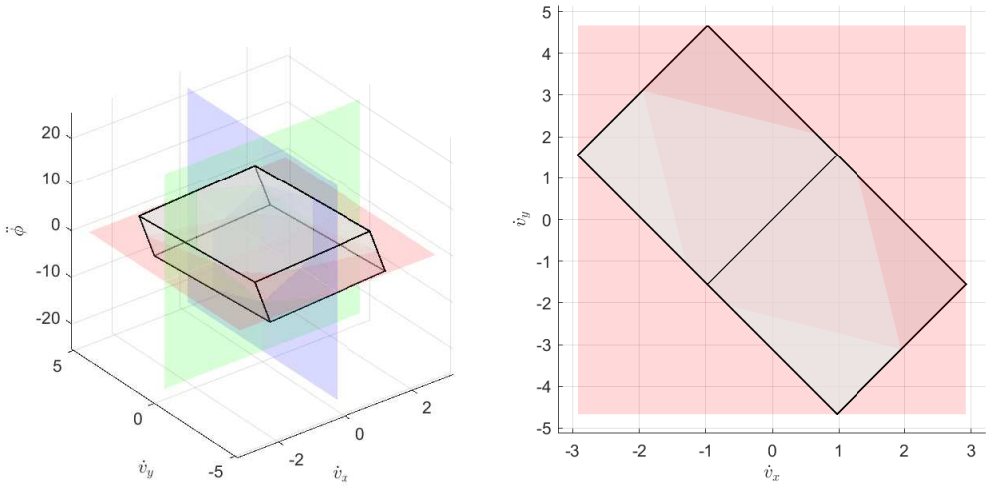
3.3.2.2 Accessible Acceleration Spaces

A second approach to defining the optimal wheel arrangement is to consider the acceleration spaces that can be achieved by each configuration in the presence of input torque constraints. The acceleration space considered here is that achievable at the stationary upright position with $|\tau| \leq \overline{\tau}$, where $\overline{\tau} = 0.1\,\mathrm{N\,m}$. This is shown in Figure 3.5 for the $n_w = 3$ case for $\alpha = [\pi/4, -\pi/4, \pi/4]$ and $\alpha = [\pi/4, \pi/4, -\pi/4]$. Examining the intersection of the red $(\dot{v}_x, \dot{v}_y)$ plane with this accessible acceleration set shows that a larger set of $\dot{v}_x$ and $\dot{v}_y$ accelerations are achievable for the $\alpha = [\pi/4, -\pi/4, \pi/4]$ case, assuming $\ddot{\phi} = 0$. For both cases, this set demonstrates a strong directional bias, and is a poor approximation of an ideal spherical or ellipsoidal acceleration set.

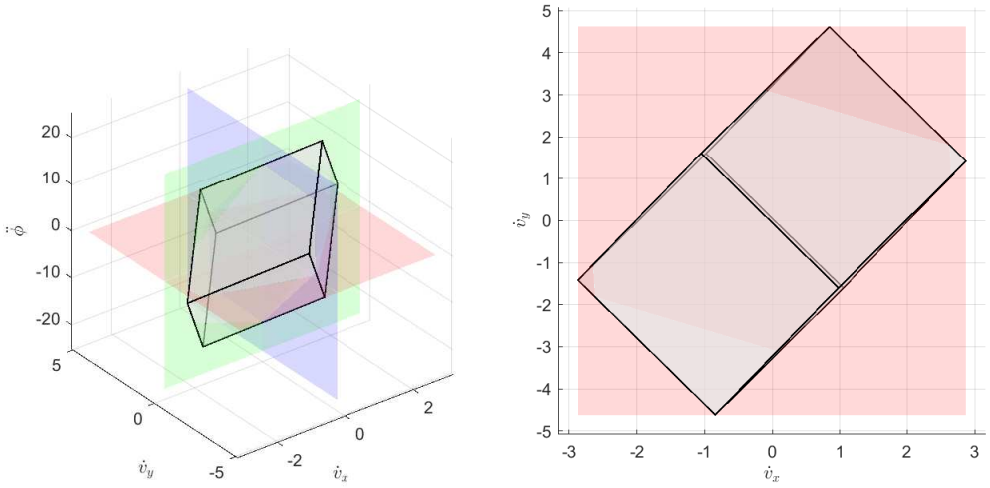
In the four wheeled case, the forward dynamics mapping is underdetermined, meaning there exists multiple combinations of input torques that yield the same net body accelerations. Two acceleration spaces are therefore considered: that achievable using directly constrained wheel torques, i.e. $\tau \in \left\{\mathbb{R}^4 : |\tau| \leq \overline{\tau}\right\}$, and a subset of this space defined by the set of accelerations for which the inverse dynamics least-squares solution yields constraint satisfying wheel torques, defined as

$$\dot{v} \in \left\{\mathbb{R}^3 : \left|B^+ M(p)\dot{v}\right| \leq \overline{\tau}\right\} \tag{3.3.5}$$

These sets are visualised in Figure 3.6, in which the inner dark space represents the inverse dynamics least-squares solution set, and the outer lighter space the less-efficient maximal achievable acceleration set. While the least-squares solution sets retain a similar form to the three wheeled case, the maximal accessible acceleration spaces exhibit shapes much closer to an ideal spherical or ellipsoidal distribution. Examining the intersection of the red $(\dot{v}_x, \dot{v}_y)$ plane with both acceleration sets shows the $\alpha = [\pi/4, -\pi/4, -\pi/4, \pi/4]$ case yields greater accessible sets of $\dot{v}_x$ and $\dot{v}_y$ accelerations when $\ddot{\phi} = 0$.

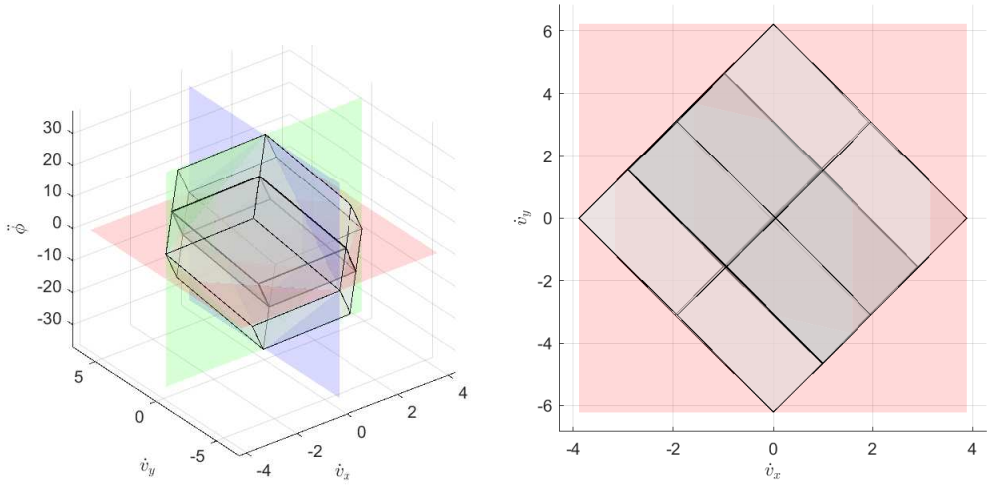


(a) $\alpha = \left[\pi/4, -\pi/4, -\pi/4\right]$

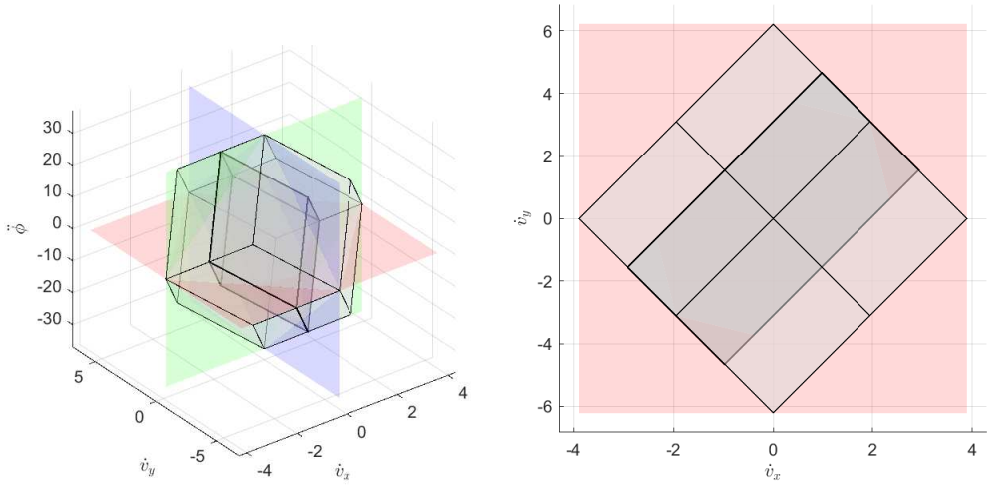


(b) $\alpha = \left[\pi/4, -\pi/4, \pi/4\right]$

Figure 3.5: Sets of feasible body accelerations with wheel torque constraints $\overline{\tau} = 0.1\,\mathrm{N\,m}$ for varying orderings of α, calculated at the stationary upright equilibrium with $n_w = 3$. The coloured planes are orthogonal, intersect the origin, and are aligned with the figure axes. The right plots show a top-down view, intended to make clear the region of intersection between the $(\dot{v}_x, \dot{v}_y)$ plane and the feasible acceleration volumes. Examining the intersection of the red $(\dot{v}_x, \dot{v}_y)$ plane with the feasible acceleration set shows that a larger set of $\dot{v}_x$ and $\dot{v}_y$ accelerations are achievable for the $\alpha = [\pi/4, -\pi/4, \pi/4]$ (b) case.



(a) $\alpha = [\pi/4, -\pi/4, \pi/4, -\pi/4]$



(b) $\alpha = [\pi/4, -\pi/4, -\pi/4, \pi/4]$

Figure 3.6: Sets of feasible body accelerations with wheel torque constraints $\overline{\tau} = 0.1\,\mathrm{N\,m}$ for varying orderings of α, calculated at the stationary upright equilibrium with $n_w = 4$. The outer lightly shaded set represents the achievable acceleration space for $\tau \in \{\mathbb{R}^4 : |\tau| \leq \overline{\tau}\}$, and the interior darker shaded subset is derived from the least-squares solutions to the inverse dynamics mapping that satisfy constraints, i.e. this represents the set of accelerations that can be generated using minimum effort. The coloured planes are orthogonal, intersect the origin, and are aligned with the figure axes. The right plots show a top down view, intended to make clear the region of intersection between the $(\dot{v}_x, \dot{v}_y)$ plane and the feasible acceleration volumes.

3.3.2.3 Translation Efficiency

The CMD's rate of energy dissipation due to viscous and rolling friction can be calculated from (3.2.34) as

$$P = -v^T F_v v \tag{3.3.6}$$

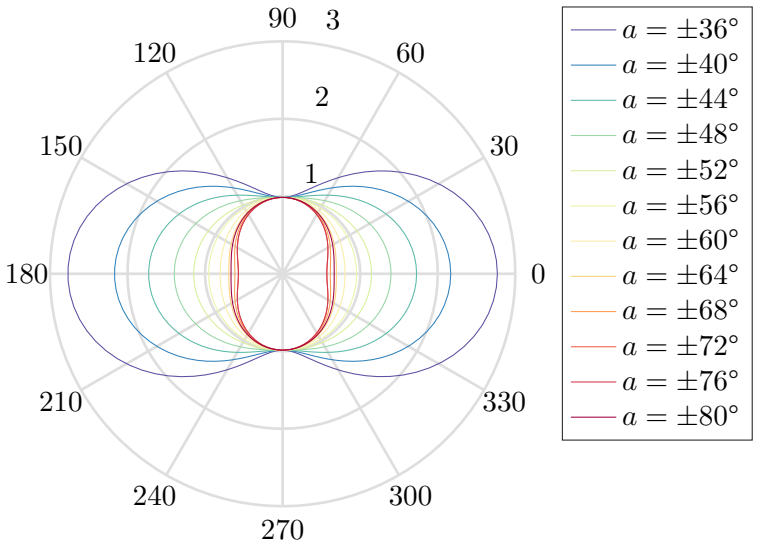
Similarly, the rate of energy dissipation due to motor resistive losses can be calculated as

$$P = R_w \sum_{i=1}^{n_w} \left(\frac{\tau_i}{K_t} \right)^2 = R_w \sum_{i=1}^{n_w} \left(\frac{(B_v^+ F_v v)_i}{K_t} \right)^2 \tag{3.3.7}$$

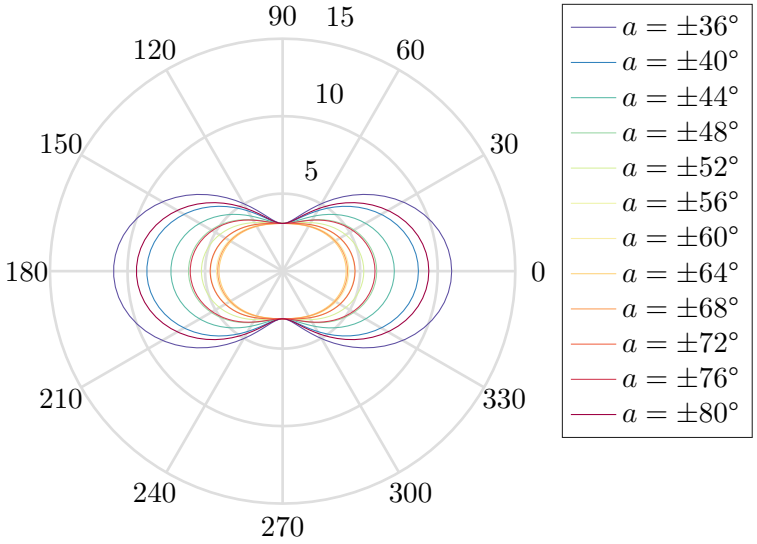
where $(B^+ Fv)_i$ denotes the ith element of the vector $B_v^+ F_v v$, K_t the motor torque constant, and R_w the motor terminal resistance.

As these are the only dissipative elements in the model, the total power required to maintain a given velocity v is therefore the sum of (3.3.6) and (3.3.7), an expression from which it is possible to analyse the efficiency of motion for any given wheel configuration. Figure (3.7) shows this function evaluated for a unit body frame velocity in all directions over a range of roller angles for $n_w = 4$ and $\alpha = [a, -a, -a, a]$, performed for both the motors used in the prototype in this thesis ($K_t = 0.037\,\mathrm{N\,m\,A^{-1}}$, $R_w = 0.61\,\Omega$) and for a smaller motor ($K_t = 0.01\,\mathrm{N\,m\,A^{-1}}$, $R_w = 1.2\,\Omega$) as a comparison. A similar analysis is not performed for the $\alpha = [a, -a, a, -a]$ case, as (3.3.6) and (3.3.7) are found to be relevantly invariant in roller ordering, and so this figure would be nearly identical to the $\alpha = [a, -a, -a, a]$ case. This analysis yields optimal roller angles of $a = \pm 74°$ for the motor used in the prototype in this thesis, and $a = \pm 65°$ for the smaller motor. For the larger motors used in this thesis, a roller angle of $a = \pm 56°$ allows for equally efficient motion in all directions. Interestingly, both of these figures again show that the standard Mecanum wheel roller angle of $\alpha = \pm 45°$ is suboptimal with respect to this metric.

For the three-wheeled case translational efficiency is found to depend heavily on choice of roller ordering. Figure 3.8 shows a comparison between the two possible orderings $\alpha = [a, -a, a]$ and $\alpha = [a, a, -a]$, using the same motor parameters as that used on the prototype in this thesis. Interestingly, this shows around a 10° variation in the optimal roller angle between the two orderings, and the most efficient choice of a for the $\alpha = [a, -a, a]$ configuration in Figure 3.8a requires 45% less power to translate along $\hat{b}_x$ than the most efficient choice of a



(a) $K_t = 0.037\,\mathrm{N\,m\,A^{-1}}$, $R_w = 0.61\,\Omega$



(b) $K_t = 0.01\,\mathrm{N\,m\,A^{-1}}$, $R_w = 1.2\,\Omega$

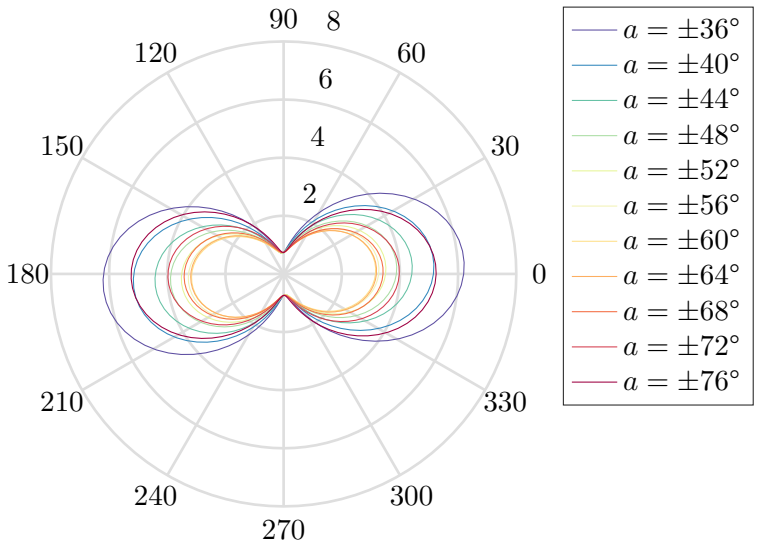
Figure 3.7: CMD translation power requirements over body relative translation direction and roller angle α, with $n_w = 4$ and $\alpha = [a, -a, -a, a]$, using motor parameters from that used on the prototype in this thesis (a) and a smaller motor with $K_t = 0.01\,\mathrm{N\,m\,A^{-1}}$ and $R_w = 1.2\,\Omega$ (b). All other CMD parameters are taken from Table 4.2. This demonstrates a strong relationship between locomotion efficiency and both choice of roller angle and motor selection.

for the $\alpha = [a, a, -a]$ configuration in Figure 3.8b. Both configurations show a substantial decrease in efficiency for translation along $\hat{b}_x$ compared to $\hat{b}_y$, even for the most efficient choices of a. This is in significant contrast to the four-wheeled case with identical motors, for which there exists choices of a that yield more efficient translation along $\hat{b}_x$ than along $\hat{b}_y$.

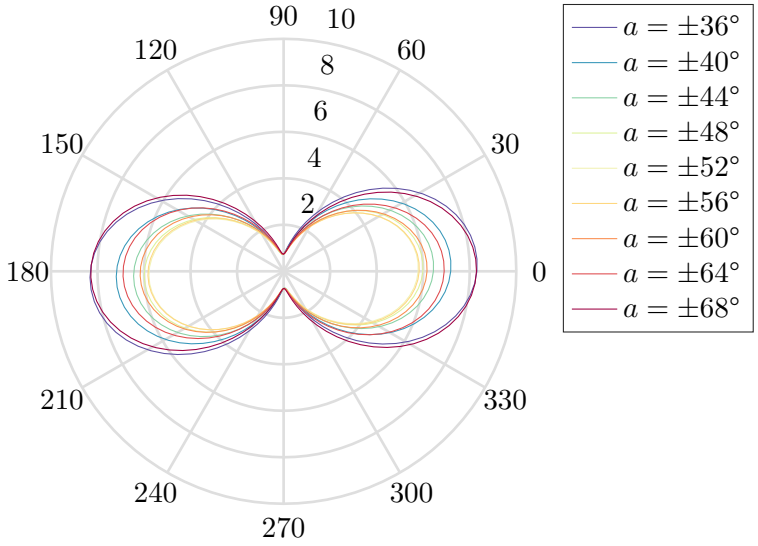
Analysis of losses purely due to the viscous and rolling friction term (3.3.6) finds this function to be invariant in roller ordering, and of significantly lower magnitude than the overall losses in Figure 3.8. This function is shown in Figure 3.9. This means that the substantial change in power requirements with direction visible in Figure 3.8 originates in the motor losses term (3.3.7), which is larger due to the increased torque requirement imposed on the single unique wheel. The efficiency of a three-wheeled configuration can therefore be greatly increased by driving the single unique angled wheel using a larger motor than the other two as to minimise resistive losses.

3.3.3 Wheel Spacing

As one would expect, maximum yaw authority is achieved by spacing the wheels as widely as possible, with symmetry across the $(\hat{b}_y, \hat{b}_z)$ plane. Less obviously, it should also be considered that the opposing forces generated by each wheel in directions parallel to $\hat{b}_y$ when moving along $\hat{b}_x$ generate a net torque on the ground between wheels of opposing handedness, with this torque proportional to wheel spacing. On a hard surface this torque is inconsequential, but on soft, movable, surfaces this could result in an undesirable rotation of the surface under the platform. This torque can be minimised by reducing the distance between wheels of opposite handedness, meaning a good configuration of four wheels is an arrangement of two pairs of wheels with opposite handedness within each pair, with minimal distance between the wheels within each pair. These can be arranged such that the inner two wheels 2 and 3 share the same roller handedness, preventing the generation of a net torque between these two wheels. As the two pairs are to be widely spaced as to provide maximum yaw control authority, it is these two inner wheels that can potentially generate the largest net ground torque when $v_x \neq 0$, which is eliminated by this choice of $\alpha = [a, -a, -a, a]$. While the pairs (1,2) and (3,4) will also each generate a net ground torque, as the wheels in each pair are tightly spaced this torque is minimised.



(a) $\alpha = [a, -a, a]$



(b) $\alpha = [a, a, -a]$

Figure 3.8: CMD translation power requirements over body relative translation direction and roller angle α, using motor parameters for the prototype, with $n_w = 3$, $l = [d, 0, -d]$ for both figures, and with roller angles $\alpha = [a, -a, a]$ (a) and $\alpha = [a, a, -a]$ (b). All other CMD parameters are taken from Table 4.2.

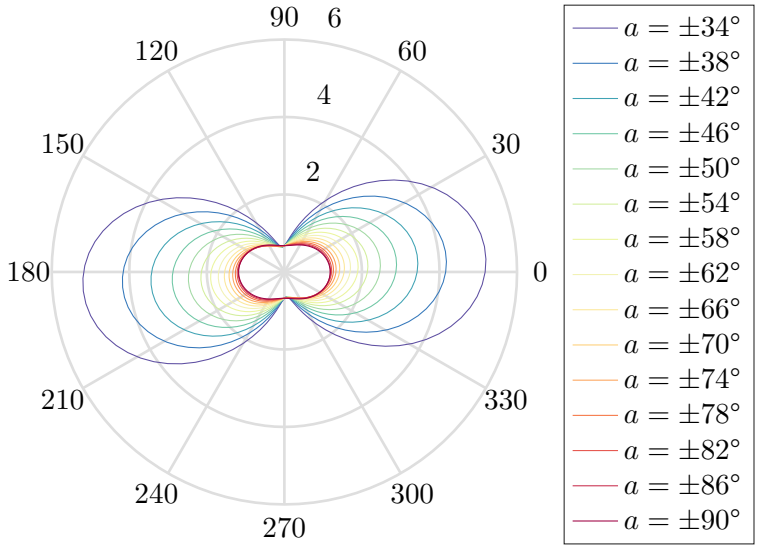


Figure 3.9: CMD rate of energy dissipation due to viscous friction for $n_w = 3$ and $l = [d, 0, -d]$. This choice of wheel spacings makes F invariant in d, so all choices of d produce identical friction profiles. All other CMD parameters are taken from Table 4.2.

3.3.4 Center of Mass Height

The effect of a varying center of mass height is studied by examining the $\dot{v}_y$ acceleration achieved when maintaining a constant θ_p, i.e. $\dot{\theta}_p = \ddot{\theta}_p = 0$. As a varying center of mass would usually also be associated with a varying I_{pbx}, the pendulum body is approximated as a solid thin rod with inertia $I_{pbx} = \frac{1}{3}(2h_p^2)m_p$. This relationship for $h_p \in [0, 2]$m is shown in Figure 3.10, in which all other states and inputs remain at zero. This shows the relationship one would intuitively expect, with higher CoMs yielding greater accelerations for the same non-zero θ_p, reaching an asymptote as $h_p \to \infty$. This function shows strong linearity over small accelerations up to around $5\,\mathrm{m\,s^{-2}}$. Two considerations must be balanced in choosing h_p; a higher CoM allows for greater acceleration for a given θ_p, allowing the system to follow Cartesian trajectories with reduced deviation from the upright pose, though with diminishing returns for $h_p \gtrapprox 0.3\,\mathrm{m}$. However, a higher CoM in turn requires larger translations of the base to move under the CoM for a given nonzero θ_p, and a higher CoM increases the inertia of the pendulum, reducing the ability of the system to control $\ddot{\theta}_p$ directly through the

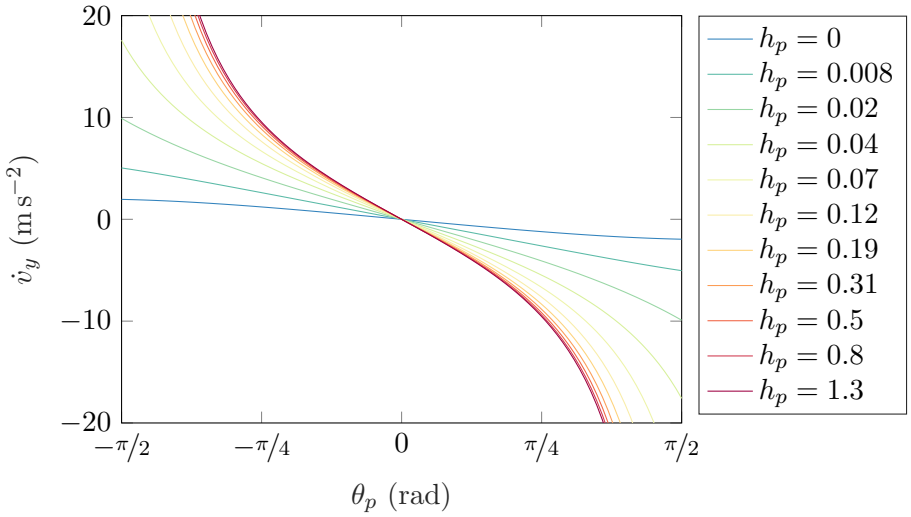


Figure 3.10: $\dot{v}_y$ over θ_p for varying positive h_p, with wheel torques controlled such that $\ddot{\theta}_p = 0$.

body reaction torque applied by the wheel motors. These effects result in the pendulum base having to make greater corrective movements to maintain balance during disturbance.

3.3.5 Overall Optimal Parameter Choice

For the $n_w = 3$ case the analyses of torque distribution, accessible acceleration spaces, and translation efficiency all point to an optimal wheel handedness ordering of $\alpha = [a, -a, a]$. For optimal translation efficiency a roller angle of $a \approx \pm 60°$ should be chosen, whereas for maximum control authority over $\dot{v}_x$ angles of $a \approx \pm 14°$ should be chosen, a significant contradiction. For the $n_w = 4$ case these analyses all point to an optimal wheel handedness ordering of $\alpha = [a, -a, -a, a]$, and again a contradiction is found between the optimal roller angle in terms of translational efficiency at $a \approx 75°$, and in terms of $\dot{v}_x$ control authority at $a = \pm 15.5°$.

While for the cases where wheels of equal roller angles are to be used it is clear that the orderings $\alpha = [a, -a, a]$ and $\alpha = [a, -a, -a, a]$ should be observed, the otherwise contradicting requirements listed above for different perspectives of wheel configuration optimality mean that no clear conclusion can be drawn as to a single or set of overall optimal wheel configurations. This analysis has, however,

shown that motor selection can have a significant impact on overall translational efficiency, and that there exists complex coupled interactions between the selection of each parameter and these differing views of optimality. The conclusion to be drawn is therefore that the robot designer should perform a simultaneous end-to-end optimisation of all CMD and motor parameters for each specific CMD application.

As all of these conclusions are subject to modelling accuracy, a future effort should be made to undertake a full system identification of a Mecanum wheel in order to ensure that the linear friction models used in this thesis are of sufficient accuracy for the analyses in this section to remain accurate. It may be that friction in other interfaces yield significant unmodelled dynamics that would alter these results, such as friction between the roller and its mounting during axial roller loading.

3.4 Conclusion

This chapter has introduced the Collinear Mecanum Drive, a novel dynamically balancing omnidirectional locomotion system. It has derived the CMD's inverse kinematic and dynamics models, analysed the CMD in terms of controllability and accessibility, has analysed the CMD's underactuation characteristics, and has determined the size of the largest feedback linearisable subsystem. It has then explored the effect of parameter selection on control authority, achievable accelerations in the presence of wheel torque constraints, and translation efficiency, granting insight toward good CMD design practice.

Chapter 4

A CMD Experimental Prototype

This chapter serves to detail the reasoning and methodology behind the design and construction of a Collinear Mecanum Drive prototype, which is to be used for the experimental validation of the control and trajectory planning techniques developed throughout the rest of this thesis. Where necessary, experimental methods are described for the measurement of unknown model parameters, and an online nonlinear full-state estimator is derived and demonstrated.

4.1 Prototype 1 - Proof of Concept

The CMD concept was imagined within the first months of this research, at which point only a single previous CMD example existed in the literature [85], which did not conclusively demonstrate that this invention was feasibly realisable. To ascertain this, a basic proof-of-concept prototype was created, shown in Figure 4.1. This prototype had no suspension, and so constantly suffered from wheel slip, used geared brushed motors which introduced a significant amount of backlash, could only control motor open-loop speeds rather than directly controlling torques, and used external encoders, requiring custom mechanical coupling that introduced measurement error.

Despite these numerous shortcomings, this prototype demonstrated significantly better performance than the existing state-of-the-art [85], with the rudimentary open-loop tracking of a square trajectory shown in Figure 4.2. Where the system demonstrated in [85] showed very slow and indirect omnidirectional movement, taking 60 s to move approximately 10 cm along its wheel axis, the first prototype of this thesis was able to perform translations of 1 m in 6 s. This prototype therefore more conclusively proved the real-world feasibility of the CMD

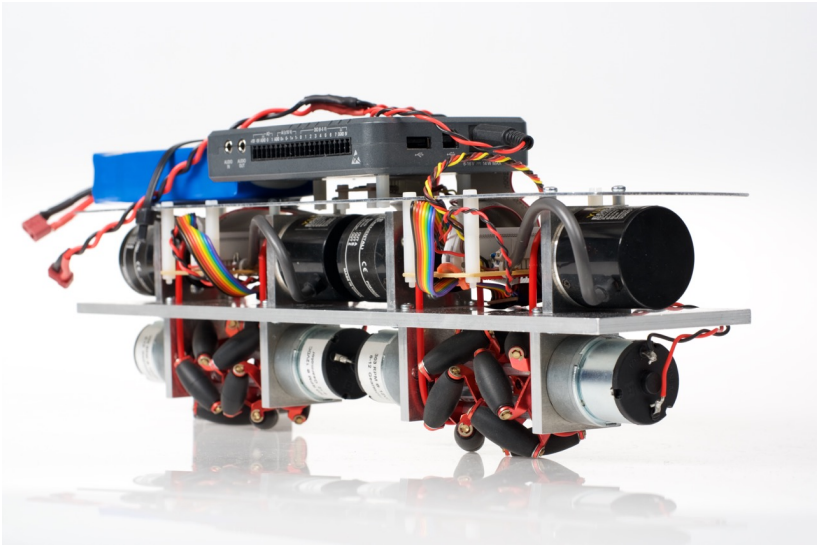


Figure 4.1: The proof-of-concept prototype created at the beginning of this PhD. This did not incorporate any form of suspension, and so suffered from constant wheel slip, used geared brushed motors with a large amount of backlash, and used external encoders with a custom coupling, introducing measurement error.

invention, but suffered from many undesirable nonlinearities, preventing this prototyping from achieving good performance or being accurately described by the derived dynamics model.

4.2 Prototype 2 - An Ideal Research Platform

To better provide experimental validation of the modelling and control developed in this thesis a second improved experimental prototype was created, shown in Figure 4.3. In order to fulfil this purpose it was important that the prototype fit the expected dynamics model as closely as possible. This presents the following idealised specification:

- No backlash between the wheel and motor, and purely viscous motor bearing friction.
- Instantaneous motor torque setpoint tracking.
- Mecanum wheels with perfectly circular profiles, and with minimal roller bearing friction that is purely viscous.

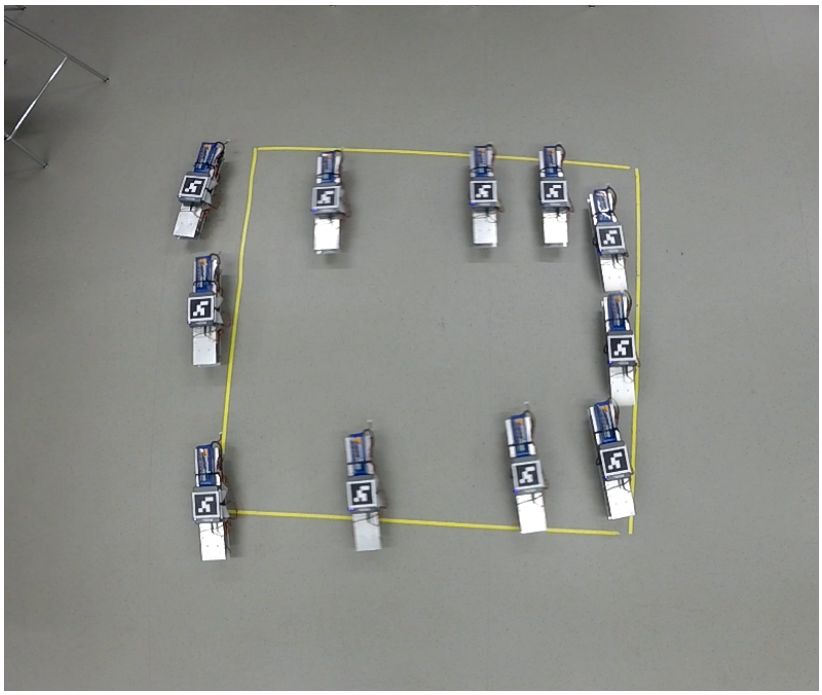


Figure 4.2: The proof-of-concept prototype in Figure 4.1 attempting to track a square trajectory whilst maintaining a constant heading. Each side of the square took approximately 6 s to traverse.

- A fully rigid chassis so as to ensure validity of the rigid body assumption.
- Even and instantaneous weight distribution across the platform's wheels.
- No transport delay in sensing, state estimation, or control.
- Low mass and inertia, so as to better enable highly dynamic manoeuvring.
- A sufficiently low overall mass as to minimise risk to surroundings during experiments.

This specification led to the following component choices:

4.2.1 Wheel Selection

There is relatively limited choice of Mecanum wheels available off-the-shelf, and their complex roller geometry and tight tolerances frustrate low-cost bespoke manufacturing. There are also no off-the-shelf omnidirectional wheels available

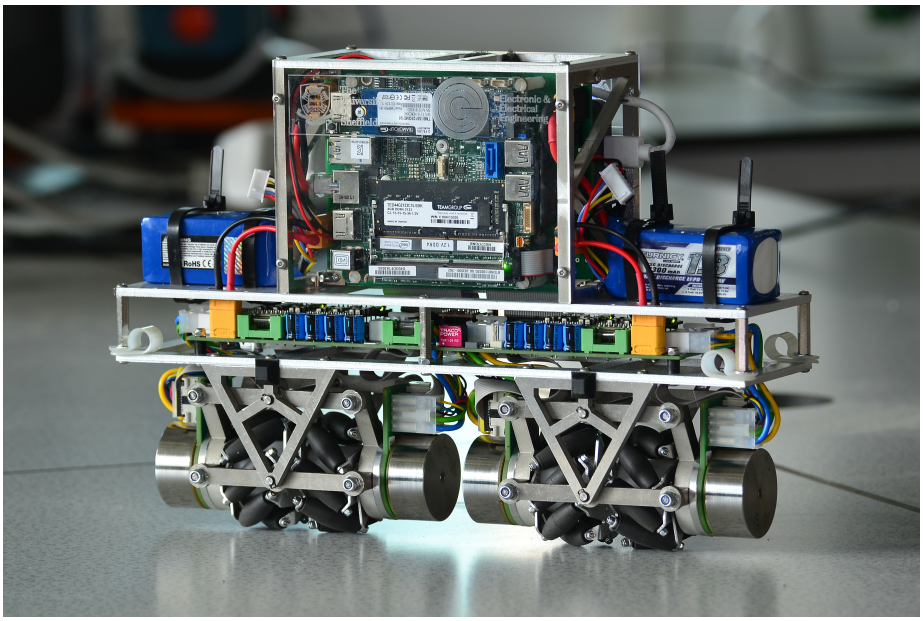


Figure 4.3: The Collinear Mecanum Drive experimental prototype used in this thesis. This features passive weight-distributing suspension and direct-drive brushless motors with integrated high precision encoders, overcoming the main actuation and sensing shortcomings of the proof-of-concept prototype. Compute is provided by both a National Instruments myRIO for low-level control, and an Intel NUC with an i7-8650U processor for high-level planning.

with roller angles other than $\alpha = \pm 45°$. An existing 60 mm Mecanum wheel is therefore chosen. This small wheel diameter was chosen to reduce torque requirements and overall prototype size.

Minimal vibration is apparent during level rotation, and so it is assumed these wheels are sufficiently round to fit the modelling assumption of constant wheel radius. It would be expected that violation of this assumption would be obvious during stationary balancing, as the vehicle would be seen to preferentially balance at certain wheel positions, and would struggle to maintain a constant low velocity. These effects were not noticed in practice. Up to around 1 mm of play exists axially between each of the rollers and the wheel structure, which is assumed to be sufficiently negligible for the purposes of this thesis. There is also a small amount of static friction present in both the roller and wheel bearings, though this is again assumed to be minor enough to be considered negligible.

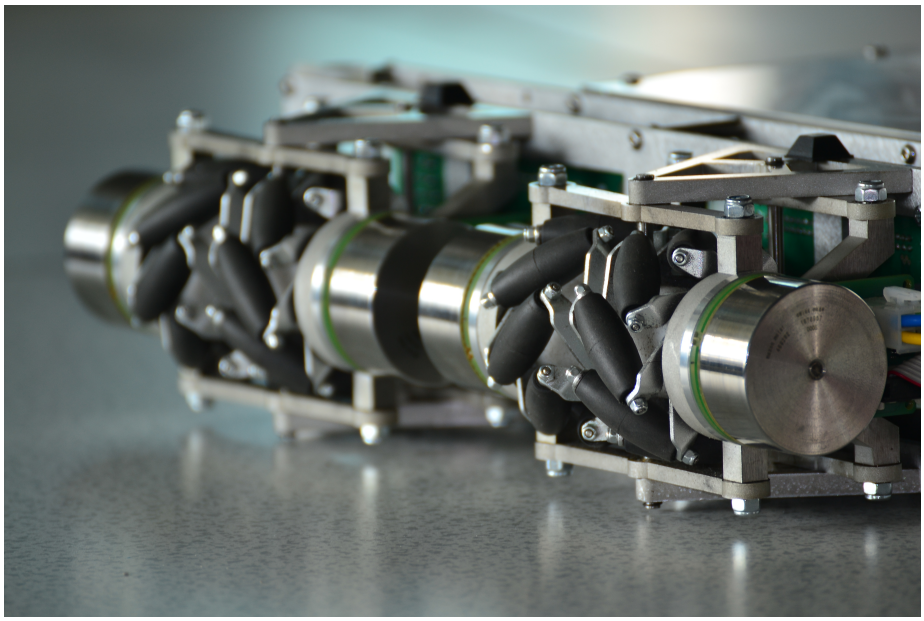


Figure 4.4: A close-up image of the drive assembly of the prototype in Figure 4.3. Visible in each of the four wheel assemblies is a brushless DC motor, attached to a suspended mounting plate and bearing assembly, with its shaft directly coupled to a 60 mm Mecanum wheel.

4.2.2 Wheel Configuration and Suspension

Despite requiring a minimum of only three wheels, in order to preserve symmetry and motor torque distribution a four-wheeled configuration is chosen, arranged in two pairs of opposite handedness. Contrary to the recommended ordering of α discussed in Chapter 3, here the form $\alpha = [a, -a, a, -a]$ is chosen, purely to maintain symmetry for aesthetic purposes.

Suspension of the wheels is required to evenly distribute the prototype's weight. For this thesis the system is assumed to be operating on a perfectly flat surface, so this suspension is not required to filter out any wheel vibration due to ground surface irregularities. This means no spring or dampener components as used in traditional suspension are required, avoiding the introduction of suspension dynamics into the model. Traditional vehicle suspension topologies are therefore disregarded in favour of a simple parallelogram weight distribution method, shown in Figure 4.5. This allows the opposing vertical movement of each wheel in a given pair until both make contact with the ground, whilst ensuring the wheel

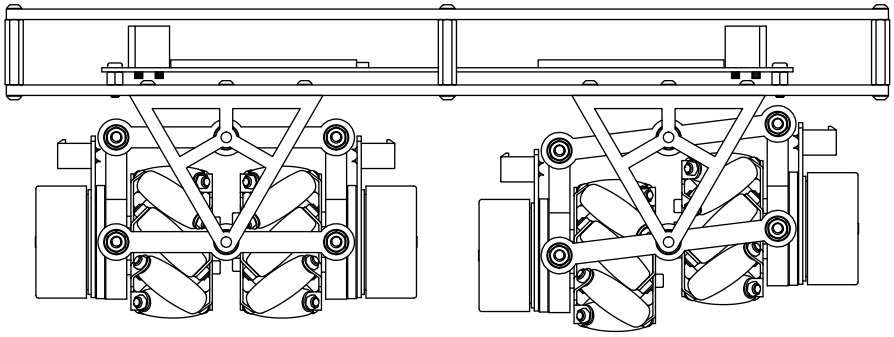


Figure 4.5: Weight distributing suspension design used in the prototype in Figure 4.3. Each individual wheel assembly is able to move vertically relative to its paired neighbour, whilst maintaining parallelism of the two wheel rotation axes, evenly distributing weight applied to each pair of wheels. As the platform only possesses two pairs of wheels weight is then shared evenly between each pair, assuming a centered CoM, yielding even weight distribution between the four wheels. Assuming a level ground surface, the body of the robot will be remain parallel to the ground, and thus the wheel rotation axes also remain parallel to the ground.

rotation axes remain parallel to the ground. This weight distribution is nearly instantaneous, and can be assumed to be perfectly even.

4.2.3 Motor Selection

In order to eliminate backlash a direct drive motor configuration is chosen. Assuming a total mass of 4 kg that is evenly distributed across all four wheels, and assuming a roller to ground static friction coefficient of 0.5, the maximum torque that can be applied without slipping is calculated as 0.15 N m, upper bounding the required motor rated torque. The maximum required speed of the prototype is assumed to be $3\,\mathrm{m\,s^{-1}}$, giving a maximum steady state wheel angular velocity of 955 RPM. As accurate torque control depends on an accurate measurement of motor phase currents, brushless motors are preferred so as to avoid the introduction of noise through mechanical commutation. In order to improve control, any intrinsic torque ripple due to motor construction is to be minimised. A high accuracy measurement of motor position is required, favouring integrated encoder solutions over externally mounted in order to reduce complexity and mounting

Table 4.1: Motor parameters for that used in the prototype in Figure 4.3

Parameter	Value	Unit
Nominal voltage	24	V
No-load speed	6110	rev/min
Nominal speed	4860	rev/min
Nominal torque	128	mN m
Nominal current	3.21	A
Stall torque	1460	mN m
Stall current	39.5	A
Terminal resistance	0.608	Ω
Torque constant	36.9	$\mathrm{mN\,m\,A^{-1}}$
Speed constant	259	rev/min/V
Number of pole pairs	8	
Encoder precision	2048	steps/rev

error. As this application requires operation around zero speed Hall effect sensors are also required. Finally, given the collinear arrangement of wheels a low motor length is desirable to reduce the overall size of the prototype.

This specification makes external rotor brushless motors attractive, as this topology yields a higher torque-to-weight ratio than internal rotor motors, typical exhibits low torque ripple due to a larger number of pole-pairs, are more efficient at the low speeds used in this application, and often have larger diameter to length ratios. An external rotor brushless sensored motor from Maxon Motor is chosen, with model number `397172` and specification given in Table 4.1.

4.2.4 Physical Construction

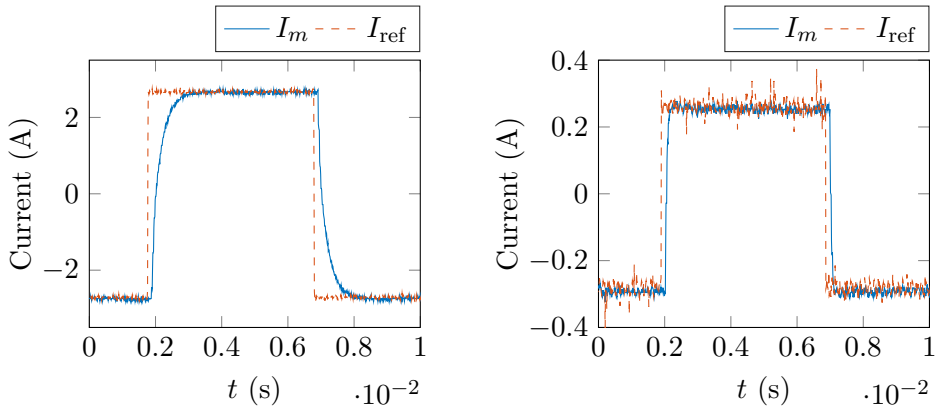
In order to enable highly dynamic manoeuvring, to better highlight modelling error, and to better enable near-stationary balancing, the system is designed to minimise both the height of the center of mass h_p and the pendulum body inertia term I_{px}. The use of a direct drive configuration places a relatively large lower bound on the dimensions of the prototype in the b_x dimension, so this dimension is determined by a relatively tight packing of the wheels and motors. External wheel bearings are incorporated due to the low radial load capacity

of the chosen motors, a characteristic of external rotor motors, at an expense of a small increase in friction. An aluminium chassis construction is chosen to maximise rigidity whilst minimising weight, ensuring validity of the rigid body assumption used in the modelling of the system dynamics. The prototype has bounding dimensions of a width of 337 mm, a depth of 82 mm, and a height of 233 mm.

4.2.5 Motor Control

Motor controllers are required to drive the phase currents of each of the motors to that specified by given setpoints, along with performing electronic commutation. This control is to be as aggressive and with as little transport delay as possible so as to allow for maximum control bandwidth. Whilst this will come at an expense of increased power consumption, gains can be reduced for a particular application and desired control performance if necessary. As similar systems such as quadcopters perform control at update rates of around 400 Hz [94], a motor current loop update rate of at least an order of magnitude greater frequency is required. The controller must have a rated current of at least the nominal current of the chosen motor.

Maxon `ESCON 50/5` controllers are therefore chosen, rated at 5 A. These take an analogue reference signal with 12 bit resolution, and perform PI current control with a loop rate of 54 kHz. Motor current response to small and large reference signals is shown in Figure 4.6, using proportional gain $K_p = 250$ and integral time constant $\tau_i = 350\,\mu s$. Whilst ideally these results would demonstrate a worst case rise time of less than 10% of the system's maximum control update rate, in practice this cannot be achieved without introducing excessive noise into the motor phase currents. As the absolute worst case rise time is close to this target at 20% of the control period, and as the more typical smaller setpoint change achieves a rise time of 5% of the maximum control update rate, it is decided that this controller demonstrates sufficiently high bandwidth torque setpoint tracking to validate the assumption of instantaneous torque setpoint, with the knowledge that the maximum achievable system control aggressiveness will be slightly reduced.



(a) $\tau_r = \pm 0.1\,\text{N m}$ (b) $\tau_r = \pm 0.01\,\text{N m}$

Figure 4.6: Motor current setpoints I_{ref} and tracked trajectories I_m for $\pm 0.1\,\text{N m}$ (a) and $\pm 0.01\,\text{N m}$ (b) 100 Hz square wave torque setpoints, showing a constant 130 µs transport delay within the motor controller and a 486 µs worst case rise time.

4.2.6 Inertial Sensing

Angular rate and linear acceleration sensing is performed using an MPU9250 inertial measurement unit (IMU) from Invensense [95], with full scale ranges of $\pm 500\,^\circ\,\text{s}^{-1}$ and $\pm 2g\,\text{m s}^{-2}$. Communication is performed over SPI at 20 MHz for minimum communication delay, with interrupt driven sampling of the gyroscope and accelerometer at 32 kHz and 1 kHz respectively. Gyroscope measurements are low-pass filtered using a 2nd order Butterworth IIR filter with corner frequency $f_c = 80\,\text{Hz}$, giving a worst case group delay of 1.2 ms and $-27\,\text{dB}$ attenuation at 1 kHz. This filtering is performed on an FPGA to downsample the gyroscope measurements to a more computationally tractable sampling time of $f_s = 1\,\text{kHz}$, whilst avoiding aliasing.

4.2.7 Compute

Data acquisition, state estimation, low-level control, and telemetry are performed using a National Instruments myRIO [96]. This features an ARM A9 processor and a field programmable gate array (FPGA), interfaced by a high speed interconnect. Data acquisition and filtering is performed entirely on the FPGA for minimum jitter, with all remaining tasks performed within Labview RT, running

within a real-time Linux-based operating system on the ARM processor. This is able to run a 1 kHz control loop whilst performing complex background tasks with approximately 10% jitter. A Robot Operating System (ROS) node runs on the ARM processor to provide a standardised protocol for communication with other robotics oriented software. Communication with the controlling desktop PC is performed over WiFi. A second single board computer (SBC) featuring a 64-bit Intel i7-8650U processor is used to perform more computationally demanding planning and control tasks. In floating point benchmarks this processor achieves a performanceo f 18GFLOPS. This is interfaced to the myRIO via Gigabit Ethernet.

Figure 4.7 shows a graphical overview of the organisation of the prototype's subsystems and their interconnections, and Figure 4.8 shows a high-level control diagram.

4.3 Motor Cogging Torque and Kinetic Friction Compensation

Cogging torque is a no-current torque generated by interaction between the rotor permanent magnets and stator slots. This torque causes an undriven motor under a small load to 'jump' between fixed angular positions, and is undetectable from measurement of stator current, meaning it cannot be compensated for by closed-loop current feedback. The magnitude of this torque is independent of stator current, meaning it results in a proportionally larger output torque error for small torque setpoints than large. When a balancing system is exactly at the upright equilibrium no motor torque is required to maintain this in the absence of disturbance, and only small correction torques are required for small deviations from this equilibrium. Output torque error due to cogging torque can therefore have a noticeable effect on control performance during stationary balancing.

Fortunately, as cogging torque is entirely dependent on physical motor construction and rotor angular position, it may be described by a time invariant function of rotor position, which once identified allows an open-loop compensation. In practice, this function is obtained by implementing an angular position controller with integral action to control motor absolute position, using the incremental encoder with a known initial position as an absolute position feedback.

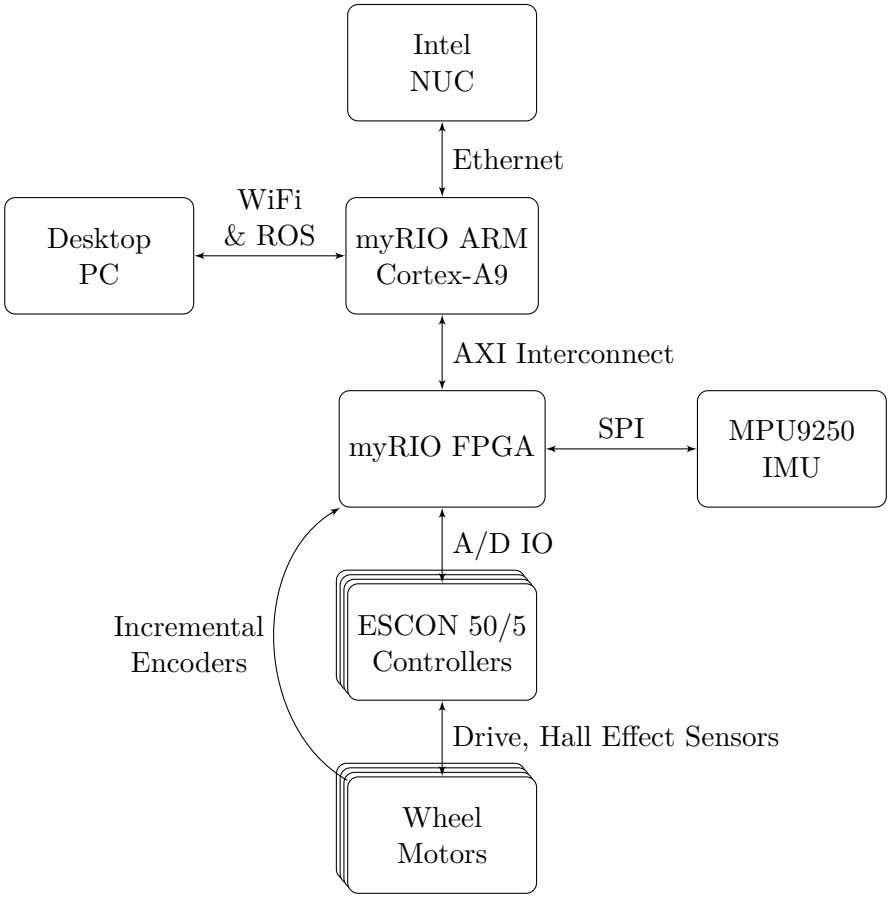


Figure 4.7: A diagram showing the organisation of the prototype's subsystems

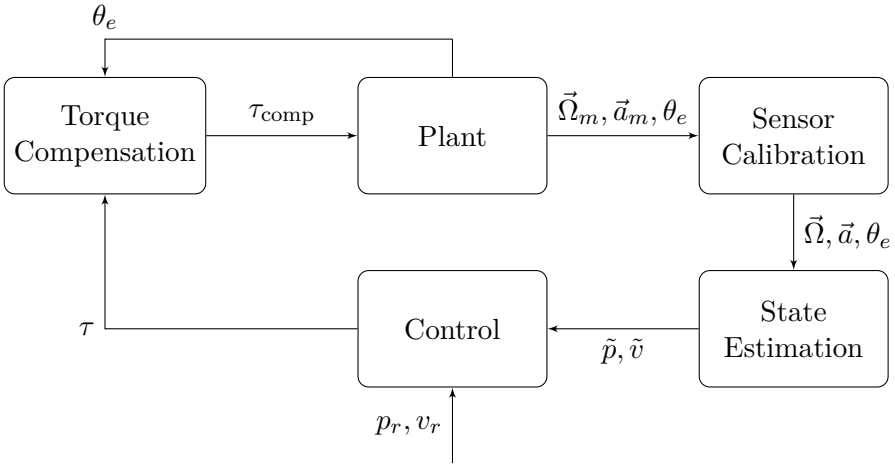


Figure 4.8: Control diagram for the Collinear Mecanum Drive prototype

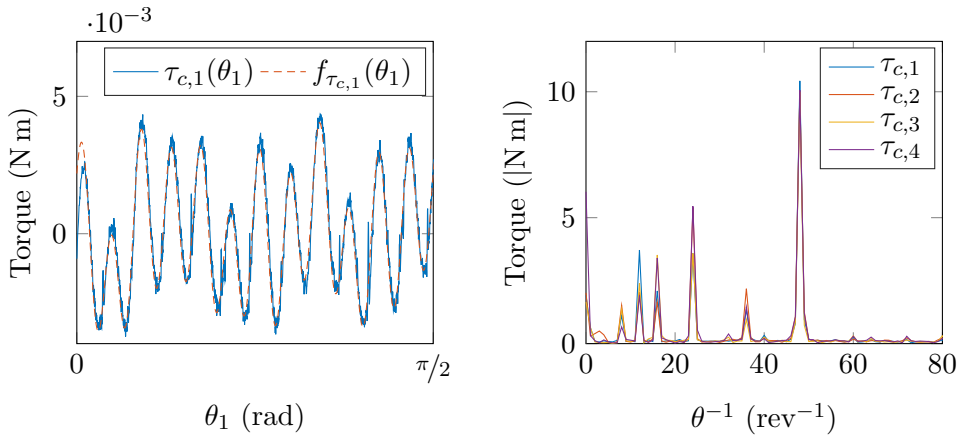


Figure 4.9: Raw cogging torque $\tau_{c,1}$ data for $\theta_1 \in [0, \pi/2]$ (a, blue), the Fourier transform of this data for all four wheels (b), and the resulting cogging torque prediction function derived by the selection of dominant frequency components from the FFT for $\tau_{c,1}$ (a, red).

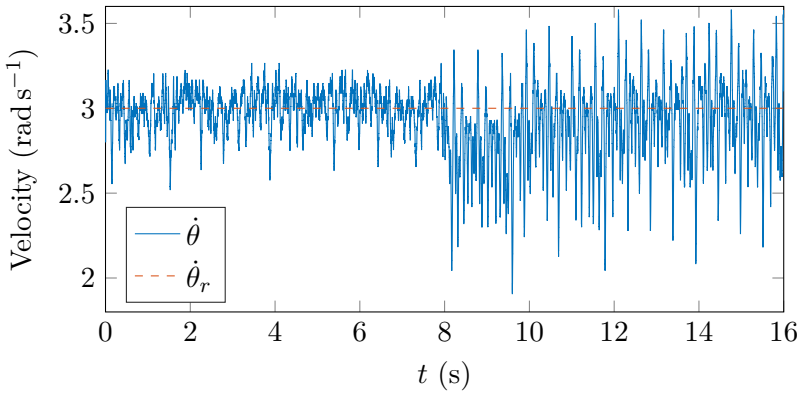


Figure 4.10: Wheel angular velocity whilst running a closed-loop PI speed controller, with proportional gain $K_p = 1$, integral time constant $\tau_i = 6\,\mathrm{s}$, and constant setpoint $\dot{\theta}_r = 3\,\mathrm{rad\,s^{-1}}$, in which cogging torque compensation is enabled until $t = 8\,\mathrm{s}$.

The angular position setpoint is slowly ramped through $\theta \in [0, 2\pi]$, such that motor dynamics have minimal effect on the output response, and so that a steady state is achieved for every encoder position. The resulting steady state torque achieved is therefore that required to hold the rotor in position against the cogging torque, providing a measurement at each encoder position. This is averaged over equal numbers of forward and backward rotations to eliminate static and kinetic friction effects, and to reduce the impact of tracking error on fitting accuracy, with the resulting function for wheel 1 shown in Figure 4.9a, and in the frequency domain for all wheels in Figure 4.9b. The FFT shows a frequency component at 48 cycles/rev with equal magnitude for all wheels, representing six cogging steps per electrical revolution multiplied by the eight pole pairs of the motors. A number of other components exist at factors of this frequency with varying amplitudes, capturing unique variations in motor construction, necessitating individualised cogging torque models. The cogging torque function is determined by selecting only the dominant frequencies from the FFT. No effort is made to compensate for friction at this point, as this is dependent on direction of rotation and wheel loading. Figure 4.10 shows motor angular velocity under a closed-loop PI controller, with a slow constant setpoint of $\dot{\theta}_r = 3\,\mathrm{rad\,s^{-1}}$. Cogging torque compensation is enabled until $t = 8\,\mathrm{s}$, from which a decrease in tracking performance. This serves to demonstrate the effectiveness and value of this compensation.

Once cogging torque can be assumed to be eliminated, it is possible to compensate for kinetic friction in the motor bearings, modelled as the discontinuous torque

$$\tau_k = -\operatorname{sgn}(\dot{\theta}) F_N \mu_k \tag{4.3.1}$$

where F_N denotes the normal force acting through the wheel, and μ_k the coefficient of kinetic friction.

Measurement of this torque can be performed by applying an increasing drive torque to a free-spinning wheel, with τ_k identified as the torque at which an initially slowly spinning wheel exhibits only a very slow exponential deceleration. In order to correctly approximate F_N whilst allowing the wheel to spin free, a mass of $m_p/4$ is attached to the motor in the place of the wheel. This finds a kinetic friction torque of $\tau_k = 3.5\,\mathrm{mN\,m}$, a 3.5% error over the full torque range, proportionally a much larger error over the small range of torques required to

achieve stationary balancing. This compensation yields the overall compensated torque function

$$\tau_{\text{comp}} = \tau - \tau_c + \begin{cases} \text{sgn}(\dot{\theta})\tau_k & \text{for } |\dot{\theta}| > 0 \\ \text{sgn}(\tau)\tau_k & \text{for } \dot{\theta} = 0, \quad \tau \neq 0 \\ 0 & \text{for } \tau\dot{\theta} = 0 \end{cases} \tag{4.3.2}$$

4.4 Parameter Estimation

A number of model parameters are not directly measurable without the use of expensive test equipment. While they can be derived from a perfect CAD model, a number system components are modelled in CAD as homogeneous volumes, meaning they possess incorrect center of mass and inertial properties. Furthermore, the CAD model does not include minor components such as cabling and some fasteners and fixings. This section outlines the experiments used to instead provide a real-world measurement of these parameters.

4.4.1 Wheel Bearing Viscous Friction Coefficient

Applying a slowly ramping torque to each freely rotating wheel connected to a mass of $m_p/4$ allows the measurement of a steady state torque over angular velocity curve. Assuming prior compensation of cogging torque and kinetic friction, these steady state torques are entirely due to viscous friction within the wheel bearings, and motor effects that can be similarly modelled. This experiment is shown for both rotation directions in Figure 4.11, showing a relatively constant gradient between wheels. A linear fitting on this data gives a viscous friction coefficient measurement of $k_{vw} = 2.3 \times 10^{-5}\,\text{N}\,\text{m}\,\text{rad}^{-1}\,\text{s}$.

4.4.2 Wheel Rolling Friction and Roller Viscous Friction Coefficients

Combining (3.1.5) and (3.1.7) for the case where $v_x = \dot{\phi} = 0, v_y \neq 0$ shows that no roller rotation relative to the wheel occurs when translating purely along $\hat{b}_y$. It is therefore possible to measure the effects of wheel rolling friction independently of any friction due to roller rotation. The wheel rolling friction coefficient k_{rw} can be measured by maintaining a constant velocity $v_y \neq 0$ whilst maintaining all

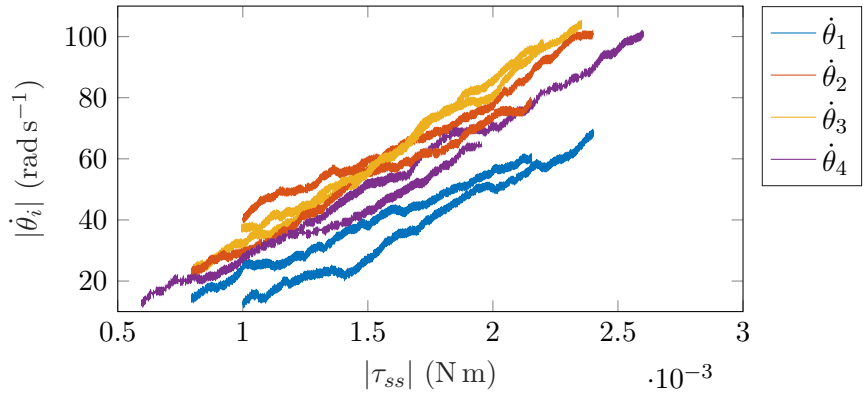


Figure 4.11: Steady state wheel angular velocities over a slowly ramping torque input for both rotation directions. The gradient of this data represents the wheel viscous friction coefficient k_{vw}.

others velocities at $v_x = \dot{\phi} = \dot{\theta}_p = 0$, examining the torque required to maintain this v_y velocity in steady state. Substituting this and the knowledge that $\dot{v} = 0$ in steady state into (3.2.34), substituting roller angles, and assuming a symmetrical wheel spacing yields $M_v(p)\dot{v} = 0$ and $C(p, v)v = 0$, allowing the dynamics model to be simplified to

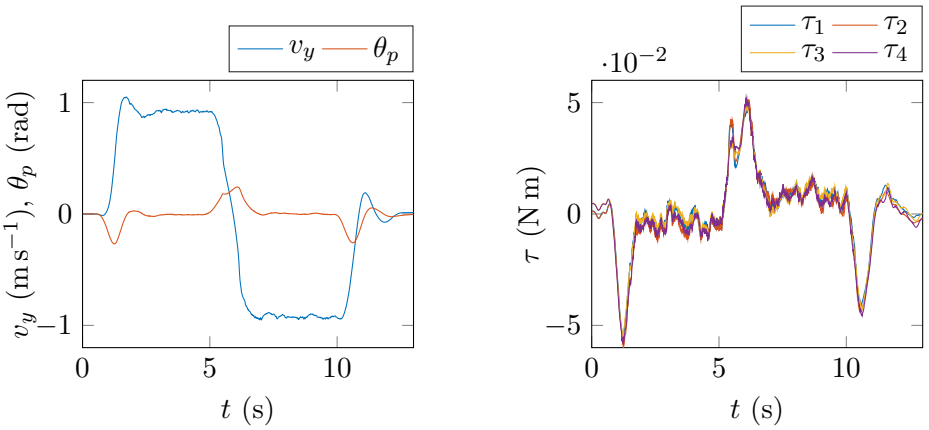
$$\underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ -gh_p m_p \sin(\theta_p) \end{bmatrix}}_{G(p)} + \underbrace{\begin{bmatrix} 0 \\ \frac{-4v_y(k_{vw}+k_{rw})}{r_w^2} \\ 0 \\ \frac{-4v_y k_{vw}}{r_w} \end{bmatrix}}_{Fv} = \underbrace{\frac{1}{r_w}\begin{bmatrix} -\tau_1 + \tau_2 - \tau_3 + \tau_4 \\ -\tau_1 - \tau_2 - \tau_3 - \tau_4 \\ -l_1\tau_1 - l_2\tau_2 - l_3\tau_3 - l_4\tau_4 \\ -r_w(\tau_1 + \tau_2 + \tau_3 + \tau_4) \end{bmatrix}}_{B\tau} \tag{4.4.1}$$

from which k_{rw} can be calculated as

$$k_{rw} = -k_{vw} + \frac{r_w}{4v_y}\sum_{i=1}^{4}\tau_i \tag{4.4.2}$$

which using the steady state data from the experiment shown in Figure 4.12 and the previously measured value for k_{vw} yields $k_{rw} = 1.97 \times 10^{-4}\,\mathrm{N\,s}$.

With k_{rw} known, this procedure can be completed for a constant velocity



(a) (b)

Figure 4.12: Velocity (a) and torque data (b) for a translation $y = [0, 3, 0]$, in which a constant velocity is attained for the majority of each movement.

$v_x \neq 0$ using the similarly derived expression

$$\frac{4k_{vr}v_x}{r_w(r_r - r_w)} - \frac{4v_x(k_{vw} + k_{rw})}{r_w^2} - \frac{8k_{vr}v_x}{r_r r_w} = \frac{1}{r_w}\left(-\tau_1 + \tau_2 - \tau_3 + \tau_4\right) \tag{4.4.3}$$

to calculate k_{vr} as

$$k_{vr} = \frac{r_r(r_r - r_w)(\tau_1 - \tau_2 + \tau_3 - \tau_4)}{4v_x(r_r - 2r_w)} - \frac{r_r(k_{rw} + k_{vw})(r_r - r_w)}{r_w(r_r - 2r_w)} \tag{4.4.4}$$

which for the data in Figure 4.13 yields

$$k_{vr} = 1.01 \times 10^{-4}\,\mathrm{N\,m\,rad^{-1}\,s} \tag{4.4.5}$$

4.4.3 Pendulum Center of Mass Height

It is common knowledge that if an object is suspended by a single point its center of mass will lie directly under the suspension point in steady state. Provided there exists some method of measuring the body-relative position of this vector, multiple measurements can be used to triangulate the center of mass of any object. Fortunately, for this prototype it is reasonable to assume that symmetry exists in the $(\hat{b}_y, \hat{b}_z)$ and $(\hat{b}_x, \hat{b}_z)$ planes, meaning the center of mass must lie

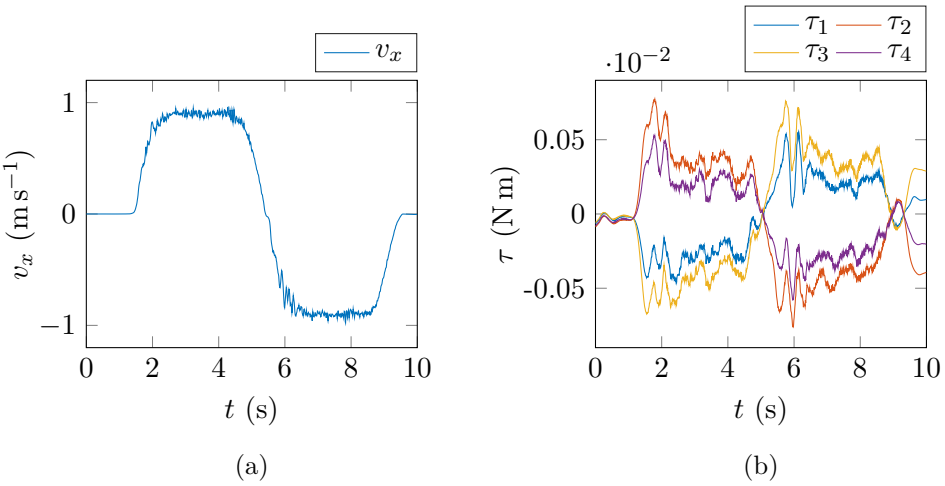


Figure 4.13: Velocity (a) and torque data (b) for a translation $x = [0, 3, 0]$, in which a constant velocity is attained for the majority of each movement.

somewhere along $\hat{b}_z$. This assumption has already been included in the derivation of the dynamics model, with the position of the center of mass on $\hat{b}_z$ parametrised as h_p.

Suspending the prototype from a single point will therefore yield a vector that intersects $\hat{b}_z$, allowing measurement of the overall system's center of mass h_{cm}. This experiment is performed twice for two different suspension points, shown in Figure 4.14, with both experiments yielding $h_{cm} = 0.072\,\text{m}$.

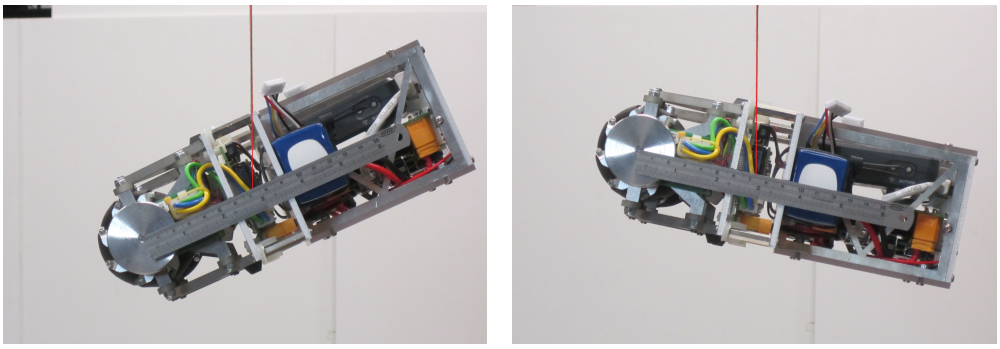
Knowing the mass and location of the wheels, it is then possible to calculate the location of the pendulum's center of mass h_p as

$$h_p = \frac{4m_w h_{cm}}{m_p} + h_{cm} \tag{4.4.6}$$

4.4.4 Pendulum Inertia about the Balance Axis

Rigidly fixing the prototype's wheels and inverting it to allow free rotation of the body about and beneath the wheel axis allows the dynamics model to be simplified to that of a physical pendulum as

$$\ddot{\theta}_p = \frac{-m_p g h_p \sin(\theta_p)}{I_{pbx}} - k_v \dot{\theta}_p \tag{4.4.7}$$



(a) (b)

Figure 4.14: Experiment used to measure h_p, performed by measuring the intersection of the vertical and a ruler attached along $\hat{b}_z$. A red line is superimposed along the string to allow the measurement of h_p. Both measurements yield exactly $h_p = 0.072\,\text{m}$.

in which a viscous friction term $k_v\dot{\theta}_p$ is introduced to model aerodynamic and friction forces damping the oscillation of the pendulum. As the wheels are fixed θ_p can be directly measured by use of the wheel encoders to eliminate state estimation error. Parameters I_{pbx} and k_v can then be estimated by nonlinear least-squares fitting, provided suitable initial conditions as to avoid local minima. An example of this data and the estimated model is shown in Figure 4.15, in which a 99.1% fit is achieved.

4.4.5 Pendulum Inertia about Vertical and Longitudinal Axes

As accurate identification of I_{py} and I_{pz} is a less critical requirement for good control performance, these are simply estimated by modelling the prototype as a solid cuboid as

$$I_{py} = \frac{1}{12}m_p(w^2 + h^2), \qquad I_{pz} = \frac{1}{12}m_p(w^2 + d^2) \tag{4.4.8}$$

where w, d, h denote the bounding width, depth, and height of the prototype respectively, given in Table 4.2.

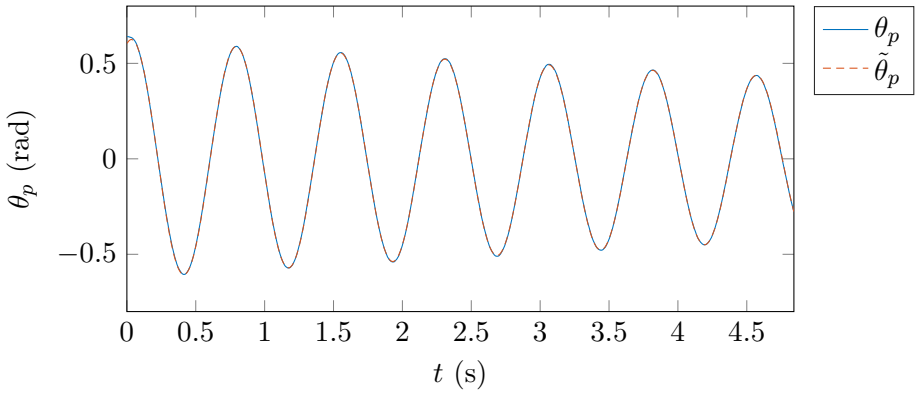


Figure 4.15: Experimental data $\theta_p(t)$ from physical pendulum experiment (blue), with the response predicted by the fitted physical pendulum model (4.4.7) $\tilde{\theta}_p$ (red), achieving a 99.1% fit. θ_p is normalised about the pendulum stable equilibrium.

4.4.6 Wheel Mass and Inertia

Wheel mass m_w is assumed to be equal to the weight of a single Mecanum wheel plus half the weight of an individual drive motor, as a rough approximation of the weight of just the rotor. To estimate the inertia I_{wx} of the wheel and rotor assembly about $\hat{b}_x$ the prototype is secured in a position that allows free rotation of the wheels, and a square wave torque profile with zero mean is applied to the motor. Kinetic and viscous friction can be eliminated by terms similar to (4.3.2) and that in Section 4.4.1, allowing the simple model

$$\ddot{\theta}_w = \frac{\tau + \tau_v + \tau_k}{I_{wx}} \tag{4.4.9}$$

to be used for estimation. Figure 4.16 shows a dataset generated by this method, in which measurements taken around $\dot{\theta} = 0$ are removed due to inaccurate speed measurement at this point. The gradient of a linear fitting of this data represents I_{wx}^{-1}, yielding $I_{wx} = 5.12 \times 10^{-5}\,\mathrm{kg\,m^2}$.

4.4.7 Comparison of Experimentally Measured Parameters with CAD Derived Estimates

The resulting experimentally estimated parameters and their CAD derived values are given in Table 4.2. The CAD model, shown in Figure 4.17, includes all

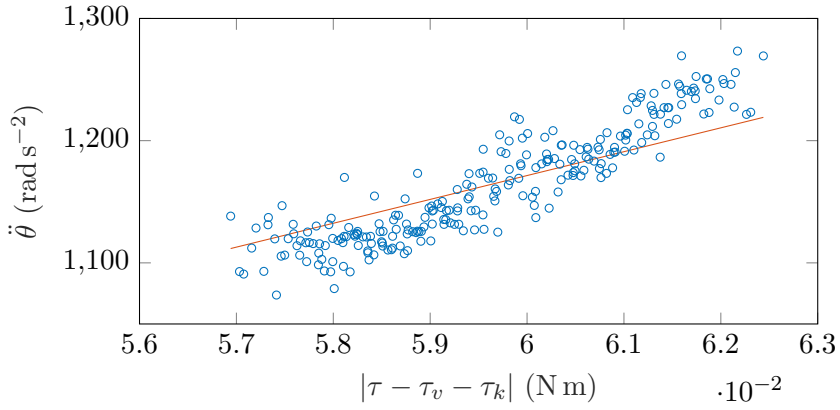


Figure 4.16: Wheel angular acceleration over viscous and kinetic friction compensated torque for $\tau = \pm 0.06\,\mathrm{N\,m}$, along with the linear fit representing an estimate of I_{wx}, in which intersection with the origin is enforced.

of the main system components, but omits minor complex parts such as cable assemblies. This results in an overall mass difference of 0.1 kg, with similar differences in center of mass height and pendulum inertia. A comparison of these values shows a close fit, which is assumed to sufficiently verify the validity of the experimental methods used.

4.5 Sensing

None of the system states (p, v) are directly measurable, meaning they must be estimated using available sensor data. This data is provided by the following sources:

4.5.1 Wheel Encoders

Each Mecanum wheel motor is equipped with a 2048 count/rev incremental quadrature encoder. This provides two square waves in quadrature that can be used to measure a relative change in angular position. These sensors exhibit no drift, and only a small amount of quantization error. This angular position measurement θ_{ei} is relative to the pendulum body as

$$\theta_{ei} = \theta_i - \theta_p + \omega_i \tag{4.5.1}$$

Table 4.2: Table of experimentally and CAD derived parameters for the prototype depicted in Figure 4.3.

Parameter	Unit	Experimentally Measured Value	CAD Derived Value
α_1, α_3	rad	N/A	$\pi/4$
α_2, α_4	rad	N/A	$-\pi/4$
I_{pbx}	$\mathrm{kg\,m^2}$	0.0315	0.0289
I_{pby}	$\mathrm{kg\,m^2}$	N/A	0.0534
I_{pbz}	$\mathrm{kg\,m^2}$	N/A	0.0271
I_{wx}	$\mathrm{kg\,m^2}$	5.12×10^{-5}	4.74×10^{-5}
I_{wyz}	$\mathrm{kg\,m^2}$	N/A	1.1×10^{-4}
m_p	kg	2.64	2.53
m_w	kg	0.145	0.145
h_{cm}	m	0.072	0.070
h_p	m	0.0874	0.0854
$-l_1, l_4$	m	N/A	0.105
$-l_2, l_3$	m	N/A	0.063
r_w	m	0.030	0.030
r_r	m	N/A	0.0055
k_{vw}	$\mathrm{N\,m\,rad^{-1}\,s}$	2.3×10^{-5}	N/A
k_{vr}	$\mathrm{N\,m\,rad^{-1}\,s}$	1.01×10^{-4}	N/A
k_{rw}	N s	1.97×10^{-4}	N/A
h	m	N/A	0.233
w	m	N/A	0.337
d	m	N/A	0.082

where ω_i represents quantisation noise.

4.5.2 Gyroscopes

Three orthogonal gyroscopes are used to provide angular rate measurements about the three pendulum axes as $\vec{\Omega} = \begin{bmatrix}\Omega_p & \Omega_q & \Omega_r\end{bmatrix}^T$. These measurements possess a high signal-to-noise ratio, but are distorted by a significant temperature and time varying bias about zero, sufficiently modelled as an integrated

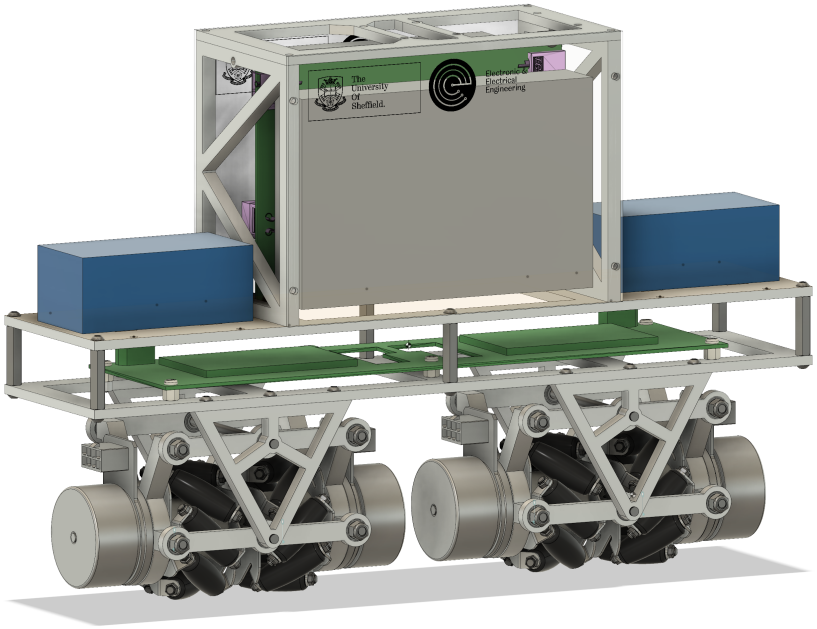


Figure 4.17: CMD CAD model from which parameter verification values are derived. Only components which are straightforward to model are included, i.e. parts such as cabling are omitted.

additive Gaussian white noise of variance σ_g^2. These sensors also suffer from axis misalignment, meaning the gyroscope axes are only approximately orthogonal to one another, along with the IMU package axes not being perfectly aligned with that of the body, presenting as a 'cross-talk' when performing rotations exactly about each of the body axes. For the MPU9250 this cross-axis sensitivity is reported as typically $\pm 2\,\%$ [95], which combined with package mounting error can yield a significant overall misalignment. A scale error is also present, for the MPU9250 reported as $\pm 3\,\%$ at $25\,^{\circ}\mathrm{C}$. Combining these error sources gives the linear sensor model mapping measured angular rates $\vec{\Omega}_m$ to true rates $\vec{\Omega}$ as

$$\vec{\Omega} = \begin{bmatrix} \Omega_p \\ \Omega_q \\ \Omega_r \end{bmatrix} = \begin{bmatrix} 1 & M_{pq} & M_{pr} \\ M_{qp} & 1 & M_{qr} \\ M_{rp} & M_{rq} & 1 \end{bmatrix} \begin{bmatrix} 1/S_p & 0 & 0 \\ 0 & 1/S_q & 0 \\ 0 & 0 & 1/S_r \end{bmatrix} \begin{bmatrix} \Omega_{m,p} - b_p \\ \Omega_{m,q} - b_q \\ \Omega_{m,r} - b_r \end{bmatrix} \tag{4.5.2}$$

The gyroscope biases $\vec{b}_\Omega = \begin{bmatrix} b_p & b_q & b_r \end{bmatrix}^T$, while correlated with sensor tempera-

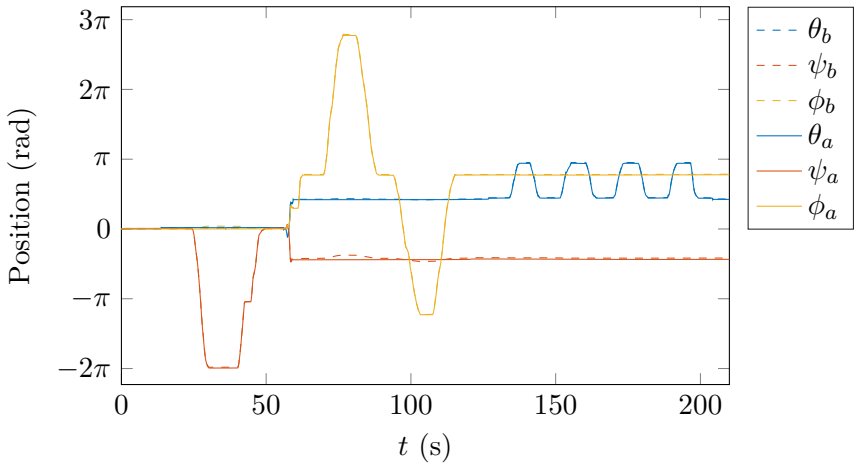


Figure 4.18: Integrated gyroscope scale and misalignment calibration data pre- (dashed) and post-calibration (solid), where $(\theta, \psi, \phi) = \int (p, q, r)\ dt$. Data is acquired by rotating the system exactly about each of the individual body axes by known multiples of $\pi/2$, allowing a comparison of the measured integrals with the known angle of rotation. A gyroscope free from scale and misalignment error would have no error between the measured and integrated angles, and would measure zero rotation on axes that are not in motion. This misalignment error is exampled at $t = 80\,\text{s}$ where a movement in Ω_r couples into Ω_q. This is completely removed by calibration.

ture, largely follow a random walk about zero, and therefore cannot be calibrated offline. Scale and cross-axis misalignment errors, however, are relatively time invariant, and can be measured by rotating the sensor at known angular rates exactly about each of the body frame axes. This requires the use of a rate table, an expensive piece of test equipment, so this could not be performed. Instead, the system is rotated exactly about its body axes whilst recording angular rate data, which is then integrated to provide a measurement of each rotation angle. Comparison of this with knowledge of the exact angle through which the system was rotated allows an estimation of these scale and misalignment error matrices. This experiment is shown in Figure 4.18, with measured angles pre- and post-calibration. A significant misalignment error is visible in the ψ angle, corresponding to the Ω_q gyroscope, and small scale errors are visible in all three sensors. The calibrated data shows these errors have been eliminated, with no visible cross-coupling remaining.

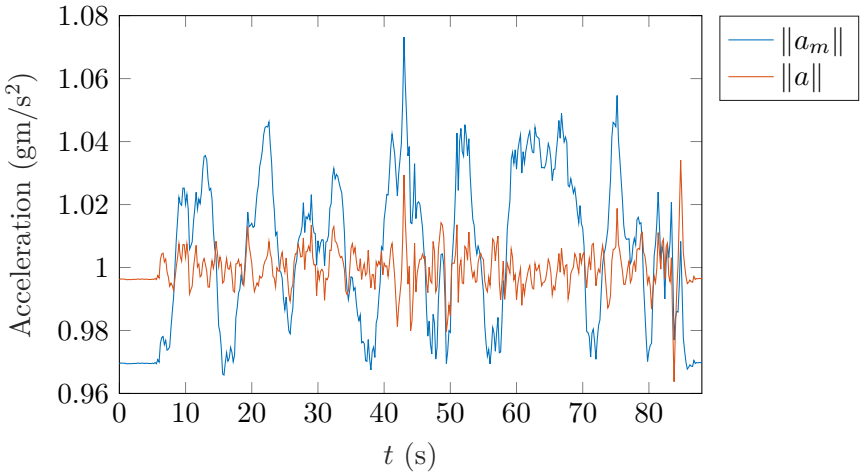


Figure 4.19: The norms of a set of acceleration data pre- (blue) and post-calibration (red). This data was gathered by slowly rotating the platform through a wide range of angles to provide a measurement of g from all orientations, The data is normalised to g and smoothed for readability.

4.5.3 Accelerometers

Three orthogonal accelerometers provide measurements of acceleration along three axes parallel to $\{\hat{b}_x, \hat{b}_y, \hat{b}_z\}$, with the sensor mounted approximately at the overall system center of mass to minimise measurement of acceleration due to angular acceleration of the system. These measure both dynamic acceleration due to movement of the system, as well as the constant acceleration of the gravity vector, defined in B as $\vec{g} = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T$. These measurements are very noisy due to mechanical vibration of the system when in motion, with this noise source modelled as an additive white noise of variance σ_a^2. Accelerometers are also affected by cross-axis misalignment, scale, and bias errors. These errors are relatively time and temperature invariant, allowing a fixed compensation. The measured accelerometer reading $\vec{a}_m$ is therefore mapped to the true reading $\vec{a}$ using a similar linear sensor model

$$\vec{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} 1 & M_{xy} & M_{xz} \\ M_{yx} & 1 & M_{yz} \\ M_{zx} & M_{zy} & 1 \end{bmatrix} \begin{bmatrix} 1/S_x & 0 & 0 \\ 0 & 1/S_y & 0 \\ 0 & 0 & 1/S_z \end{bmatrix} \begin{bmatrix} a_{m,x} - b_x \\ a_{m,y} - b_y \\ a_{m,z} - b_z \end{bmatrix} \tag{4.5.3}$$

Figure 4.19 shows the magnitude of a set of acceleration data gathered by slowly rotating the platform through a wide range of angles, normalised to g. As this data is for a nearly static system this should be mostly just a measurement of gravity, and should therefore be of relatively constant magnitude, which is clearly not the case for the uncalibrated data. The magnitude of the calibrated data remains much closer to 1, with some as variation expected due to minor acceleration of the platform whilst it is rotated.

4.5.4 Omitted Sensors

A number of sensors commonly used in robotics could have been included on this prototype, but were omitted for the following reasons:

- Magnetometers - These allow the measurement of the earth's magnetic field, allowing an absolute measurement of heading. However, when operating inside modern buildings containing large amounts of structural steel the Earth's weak ambient field is often heavily distorted, making these sensors ineffective indoors. It is therefore assumed that a small amount of drift will be present in any derived yaw estimate.
- Computer Vision - Various optical and LIDAR sensing technologies could allow the platform to perform simultaneous location and mapping (SLAM). As the control techniques developed in this thesis do not aim to incorporate obstacle avoidance, the online generation of maps is not vital to this research. Furthermore, it was found that the dead reckoning used here instead exhibited very slow drift, and so is more than adequate for control development and evaluation.
- GPS - This would provide an absolute position measurement that could be used to correct dead reckoning error, but has low accuracy and cannot operate indoors.

4.6 State Estimation

Using the above sensors an estimate of the full system state (p, v) must be produced. By use of the inverse kinematic model (3.1.8) estimates of v_x, v_y, and $\dot{\phi}$ can be produced. This requires the measurement of wheel velocities $\dot{\theta}_i$, which

must be derived from wheel encoder angular position measurements by numerical differentiation, for which there are two obvious options. The first common approach is to measure the time between changes of encoder state, with velocity inversely proportional to Δt as $\dot{\theta}_k = \frac{\text{sgn}(\theta_k - \theta_{k-1})}{\theta_s \Delta t_k}$, where θ_s denotes the size of a single encoder step. This produces a measurement with good accuracy on every change in encoder state for the majority of speeds. However, as speed approaches zero the period between measurement updates increases, to the point where no measurement of velocity is returned for a stationary wheel. This differentiation scheme also tends to amplify high frequency noise, such as that generated by vibration about a quantisation step. Alternatively, instead of performing measurements over a fixed $\Delta\theta_{e,i}$, it is possible to fix Δt to some suitable period, and instead compares the encoder position at the start and end of the measurement window in order to calculate a speed measurement. This provides measurements at a fixed known update rate, and is able to measure a speed of zero. Unfortunately, in order to avoid acting as a low-pass filter, a relatively short value of Δt is required, which through the quantisation of position measurements leads to a coarsely quantised velocity measurement.

For these reasons, these approaches are typically superseded by observer and estimation methods [33]. These methods model wheel velocity as an internal state, which is continuously integrated to give a position estimate that can be compared with the true measurement returned by the encoder. An error in this comparison indicates an incorrect estimate of $\dot{\theta}_i$, which can be used to recursively update $\dot{\theta}_i$ until this error is eliminated. These methods are able to provide estimates at a fast fixed update rate, and are able to provide high resolution estimates even at very low or zero speed.

Rather than directly estimating each of $\dot{\theta}_i$, here body velocities v_x, v_y, and $\dot{\phi}$ are modelled as internal states instead, which can then be integrated with each iteration to give estimates of x, y, and ϕ. By the inverse kinematics model (3.1.8) these body velocities can be uniquely mapped to equivalent wheel velocities, that can in turn be integrated to give an expected change in wheel position over each estimator timestep $\Delta\theta_{ei}$. This can then be compared with the measured difference in wheel position between this and the previous iteration, providing an error signal that can be used to correct the estimate of $(v_x, v_y, \dot{\phi})$.

This gives the prediction model

$$\begin{bmatrix} \widehat{v}_x \\ \widehat{v}_y \\ \widehat{\dot{\phi}} \end{bmatrix}_k = \begin{bmatrix} \widehat{v}_{x,} \\ \widehat{v}_{y,} \\ \dot{\phi} \end{bmatrix}_{k-1} + \vec{w}_k \tag{4.6.1}$$

and measurement model

$$\begin{bmatrix} \tilde{\theta}_{e1} \\ \tilde{\theta}_{e2} \\ \tilde{\theta}_{e3} \\ \tilde{\theta}_{e4} \end{bmatrix}_k = \begin{bmatrix} \theta_{e1} \\ \theta_{e2} \\ \theta_{e3} \\ \theta_{e4} \end{bmatrix}_{k-1} + \frac{1}{r_w} \begin{bmatrix} -\widehat{v}_{x,k}\cot(\alpha_1) - \widehat{v}_{y,k} - \widehat{\phi}_k l_1 - R_w(\Omega_p - b_p) \\ -\widehat{v}_{x,k}\cot(\alpha_2) - \widehat{v}_{y,k} - \widehat{\phi}_k l_2 - R_w(\Omega_p - b_p) \\ -\widehat{v}_{x,k}\cot(\alpha_3) - \widehat{v}_{y,k} - \widehat{\phi}_k l_3 - R_w(\Omega_p - b_p) \\ -\widehat{v}_{x,k}\cot(\alpha_4) - \widehat{v}_{y,k} - \widehat{\phi}_k l_4 - R_w(\Omega_p - b_p) \end{bmatrix} + \vec{v}_k \tag{4.6.2}$$

where $\vec{w}_k$ and $\vec{v}_k$ are process and observation noise vectors with covariances Q_k and R_k, both assumed to be multivariate zero mean Gaussian noise sources. Position estimates can then be derived by numerical integration with timestep δt as

$$\begin{bmatrix} \widehat{x} \\ \widehat{y} \\ \widehat{\phi} \end{bmatrix}_k = \begin{bmatrix} \widehat{x}_{k-1} + \delta t_k \left(\widehat{v}_{x,k-1}\cos(\phi_{k-1}) - \widehat{v}_{y,k-1}\sin(\phi_{k-1})\right) \\ \widehat{y}_{k-1} + \delta t_k \left(\widehat{v}_{x,k-1}\sin(\phi_{k-1}) + \widehat{v}_{y,k-1}\cos(\phi_{k-1})\right) \\ \widehat{\phi}_{k-1} + \delta t_k \dot{\phi}_{k-1} \end{bmatrix} \tag{4.6.3}$$

However, this method is found to yield excessive position estimate drift unless a very short velocity estimator time constant is used, which in turn yields excessive velocity estimate noise. A position estimate with less drift can instead be derived by discrete integration of the inertial frame inverse kinematics model evaluated

at a time-varying ϕ as

$$\begin{bmatrix} x \\ y \\ \phi \\ \theta_p \end{bmatrix}_k = \begin{bmatrix} x \\ y \\ \phi \\ \theta_p \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{\cos(\alpha_1+\phi_{k-1})}{r_w \sin(\alpha_1)} & -\frac{\sin(\alpha_1+\phi_{k-1})}{r_w \sin(\alpha_1)} & -\frac{l_1}{r_w} & -1 \\ \frac{\cos(\alpha_2+\phi_{k-1})}{r_w \sin(\alpha_2)} & -\frac{\sin(\alpha_2+\phi_{k-1})}{r_w \sin(\alpha_2)} & -\frac{l_2}{r_w} & -1 \\ \frac{\cos(\alpha_3+\phi_{k-1})}{r_w \sin(\alpha_3)} & -\frac{\sin(\alpha_3+\phi_{k-1})}{r_w \sin(\alpha_3)} & -\frac{l_3}{r_w} & -1 \\ \frac{\cos(\alpha_4+\phi_{k-1})}{r_w \sin(\alpha_4)} & -\frac{\sin(\alpha_4+\phi_{k-1})}{r_w \sin(\alpha_4)} & -\frac{l_4}{r_w} & -1 \end{bmatrix}^{-1} \cdot \left(\begin{bmatrix} \theta_{e,1} \\ \theta_{e,2} \\ \theta_{e,3} \\ \theta_{e,4} \end{bmatrix}_k - \begin{bmatrix} \theta_{e,1} \\ \theta_{e,2} \\ \theta_{e,3} \\ \theta_{e,4} \end{bmatrix}_{k-1} \right) \tag{4.6.4}$$

Assuming known measurements of ϕ and θ_p, the accuracy of this inversion can be improved, yielding

$$\begin{bmatrix} x \\ y \end{bmatrix}_k = \begin{bmatrix} x \\ y \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{\cos(\alpha_1+\phi_{k-1})}{r_w \sin(\alpha_1)} & -\frac{\sin(\alpha_1+\phi_{k-1})}{r_w \sin(\alpha_1)} \\ \frac{\cos(\alpha_2+\phi_{k-1})}{r_w \sin(\alpha_2)} & -\frac{\sin(\alpha_2+\phi_{k-1})}{r_w \sin(\alpha_2)} \\ \frac{\cos(\alpha_3+\phi_{k-1})}{r_w \sin(\alpha_3)} & -\frac{\sin(\alpha_3+\phi_{k-1})}{r_w \sin(\alpha_3)} \\ \frac{\cos(\alpha_4+\phi_{k-1})}{r_w \sin(\alpha_4)} & -\frac{\sin(\alpha_4+\phi_{k-1})}{r_w \sin(\alpha_4)} \end{bmatrix}^{-1} \cdot \left(\begin{bmatrix} \theta_{e,1} \\ \theta_{e,2} \\ \theta_{e,3} \\ \theta_{e,4} \end{bmatrix}_k - \begin{bmatrix} \theta_{e,1} \\ \theta_{e,2} \\ \theta_{e,3} \\ \theta_{e,4} \end{bmatrix}_{k-1} - \begin{bmatrix} -\frac{l_1}{r_w} & -1 \\ -\frac{l_2}{r_w} & -1 \\ -\frac{l_3}{r_w} & -1 \\ -\frac{l_4}{r_w} & -1 \end{bmatrix} \begin{bmatrix} \phi_k - \phi_{k-1} \\ \theta_p - \theta_{p,k-1} \end{bmatrix} \right) \tag{4.6.5}$$

No absolute position measurement is available on this system, preventing the correction of dead reckoning integration drift. This integration is therefore performed outside of the state estimator to reduce the size of the estimator's state vector.

The wheel encoders and inverse kinematics model alone are not able to provide an estimate of θ_p. Biased measurements of $\dot{\phi}$ and $\dot{\theta}_p$ can be obtained by rotation of Ω by R_{bp}, and it is known that no rotation occurs about $\hat{b}_y$ when assuming

operation on a level surface, allowing the definition of the mapping

$$\begin{bmatrix} \dot{\theta}_p \\ 0 \\ \dot{\phi} \end{bmatrix} = R_{bp} \left(\Omega - \begin{bmatrix} b_p \\ b_q \\ b_r \end{bmatrix} \right) \tag{4.6.6}$$

$$= \begin{bmatrix} \Omega_p - b_p \\ \cos(\theta_p)(\Omega_q - b_q) - \sin(\theta_p)(\Omega_r - b_r) \\ \sin(\theta_p)(\Omega_q - b_q) + \cos(\theta_p)(\Omega_r - b_r) \end{bmatrix} \tag{4.6.7}$$

These measurements of $\dot{\phi}$ and $\dot{\theta}_p$ can be integrated to yield estimates of ϕ and θ_p, giving the expanded prediction model

$$\begin{bmatrix} \widehat{\phi} \\ \widehat{\theta}_p \\ \widehat{v}_x \\ \widehat{v}_y \\ \widehat{b}_p \\ \widehat{b}_q \\ \widehat{b}_r \end{bmatrix}_k = \begin{bmatrix} \widehat{\phi}_{k-1} + \delta t_k \Big(\sin(\theta_{p,k-1})(\Omega_{q,k} - \widehat{b}_{q,k-1}) \\ \qquad + \cos(\theta_{p,k-1})(\Omega_{r,k} - \widehat{b}_{r,k-1}) \Big) \\ \widehat{\theta}_{p,k-1} + \delta t_k (\Omega_p - b_p) \\ \widehat{v}_{x,k-1} \\ \widehat{v}_{y,k-1} \\ \widehat{b}_{p,k-1} \\ \widehat{b}_{q,k-1} \\ \widehat{b}_{r,k-1} \end{bmatrix} + \vec{w}_k \tag{4.6.8}$$

While $\dot{\phi}$ and $\dot{\theta}_p$ could be modelled as internal states to allow for filtering of gyroscope noise, this is instead assumed to be sufficiently performed by the FPGA low-pass filter, leaving a noise-free but biased signal.

Determination of gyroscope biases and elimination of integral drift in the estimates of ϕ and θ_p each require the incorporation of an additional measurement. A measurement of θ_p can be calculated by comparison of the measured body acceleration vector with that expected by a rotation of $\vec{g}$ by R_{bp}^T. From (4.6.6), when $\theta_p = 0$ a measurement of b_q is available, distorted by noise due to non-level terrain. No measurement of ϕ is available to eliminate integral drift when $\theta_p = 0$, and the inverse kinematics derived measurement of $\dot{\phi}$ is found to suffer from a greater time varying bias than Ω_r, so this cannot be used to improve the estimate of b_r. This bias is therefore unobservable, and must be set to a constant value measured while the system is stationary, and so the estimate of ϕ will always be

subject to a slow drift.

This yields the final prediction and measurement models

$$\begin{bmatrix} \widehat{\theta}_p \\ \widehat{v}_x \\ \widehat{v}_y \\ \widehat{b}_p \\ \widehat{b}_q \end{bmatrix}_k = \begin{bmatrix} \widehat{\theta}_{p,k-1} + \delta t_k(\Omega_p - b_p) \\ \widehat{v}_{x,k-1} \\ \widehat{v}_{y,k-1} \\ \widehat{b}_{p,k-1} \\ \widehat{b}_{q,k-1} \end{bmatrix} + \vec{w}_k \tag{4.6.9}$$

$$\begin{bmatrix} \tilde{\theta}_{e1} \\ \tilde{\theta}_{e2} \\ \tilde{\theta}_{e3} \\ \tilde{\theta}_{e4} \\ \tilde{a}_y \\ \tilde{a}_z \\ 0 \end{bmatrix}_k = \begin{bmatrix} \tilde{\theta}_{e1,k-1} + \frac{1}{r_w}(-\widehat{v}_{x,k}\cot(\alpha_1) - \widehat{v}_{y,k} - \widehat{\phi}_k l_1) - (\Omega_{p,k} - b_{p,k}) \\ \tilde{\theta}_{e2,k-1} + \frac{1}{r_w}(-\widehat{v}_{x,k}\cot(\alpha_2) - \widehat{v}_{y,k} - \widehat{\phi}_k l_2) - (\Omega_{p,k} - b_{p,k}) \\ \tilde{\theta}_{e3,k-1} + \frac{1}{r_w}(-\widehat{v}_{x,k}\cot(\alpha_3) - \widehat{v}_{y,k} - \widehat{\phi}_k l_3) - (\Omega_{p,k} - b_{p,k}) \\ \tilde{\theta}_{e4,k-1} + \frac{1}{r_w}(-\widehat{v}_{x,k}\cot(\alpha_4) - \widehat{v}_{y,k} - \widehat{\phi}_k l_4) - (\Omega_{p,k} - b_{p,k}) \\ g\sin(\widehat{\theta}_{p,k}) \\ g\cos(\widehat{\theta}_{p,k}) \\ \cos(\theta_{p,k})(\Omega_{q,k} - b_{q,k}) - \sin(\theta_{p,k})(\Omega_{r,k} - b_r) \end{bmatrix} + \vec{v}_k \tag{4.6.10}$$

The nonlinearity of (4.6.10) necessitates the use of a nonlinear state estimator. In the field of attitude estimation the extended Kalman filter (EKF) is the *de facto* standard [97], making this an obvious choice. The standard EKF equations, omitted here for brevity, are therefore evaluated with every new set of sensor data to iteratively estimate the modelled state. While this estimator has no guarantee of optimality or convergence, in practise this occurs very quickly, even for an initially non-stationary system.

Behaviour of the filter is controlled by selection of the process and measurement covariance matrices Q and R, assumed to be diagonal. The element of Q mapping to $\widehat{\theta}_p$ represents uncertainty due to integration error and any remaining gyroscope noise, so is set to a very small value. The elements of Q that map to the estimates of v_x and v_y control the rate at which these velocities are expected to change. They are therefore selected to provide a suitably quick velocity decay to zero in the absence of a change in wheel position, whilst avoiding the production of a sawtooth shaped estimate when operating at a small constant velocity. Finally, rows mapping Q to $\widehat{b}_p$ and $\widehat{b}_q$ reflect the rate at which these biases are expected to drift, and therefore are made very small so that once a correct estimate is achieved it only changes very slowly.

Rows of R that map to wheel encoder angular positions are chosen to reflect encoder quantization noise and uncertainty in the inverse kinematics model due to imperfect wheel geometry and unmodelled effects such as backlash in the roller mountings and wheel slip. This is therefore chosen by trial and error until desirable body velocity estimates are achieved. Rows of R that map to $\tilde{a}_y$ and $\tilde{a}_z$ partially represent intrinsic sensor noise in the accelerometers, but are mostly dominated by noise due to vibration and dynamic acceleration of the system. These are therefore tuned in tandem with Q to ensure suitable time constants for the estimation of the gyroscope biases and sufficiently fast correction of drift of the estimate of θ_p, whilst avoiding the coupling of any noticeable amount of accelerometer noise into these estimates. These elements are scaled by the factor $1 + e^{K|\|a\|_2 - g|}$, where $K \gg 1$, such that the accelerometer is trusted less when acceleration other than that due to gravity is measured. Finally, the element of R mapping to $\widehat{b}_q$ is chosen to ensure a desirable rate of convergence of $\widehat{b}_q \to b_q$.

Figure 4.20 shows the output of this estimator for a set of experimental data. This experiment starts with the system lying down, with a movement at $t = 3.5\,\text{s}$ to a known upright position. This is maintained until $t = 20\,\text{s}$, before random movements in all states are performed until $t = 68\,\text{s}$, ending at the same location as at $t = 20\,\text{s}$. This shows a total position drift due to dead reckoning error of $0.073\,\text{m}$. This increases to $0.094\,\text{m}$ if ϕ is derived from the inverse kinematics model rather than the gyroscopes. The plot of ϕ contains a second trajectory ϕ_b, representing the ϕ estimate generated when assuming $b_q = 0$, demonstrating the value in estimating this bias. θ_p is seen to quickly converge to the correct estimate from the origin, with all of the integral drift visible in the raw integral of Ω_p eliminated. Gyroscope bias estimates $\widehat{b}_p$ and $\widehat{b}_q$ show mostly monotonically decreasing variances, converging to nearly constant values as intended.

4.7 Conclusion

This chapter has described the reasoning and methodology behind the design and construction of a Collinear Mecanum Drive prototype. Methods for the compensation of undesirable actuation nonlinearities are presented, allowing the assumption minimal input model mismatch. A methodology for the experimental measurement of all model parameters for a given CMD has been demonstrated,

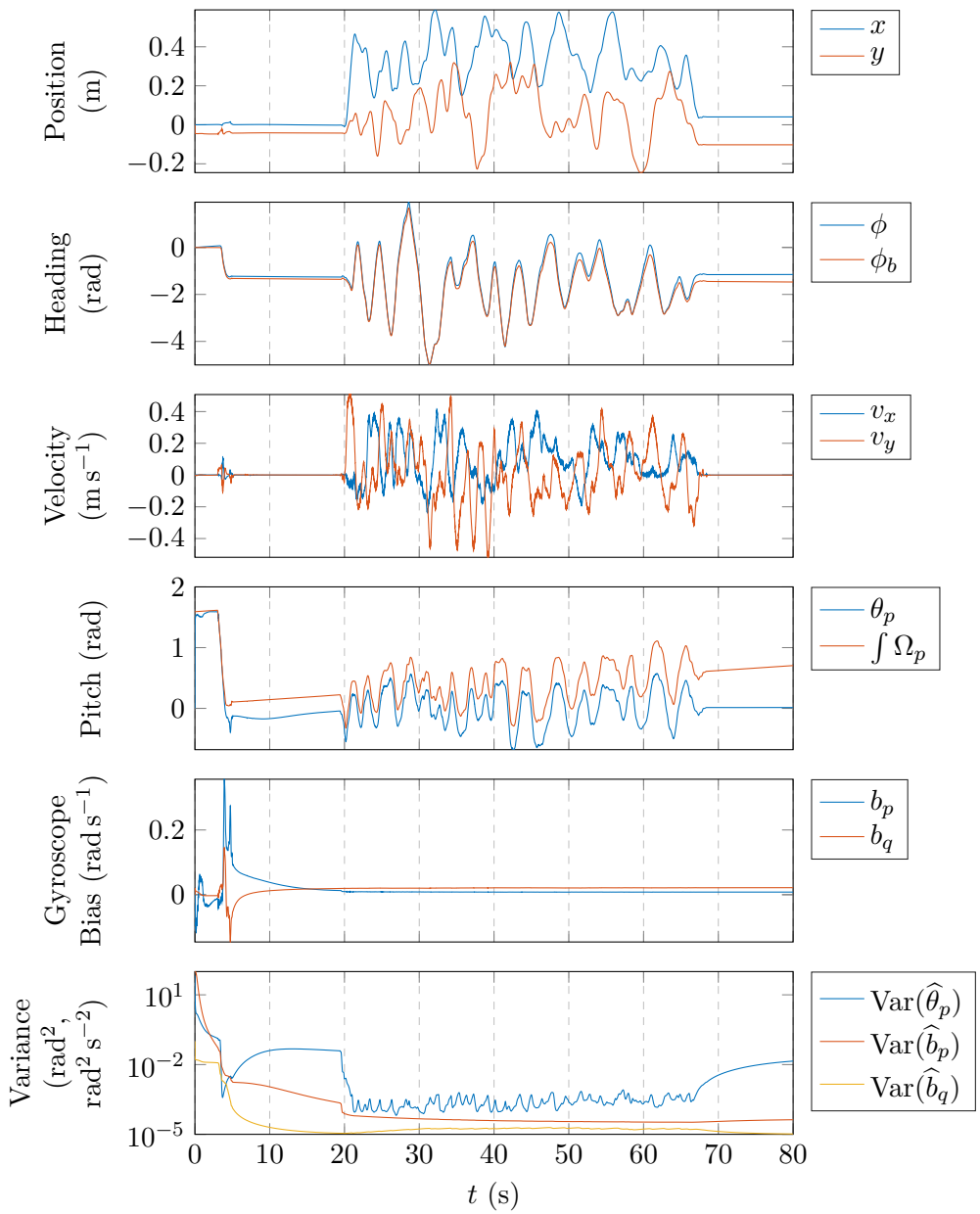


Figure 4.20: State estimation data gathered by moving the system through a range of random trajectories, with the same real-world position and pose maintained at $t = 20\,\mathrm{s}$ and $t \geq 68\,\mathrm{s}$ to allow a measurement of dead reckoning drift. ϕ_b shows an estimate of ϕ generated whilst forcing $b_q \equiv 0$, demonstrating the value of estimating b_q. Similar, plot 4 shows both $\widehat{\theta}_p$ and an estimate achieved by integration of Ω_p, demonstrating rapid divergence due to gyroscope bias and integral drift. All states are initialised at the origin, and the EKF covariance matrix is initialised to $P_0 = I_{5\times5} \times 10^2$ to represent unknown initial conditions.

with the results of this verified by comparison with that expected from a CAD model of the prototype. Finally, the sensor calibration and state estimation methods necessary for the accurate online estimation of the full system state from distorted sensor data has been derived and demonstrated experimentally.

It is therefore assumed that the prototype developed in this chapter accurately fits the model derived in Chapter 3. This will allow for the meaningful experimental evaluation of control and trajectory planning techniques developed in the remainder of this thesis.

Chapter 5

Nonlinear Control

This chapter introduces the first of three control methods explored in this thesis. The CMD is first locally partially feedback linearised by a local state space diffeomorphism and nonlinear feedback, expressing the system dynamics as five linear and three nonlinear ODEs with three new inputs. Three nonlinear controllers are then derived using a backstepping approach, controlling system local body frame velocities, inertial frame body velocities, and global Cartesian position. Lyapunov functions are derived for all three controllers to guarantee stability, and asymptotic convergence to the desired references from a bounded set of initial states is proven. These controllers are evaluated in both simulation and on the experimental prototype.

Feedback linearisation is a procedure by which a nonlinear system can be transformed into an equivalent fully or partially linear system, achieved using a change of control input, along with either a change of state space coordinates, or a transformation of the output [98]. The extent to which a system can be linearised by these methods can be determined by examining the system's relative degree; only systems with a maximum relative degree equal to the size of their state space can be fully linearised by feedback. These methods result in a system that is either partially or fully linear, allowing the application of classical linear control and analysis techniques to a previously nonlinear plant. In the partially linearised case, the remaining nonlinear subsystems can then be controlled using nonlinear control techniques, which is typically an easier task than applying these techniques to the original higher dimensional nonlinear system.

Feedback linearisation of systems with a relative degree of less than n will

yield systems that contain zero dynamics, new states and dynamics that are unobservable from the new outputs, which may be unstable. In practise it can be dangerous for these unobservable states to be allowed to grow unboundedly, so their behaviour must be considered during control design.

These techniques have been applied to various forms of inverted pendulum, such as the single and double cart-pole inverted pendulums [99,100], the reaction wheel inverted pendulum [101], the acrobot [102, 103], and most relevantly, the two-wheeled inverted pendulum [51, 56, 104]. Partial feedback linearisation has also been utilised in the control of quadrotors [105, 106]. These methods have never been applied to a ball-balancing system. As all of these systems are underactuated only partial feedback linearisation is achieved, with nonlinear controllers subsequently designed to control the remaining nonlinear dynamics.

5.1 Partial Feedback Linearisation

In order to facilitate the derivation of a feedback linearising control, the input vector fields of (3.2.44) are first simplified using a change of input $v = P(x)u$ to define a new decoupled input v. It is found that

$$P(x) = \begin{bmatrix} g_{15} & g_{25} & g_{35} \\ g_{17} & g_{27} & g_{37} \\ g_{18} & g_{28} & g_{38} \end{bmatrix} \tag{5.1.1}$$

is nonsingular for $|\theta_p| \lesssim 2.39\,\text{rad}$ using the parameters in Table 4.2, and is therefore invertible under this condition, allowing the definition of the new input vector fields

$$\begin{bmatrix} \tilde{g}_1 \\ \tilde{g}_2 \\ \tilde{g}_3 \end{bmatrix}^T = \begin{bmatrix} & 0_{3\times 4} & \\ 1 & 0 & 0 \\ \tilde{g}_{16} & \tilde{g}_{26} & \tilde{g}_{36} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1.2}$$

where

$$\begin{bmatrix} \tilde{g}_{16} \\ \tilde{g}_{26} \\ \tilde{g}_{36} \end{bmatrix}^T = \begin{bmatrix} g_{16} \\ g_{26} \\ g_{36} \end{bmatrix}^T P^{-1} \tag{5.1.3}$$

in which $(\tilde{g}_{16}, \tilde{g}_{26}) = 0$ for the parameters in Table 4.2, and $\tilde{g}_{36}$ is scalar valued function that is again smooth for $|\theta_p| \lesssim 2.39\,\text{rad}$. The $\dot{x}_5$, $\dot{x}_7$, and $\dot{x}_8$ subsystems can then be linearised by the feedback

$$\begin{aligned} v_1 &= w_1 - f_5(x) \\ v_2 &= w_2 - f_7(x) \\ v_3 &= w_3 - f_8(x) \end{aligned} \tag{5.1.4}$$

in which $w = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}^T$ is used as the new input, yielding the new drift and unchanged input vector fields

$$\tilde{f}(x) = \begin{bmatrix} \cos(x_3)x_5 - \sin(x_3)x_6 \\ \sin(x_3)x_5 + \cos(x_3)x_6 \\ x_7 \\ x_8 \\ 0 \\ f_6 - \tilde{g}_{16}(x_4)f_5(x) - \tilde{g}_{26}(x_4)f_7(x) - \tilde{g}_{36}(x_4)f_8(x) \\ 0 \\ 0 \end{bmatrix} \tag{5.1.5}$$

$$\begin{bmatrix} \tilde{g}_1(x_4) \\ \tilde{g}_2(x_4) \\ \tilde{g}_3(x_4) \end{bmatrix}^T = \begin{bmatrix} 0_{1\times4} & 0_{1\times4} & 0_{1\times4} \\ 1 & 0 & 0 \\ \tilde{g}_{16}(x_4) & \tilde{g}_{26}(x_4) & \tilde{g}_{36}(x_4) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1.6}$$

In order for the coordinates x to fully span $\mathbb{R}^8$ they must be linearly independent, meaning their gradients $\dot{x}$ must be linearly independent of one another [89]. Clearly in (5.1.5) this property has been lost, as $\dot{x}_6$ is now a linear function of $\dot{x}_5$, $\dot{x}_7$, and $\dot{x}_8$. A state transformation $\mathrm{T} : x \to z$ is therefore required to transform x into some new set of linearly independent coordinates z as $z = \mathrm{T}(x)$. As $\dot{x}_i$ for $i = [1 \,..\, 5, 7, 8]$ are already linearly independent, these can be mapped as $z_i = x_i$.

As $w \equiv \begin{bmatrix} \dot{x}_5 & \dot{x}_7 & \dot{x}_8 \end{bmatrix}^T$, it is required that

$$\frac{\partial z_6}{\partial x} \begin{bmatrix} \tilde{g}_1(x_4) \\ \tilde{g}_2(x_4) \\ \tilde{g}_3(x_4) \end{bmatrix}^T = 0 \tag{5.1.7}$$

Writing $\nabla z_6 = \begin{bmatrix} \alpha_1 & \dots & \alpha_8 \end{bmatrix}$, by (5.1.7) it is implied that

$$\begin{aligned} \alpha_5 + \alpha_6 \tilde{g}_{16} &= 0 \\ \alpha_7 + \alpha_6 \tilde{g}_{26} &= 0 \\ \alpha_8 + \alpha_6 \tilde{g}_{36} &= 0 \end{aligned} \tag{5.1.8}$$

which is satisfied for

$$\begin{aligned} \alpha_5 &= -\lambda \tilde{g}_{16} \\ \alpha_6 &= \lambda \\ \alpha_7 &= -\lambda \tilde{g}_{26} \\ \alpha_8 &= -\lambda \tilde{g}_{36} \end{aligned} \tag{5.1.9}$$

Choosing $\lambda = 1$, z_6 can be defined as

$$z_6 = x_6 - x_5 \tilde{g}_{16} - x_7 \tilde{g}_{26} - x_8 \tilde{g}_{36} \tag{5.1.10}$$

defining the transformation $\mathrm{T}(x)$ as

$$z = \mathrm{T}(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 - x_5 \hat{g}_{16} - x_7 \hat{g}_{26} - x_8 \hat{g}_{36} \\ x_7 \\ x_8 \end{bmatrix} \tag{5.1.11}$$

For the parameters in Table 4.2

$$\det\left(\frac{\partial \mathrm{T}(x)}{\partial x}\right) \neq 0 \ \forall \ \{x \in \mathbb{R}^8 \mid x_4 \bmod 2\pi \neq \pm 2.39\,\mathrm{rad}\} \tag{5.1.12}$$

therefore the Jacobian of T is locally invertible, meaning T is a local diffeomorphism under this condition [89], with an inverse mapping $x = \mathrm{T}^{-1}(z)$. Taking the differential of $\mathrm{T}(x)$ w.r.t. time allows $\dot{z}$ to be expressed in terms of x as

$$\begin{aligned} \dot{z}_i &= \dot{x}_i, \quad i = [1, \ldots, 5, 7, 8] \\ \dot{z}_6 &= -\dot{x}_5\tilde{g}_{16} - x_5\frac{\partial \tilde{g}_{16}}{\partial x_4} - \dot{x}_7\tilde{g}_{26} - x_7\frac{\partial \tilde{g}_{26}}{\partial x_4} - \dot{x}_8\tilde{g}_{36} - x_8\frac{\partial \tilde{g}_{36}}{\partial x_4} \end{aligned} \tag{5.1.13}$$

which when substituted with differentials of x from (5.1.5) yields the new set of dynamic equations

$$\dot{z}_1 = \cos(z_3)z_5 - \sin(z_3)(z_6 + z_5\tilde{g}_{16}(z) + z_7\tilde{g}_{26}(z) + z_8\tilde{g}_{36}(z)) \tag{5.1.14}$$

$$\dot{z}_2 = \sin(z_3)z_5 + \cos(z_3)(z_6 + z_5\tilde{g}_{16}(z) + z_7\tilde{g}_{26}(z) + z_8\tilde{g}_{36}(z)) \tag{5.1.15}$$

$$\dot{z}_3 = z_7 \tag{5.1.16}$$

$$\dot{z}_4 = z_8 \tag{5.1.17}$$

$$\dot{z}_5 = w_1 \tag{5.1.18}$$

$$\begin{aligned} \dot{z}_6 = f_6(z) - \tilde{g}_{16}(z_4)f_5(z) - \tilde{g}_{26}(z_4)f_7(z) - \tilde{g}_{36}(z_4)f_8(z) \\ - z_5\frac{\partial \tilde{g}_{16}(z)}{\partial z_4} - z_7\frac{\partial \tilde{g}_{26}(z)}{\partial z_4} - z_8\frac{\partial \tilde{g}_{36}(z)}{\partial z_4} \end{aligned} \tag{5.1.19}$$

$$\dot{z}_7 = w_2 \tag{5.1.20}$$

$$\dot{z}_8 = w_3 \tag{5.1.21}$$

where all $f(x)$ and $\tilde{g}(x)$ have been rewritten in terms of z using $x = \mathrm{T}^{-1}(z)$. Under this state transformation and feedback it is evident that w has been eliminated from the expression for $\dot{z}_6$, meaning $\dot{z}_6$, $\dot{z}_1$, and $\dot{z}_2$ now represent internal dynamics, and in which $\dot{z}_5$, $\dot{z}_7$, and $\dot{z}_8$ are now independent of the drift vector, and are linear and decoupled in the new input w.

The internal dynamics (5.1.19) are found to contain zeroth to second time derivatives of θ_p, and cannot be integrated to eliminate either of these velocity or acceleration terms. This expression therefore forms a second order nonholonomic constraint, also referred to as a dynamic constraint.

Examining the zero dynamics found by setting $w = z_5 = z_7 = z_8 = 0$ in $(\dot{z}_1, \dot{z}_2, \dot{z}_6)$, and whilst assuming defined roller angles, wheel spacing symmetry,

and zero friction for model simplification, yields

$$\dot{z}_1 = -z_6 \sin(z_3) \tag{5.1.22}$$

$$\dot{z}_2 = z_6 \cos(z_3) \tag{5.1.23}$$

$$\dot{z}_6 = -\frac{g h_p m_p r_w \sin(z_4)}{4I_{wx} + m_p r_w^2 + 4m_w r_w^2 + h_p m_p r_w \cos(z_4)} \tag{5.1.24}$$

in which it is clear that the zero dynamics do not have a stable equilibrium for $z_4 \neq 0$, and so the system is non-minimum phase [107].

To summarise, through input transformation, coordinate transformation, and nonlinear feedback, the nonlinear system (3.2.35) has been transformed into an equivalent system of five linear and three nonlinear ODEs, with new input $w \equiv \begin{bmatrix} \dot{v}_x & \ddot{\phi} & \ddot{\theta}_p \end{bmatrix}^T$. Actual motor torques are retrieved by the mapping $w \to v \to u \to \tau$. The simulated response of this system to a 0.25 Hz square wave input of unit amplitude to each of w is shown in Figure 5.1, demonstrating the correct linear response of the feedback linearised subsystems v_x, $\dot{\phi}$, and $\dot{\theta}_p$, and unbounded growth of v_y as expected.

5.2 Nonlinear Control of the Partially Feedback Linearised CMD

Numerous approaches exist for the control of the remaining nonlinear dynamics. The simplest methods approximate the nonlinear dynamics by linearisation about a desired operating point [104], in this case the unstable upright equilibrium. This typically yields only a small region about the operating point in which desirable behaviour is achieved, and does not lend itself to straightforward stability analysis of the closed-loop nonlinear system. This region of operation can be extended using gain scheduling, in which the system is linearised about multiple operating points distributed about an operating region of interest within the state space, from which linear controllers relevant to each subset of the operating region can be derived. A switching mechanism then selects the relevant controller based on the current state. While improving performance, the required quantity of controllers increases exponentially with system order, making application to complex systems such as that in this thesis challenging [108]. Again, this method does not facilitate the derivation of a stability proof for the full nonlinear system.

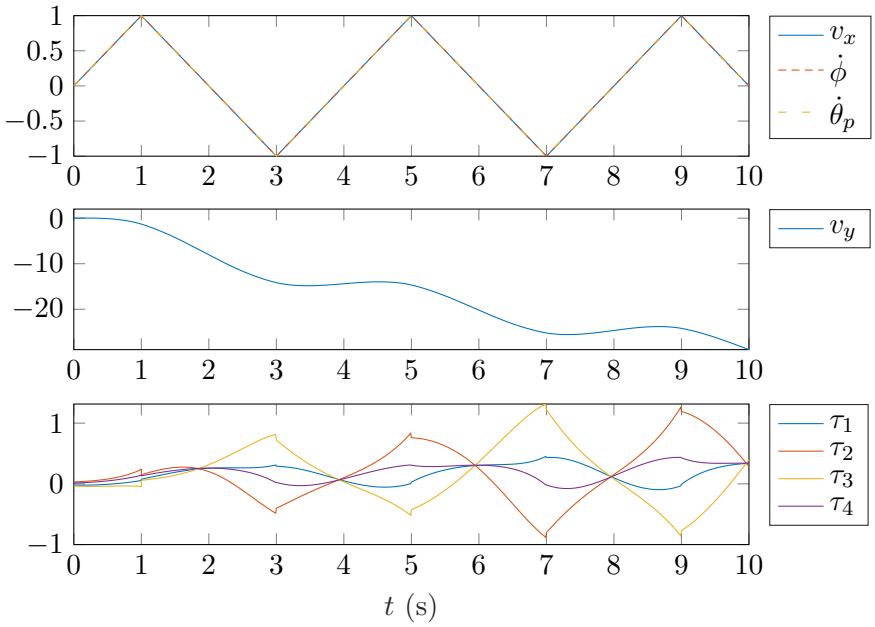


Figure 5.1: Simulated state trajectories of the partially feedback linearised CMD, initialised at the origin and with each of w driven by a 0.25 Hz square wave of unit amplitude. This demonstrates the expected triangular velocity profile in the v_x, $\dot{\phi}$, and $\dot{\theta}_p$ states, while v_y grows unboundedly. τ remains defined throughout the simulation, as the singularity in (5.1.2) is avoided.

Pathak controls the remaining nonlinear dynamics using a backstepping approach [51]. In this method a cascade nonlinear system is controlled by recursively stabilising each subsystem whilst 'stepping back' through the cascaded subsystems. This stabilisation is usually performed by deriving nonlinear controllers that allows each closed-loop subsystem to be formulated as a Lyapunov function, yielding an overall controlled system with stability and convergence guarantees for the full nonlinear dynamics.

A second common approach to the control of these remaining nonlinear subsystems is sliding mode control, which has been applied to the feedback linearised two-wheeled inverted pendulum [58], quadrotors [109], and an array of other forms of inverted pendulum [110]. The main advantage of sliding mode control is its robustness to modelling error and disturbance, and as a bang-bang controller sliding mode control is equivalent to time-optimal control for a number of systems. This can, however, introduce switching chatter into the control signal,

increasing energy usage and generating noise.

It is often necessary, as in this application, to enforce input and state constraints within the controller. In backstepping control these can be incorporated using Lyapunov barrier functions [111], scalar functions in which a unique minimum is attained at the desired steady state, and which tend to infinity as the constraint is approached. This allows an embedding of constraints directly into the control law, whilst retaining a stability proof for the closed-loop system.

In this chapter the former approach is chosen, opting to control the remaining nonlinear subsystems by use the use of a backstepping approach. Three separate controllers are to be derived, starting with the control of local frame body velocities. This controller is relevant to applications where the CMD is to be 'driven' by a user, such as when operating as a personal mobility vehicle or teleoperated platform. The second approach controls velocities in the fixed inertial frame, and the latter controls the position of the CMD in the fixed inertial frame. These are more useful in situations where the CMD is to operate autonomously, such as when moving between waypoints as to navigate an environment. In providing Lyapunov stability guarantees for the full nonlinear closed-loop system, these controllers can be viewed as the safety-critical control option within this thesis, albeit without a guarantee of robustness to model uncertainty.

5.3 Backstepping Control of Local Body Frame Velocities

This controller is required to drive the system local body frame velocities $(v_x, v_y, \dot{\phi})$ to setpoints $(v_{xr}, v_{yr}, \dot{\phi}_r)$. This must be performed whilst bounding deviation of θ_p from zero as to avoid attempting to translate using slip-inducing lean angles, and accelerations $\dot{v}_x$ and $\ddot{\phi}$ must be bounded to again avoid inducing wheel slip. By using a backstepping design approach each subsystem can be proven to be stable by ensuring it can be formulated as a Lyapunov function following loop closure. With appropriate embedding of constraints and bounding of reference inputs this allows the design of a controller with an asymptotic stability guarantee for the full bounded set of references and states.

Control is to be split into two layers. Aggressive control of the θ_p state is desired to provide resistance to disturbances, especially those generated by varying

friction forces when translating in the $\hat{b}_x$ direction, so the linear controller

$$w_3 = -K_{\dot{\theta}_p}\dot{\theta}_p - K_{\theta_p}(\theta_p - \theta_{pr}) \tag{5.3.1}$$

with suitable gains $K_{\dot{\theta}_p}$ and K_{θ_p} is used to provide global exponential convergence of $\theta_p \to \theta_{pr}$, where θ_{pr} represents a new internal reference signal. As this subsystem has relatively fast dynamics, and as low-noise measurements of $\dot{\theta}_p$ and θ_p are available, high gains can be used to allow rapid convergence in the region of 100 ms for large step changes. While linear controllers could also be used to control the feedback linearised v_x and $\dot{\phi}$ subsystems, these are instead to be controlled by the outer loop as to allow the embedding of acceleration constraint enforcement.

The goal of the outer controller is to generate w_1, w_2, and θ_{pr} trajectories that result in convergence of $(v_x, v_y, \dot{\phi}) \to (v_{xr}, v_{yr}, \dot{\phi}_r)$ within finite time. Unlike a TWIP, cross coupling between the (θ_p, v_y) subsystem and the v_x and $\dot{\phi}$ subsystems, for example acceleration forces acting on θ_p when $v_x\dot{\phi} \neq 0$, means that $\theta_{pr} \not\to 0$ may be required for $\dot{v}_y \to 0$ in steady state.

From (5.1.5), acceleration $\dot{v}_y$ can be expressed as

$$\begin{aligned}\dot{v}_y = f_{\dot{v}_y}(x, w) = f_6(x) - \hat{g}_{16}(x)f_5(x) - \hat{g}_{26}(x)f_7(x) - \hat{g}_{36}(x)f_8(x) \\ + \hat{g}_{16}(x)w_1 + \hat{g}_{26}(x)w_2 + \hat{g}_{36}(x)w_3\end{aligned} \tag{5.3.2}$$

This is a complex expression for which it is difficult to analyse the effect of parameter choice, so this is instead substituted with the parameters in Table 4.2, with the assumption that the properties of this function are unlikely to significantly change over realistic ranges of parameter variation. This yields an expression of the form

$$\begin{aligned}f_{\dot{v}_y}(x, w) = aw_3 - v_x\dot{\phi} - b\sin(\theta_p)\dot{\phi}^2 \\ + \frac{cw_3 - dv_y + \sin(\theta_p)\left(e\dot{\phi}^2 - f\dot{\theta}_p^2 - g\right) + hv_x\dot{\phi}}{\cos(\theta_p) + i}\end{aligned} \tag{5.3.3}$$

where a to i denote time invariant constants, and where $0 < e \ll \{a, b, f, h\} \ll \{c, d, i\} \ll g$, in which the operator $\ll$ denotes a difference of approximately an order of magnitude. For the prototype's parameters given in Table 4.2 these

coefficients evaluate to $a = 0.03$, $b = 0.072$, $c = 0.20$, $d = 0.13$, $e = 0.0091$, $f = 0.030$, $g = 9.8$, $h = 0.038$, and $i = 0.54$. For comparison the same analysis of coefficients is performed for a taller and heavier system with $h_p = 1$, $m_p = 20$, $I_{px} = 20$, yielding coefficients of approximate magnitude $0 < h \ll \{d, e\} \ll \{a, f, i\} \ll \{b, c\} \ll g$. Importantly, while some coefficients change in magnitude relative to one another, the constant g remains significantly larger than all other coefficients.

Equation (5.3.3) has no analytical solution for θ_p. However, arranging it into the form $0 = f(x, w, \dot{v}_y)$ and examining $f(x, w, \dot{v}_y)$ for $\theta_p = \pm\pi/2$ yields

$$f(x, w, \dot{v}_y) = \frac{a}{i} w_3 - \frac{d}{i} v_y - \dot{v}_y - \left(1 - \frac{h}{i}\right) \dot{\phi} v_x \mp \left(b - \frac{e}{i}\right) \dot{\phi}^2 \mp \frac{f}{i} \dot{\theta}_p^2 \mp \frac{g}{i} \qquad \text{for } \theta_p = \pm\pi/2 \quad (5.3.4)$$

Equation (5.3.3) is a continuous smooth function over the interval

$$\theta_p \in (-\cos^{-1}(-i), \cos^{-1}(-i)) \quad (5.3.5)$$

where $\cos^{-1}(-i) = 2.14\,\text{rad}$ for the prototype parameters in Table 4.2. By the intermediate value theorem as long as for a given $\{x, w_3, \dot{v}_y\} \in \mathbb{R}^7 \times \mathbb{R} \times \mathbb{R}$ these two expressions are of opposite sign, there must exist some intermediate value of θ_p for which $f(x, w, \dot{v}_y) = 0$, i.e. a solution to (5.3.3) must exist. This condition is necessary for there to exist an inverse function $\theta_p = f_{\dot{v}_y}^{-1}(x, w_3, \dot{v}_y)$ that can be used to determine the lean angle required to achieve a given $\dot{v}_y$ for some state x and input w, though the existence of this inverse also requires a unique mapping. The condition under which at least one solution exists can be written as

$$\left| a w_3 - d v_y - i \dot{v}_y - (i - h) \dot{\phi} v_x \right| \leq (bi - e) \dot{\phi}^2 + f \dot{\theta}_p^2 + g \qquad |\theta_p| \leq \pi/2 \quad (5.3.6)$$

It is apparent that the large constant term g on the rhs means this inequality is satisfied for a large set of accelerations $\dot{v}_y$ and w_3, of which the origin is contained strictly within the interior, provided the $\dot{\phi} v_x$ and v_y terms are not driven excessively large. Satisfaction of this condition can therefore be guaranteed by suitably bounding the user reference inputs $\dot{v}_{xr}$ and $\dot{\phi}_r$, whilst ensuring controller selection as to apply a suitable bound to w_3 and $\dot{v}_y$. While technically a larger

feasible set could be achieved by allowing $|\theta_p| < \cos^{-1}(i)$, as this requires intersection with the pendulum CoM and the ground this bound on θ_p is sensible, and simplifies analysis.

It is found that

$$\frac{\partial \dot{v}_y}{\partial \theta_p} = -\frac{\cos(\theta_p)\left(-e\dot{\phi}^2 + f\dot{\theta}_p^2 + g\right)}{i + \cos(\theta_p)} - b\dot{\phi}^2\cos(\theta_p) - \frac{\sin(\theta_p)^2\left(-e\dot{\phi}^2 + f\dot{\theta}_p^2 + g\right) - cw_3 + dv_y - h\dot{\phi}v_x}{(i + \cos(\theta_p))^2} \tag{5.3.7}$$

from which it is apparent that $\frac{\partial \dot{v}_y}{\partial \theta_p} < 0 \;\forall\; \theta_p \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ under the condition

$$\frac{\cos(\theta_p)\left(f\dot{\theta}_p^2 + g\right)}{i + \cos(\theta_p)} + b\dot{\phi}^2\cos(\theta_p) + \frac{\sin(\theta_p)^2\left(f\dot{\theta}_p^2 + g\right) + dv_y}{(i + \cos(\theta_p))^2} > \frac{\cos(\theta_p)e\dot{\phi}^2}{i + \cos(\theta_p)} + \frac{\sin(\theta_p)^2 e\dot{\phi}^2 + cw_3 + h\dot{\phi}v_x}{(i + \cos(\theta_p))^2} \tag{5.3.8}$$

Again, by virtue of g being very large compared to e, c, and h, this inequality is satisfied for a large region of states centered about the origin. By ensuring $\frac{\partial \dot{v}_y}{\partial \theta_p} < 0$ (5.1.5) is monotonic in θ_p, and therefore the solution to $f^{-1}_{\dot{v}_y}(x, w, \dot{v}_y)$ is guaranteed to be unique. The inverse function $f^{-1}_{\dot{v}_y}(x, w, \dot{v}_y)$ is therefore guaranteed to exist for $\theta_p \in \left[-\pi/2, \pi/2\right]$ under conditions (5.3.4) and (5.3.8). $f^{-1}_{\dot{v}_{y,ss}}(x, \dot{v}_y)$ can be efficiently solved using the Newton-Raphson method with an analytically derived Jacobian, yielding solutions in 5 µs when executing within MATLAB.

As a demonstration of the typical shape of this function, Figure 5.2 shows $\dot{v}_y$ over $\theta_p \in [-2.14, 2.14]$ for values of x and w taken independently from a zero mean normal distribution of standard deviation 3, using model parameters from Table 4.2. This shows how a solution of θ_p for $\dot{v}_y = 0$ does not exist within the interval $(-\pi/2, \pi/2)$ when condition (5.3.4) is violated.

Steady state acceleration $\dot{v}_{y,ss}$ for a given steady state value of θ_p can be found by substituting (5.3.2) with $w_3 = \dot{\theta}_p = 0$, yielding

$$f_{\dot{v}_{y,ss}}(x, w) = -v_x\dot{\phi} - b\sin(\theta_p)\dot{\phi}^2 + \frac{-dv_y + \sin(\theta_p)\left(e\dot{\phi}^2 - g\right) + hv_x\dot{\phi}}{\cos(\theta_p) + i} \tag{5.3.9}$$

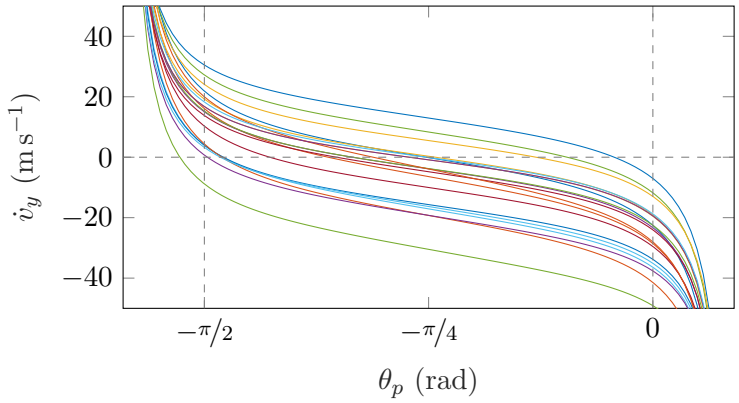


Figure 5.2: Forward acceleration $\dot{v}_y$ over $\theta_p = [-2.14, 2.14]$ for 20 samples of x and w_3 taken from a zero mean normal distribution of standard deviation 3. Note how some curves do not intersect $\dot{v}_y = 0$ within $\theta_p \in [-\pi/2, \pi/2]$, meaning a steady state value of v_y cannot be achieved for these particular choices of x and w. The monotonicity of (5.3.2) demonstrated in (5.3.7) is clearly visible in all curves.

with solutions to $f^{-1}_{\dot{v}_{y,ss}}(x, \dot{v}_y)$ for $\dot{v}_y = 0$ existing within the set bounded by

$$\left| -dv_y - (i-h)\,\dot{\phi} v_x \right| \leq (bi - e)\,\dot{\phi}^2 + g \qquad |\theta_p| < \pi/2 \tag{5.3.10}$$

From here on $f^{-1}_{\dot{v}_{y,ss}}(x, \dot{v}_y)$ shall be written as $f^{-1}_{\dot{v}_{y,ss}}(\dot{v}_y)$ for brevity.

Remark 2. *Absence of oddness property of $f_{\dot{v}_{y,ss}}(x, w)$ in θ_p*

In Pathak's [51] backstepping control of a TWIP it is shown that the TWIP's expression for steady state acceleration $f_{\dot{v}_{y,ss}}(x, w)$ is odd in θ_p, such that θ_p $f_{\dot{v}_{y,ss}}(x, w) \geq 0 \;\forall\; \theta_p \in [-\pi, \pi]$. This property requires the assumption that $\dot{\phi} = 0$. In order for any stability proof that relies on this oddness property to remain valid, such as that demonstrated by this author, it is therefore necessary for the system to perform control of ϕ and v_y separately such that $\dot{\phi}\dot{\theta}_p = 0$. The system in this thesis is required to perform these movements simultaneously, invalidating this assumption. Also, the function $f_{\dot{v}_{y,ss}}(x, w)$ for this system contains two significant even terms $v_x\dot{\phi}$. This oddness property therefore does not extend to this system and so cannot be exploited for Lyapunov function derivation, necessitating a different approach to that used by Pathak [51].

Remark 3. *Velocity equilibria in the local body frame*

For this controller it is desired for the system's local frame body velocities to converge to some user controlled reference velocities v_{xr}, v_{yr}, and $\dot{\phi}_r$, with no interest in position states other than θ_p. Defining the reduced state vector $\tilde{x} = \begin{bmatrix}\theta_p & v_x & v_y & \dot{\phi} & \dot{\theta}_p\end{bmatrix}^T$ and examining $\dot{\tilde{x}} = 0$ in (5.1.5) shows these equilibria exist at any $w = 0$, $\dot{\theta}_p = 0$, $\{v_x, v_y, \dot{\phi}\} \in \mathcal{A}_{ss}$, where $\mathcal{A}_{ss}$ is defined as the set of $\{v_x, v_y, \dot{\phi}\}$ for which solutions to $f_{\dot{v}_{y,ss}}(x) = 0$ exist. A cross-section of this set is shown in Figure 5.3, taken through v_x and $\dot{\phi}$ for $v_y = 0$, with parameters from Table 4.2. Note while a v_y term does feature in $f_{\dot{v}_{y,ss}}$, it vanishes when friction is negated and has a very small coefficient, and so does not represent significant dynamics. This cross-section is therefore relatively invariant in v_y, and in reality a sufficiently large $v_y\dot{\phi}$ term would result in rotation of the system about $\hat{b}_y$ and a subsequent loss of traction long before the shape of this figure is significantly altered.

To summarise, a steady state equilibrium can be obtained for any $\{v_x, v_y, \dot{\phi}\} \in \mathcal{A}_{ss}$, where $\mathcal{A}_{ss}$ is a large set centered about the origin. It is therefore feasible for this controller to achieve $\{v_x, v_y, \dot{\phi}\} \to \{v_{xr}, v_{yr}, \dot{\phi}_r\}$ as $t \to \infty$, i.e. asymptotic tracking of local body frame velocity references is feasible.

With the θ_p subsystem globally asymptotically stabilised by linear feedback, the outer loop is required to generate suitable θ_{pr}, w_1, and w_2 trajectories that yield asymptotic convergence of $(v_x, v_y, \dot{\phi}) \to (v_{xr}, v_{yr}, \dot{\phi}_r)$. These subsystems have substantially slower dynamics than the θ_p subsystem, allowing the assumption that the linear inner loop has converged, i.e. $\theta_p = \theta_{pr}$ and $\dot{\theta}_p = w_3 = 0$. While the v_x and $\dot{\phi}$ subsystems have been rendered linear by feedback linearisation, a nonlinear controller is still used in order to allow the embedded enforcement of the constraints $|w_1| \leq \overline{w}_1$ and $|w_2| \leq \overline{w}_2$. These constraints act to make the resulting control laws favour smooth steady accelerations over aggressive acceleration impulses during a step reference change, and can therefore be used to alleviate the risk of wheel slip by acting as an analogue for a wheel torque constraint.

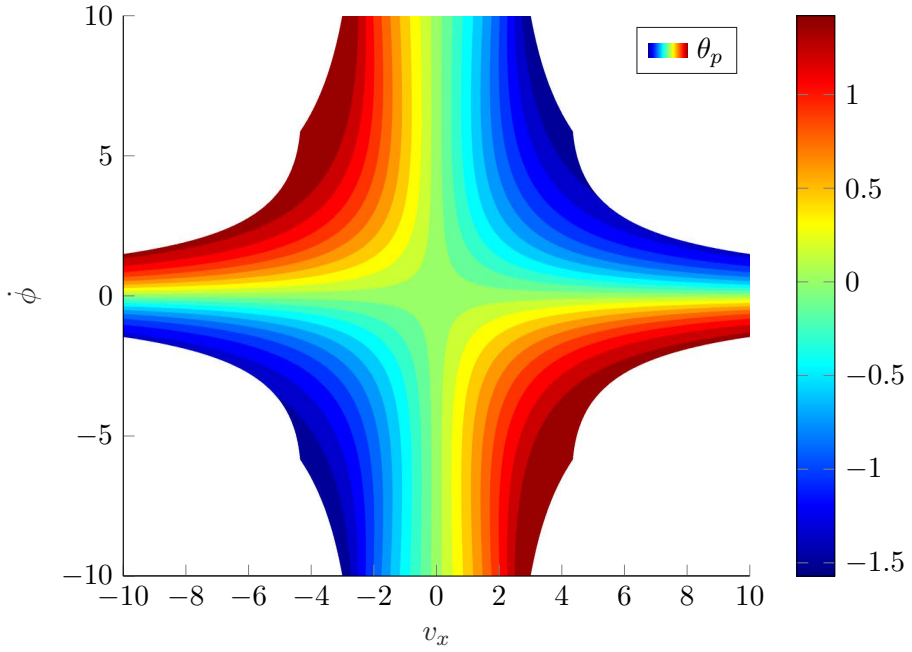


Figure 5.3: A cross section of $\mathcal{A}_{ss}$ through v_x and $\dot{\phi}$ for $\dot{v}_{y,ss} = 0$, $v_y = 0$, with colour encoding the $\theta_{p,ss}$ dimension of $\mathcal{A}_{ss}$. The accessible acceleration space under a lean angle constraint $|\theta_p| \leq \overline{\theta}_p$ can be examined by considering a subset of this space.

Consider the Lyapunov function candidate

$$V_\Sigma = \frac{\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(0)\right)^2}{2\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)} + \frac{K_v\left((v_{xr} - v_x)^2 + (v_{yr} - v_y)^2\right)}{2} + \frac{K_{\dot{\phi}}(\dot{\phi}_r - \dot{\phi})^2}{2} + \frac{1}{2\left(\overline{w}_1^2 - w_1^2\right)} + \frac{1}{2\left(\overline{\ddot{\phi}}^2 - w_2^2\right)} \quad (5.3.11)$$

with time derivative

$$\dot{V}_\Sigma = \frac{\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(0)\right)\left(\dot{f}^{-1}_{\dot{v}_{y,ss}}(0)\left(\overline{\theta}_p^2 - \theta_{pr}^2\right) + \dot{\theta}_{pr}\left(f^{-1}_{\dot{v}_{y,ss}}(0)\theta_{pr} - \overline{\theta}_p^2\right)\right)}{\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)^2}$$
$$- K_v\left(w_1(v_{xr} - v_x) + \dot{v}_y(v_{yr} - v_y)\right) - K_{\dot{\phi}} w_2(\dot{\phi}_r - \dot{\phi})$$
$$+ \frac{w_1\dot{w}_1}{\left(\overline{w}_1^2 - w_1^2\right)^2} + \frac{w_2\dot{w}_2}{\left(\overline{\dot{\phi}}^2 - w_2^2\right)^2} \quad (5.3.12)$$

The first term of (5.3.11) has a unique minimum at $f^{-1}_{\dot{v}_{y,ss}}(0) = \theta_{pr}$, i.e. it is minimised when θ_{pr} has converged to the steady-state lean angle $\theta_{p,ss}$ required to maintain $\dot{v}_y = 0$, found by solution of $\theta_{p,ss} = f^{-1}_{\dot{v}_{y,ss}}(0)$, whilst tending to infinity as $\theta_{pr} \to \pm\overline{\theta}_{pr}$, meaning θ_{pr} remains bounded. This barrier function is exampled in Figure 5.4. The second term of (5.3.12) has a unique minimum at $v_y = v_{yr}$, $v_x = v_{xr}$, with a quadratic cost on deviation from this minimum. Similarly, the third term imposes a quadratic cost on deviation of $\dot{\phi}$ from $\dot{\phi}_r$. The final two terms act as barrier functions to enforce $|w_1| \leq \overline{w}_1$ and $|w_2| \leq \overline{w}_2$, with minimums at $w_1 = 0$ and $w_2 = 0$. V_Σ is therefore globally positive semidefinite when constraints are satisfied and $\left(K_v, K_{\dot{\phi}}\right) > 0$, i.e. $V_\Sigma \geq 0$, has a single unique minimum, and by inspection is radially unbounded for states within the constrained set, but is bounded for $\theta_{pr} \to \infty$ as the first term of (5.3.11) converges to 1. As these conditions are not met for states outside of the constraints, care must be taken to initialise the system with constraints satisfied, i.e. the controller is not able to recover from a constraint violation. However, as the constrained signals exist purely internally this will never occur. The inverse function $\dot{f}^{-1}_{\dot{v}_{y,ss}}(0)$ in (5.3.12) can be calculated as

$$\dot{f}^{-1}_{\dot{v}_{y,ss}}(0) = \left.\frac{df_{\dot{v}_{y,ss}}}{dt}\right|_{\theta_p = f^{-1}_{\dot{v}_{y,ss}}(0)} \quad (5.3.13)$$

This represents the rate of change in the target steady state lean angle $\theta_{p,ss}$ due to time variation of v_x, v_y, and $\dot{\phi}$.

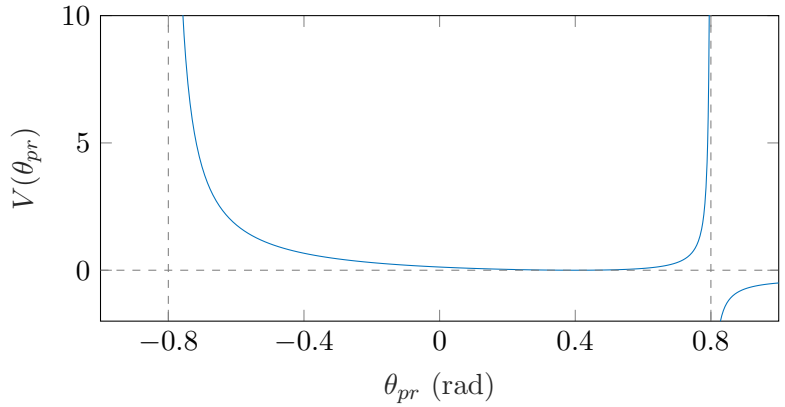


Figure 5.4: Barrier function $V(\theta_{pr}) = \frac{\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(0)\right)^2}{2\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)}$, with $\overline{\theta}_p = 0.8\,\text{rad}$ (dashed, vertical) and $f^{-1}_{\dot{v}_{y,ss}}(0) = 0.4\,\text{rad}$. Note the single unique minimum at $V(f^{-1}_{\dot{v}_{y,ss}}(0)) = 0$, and how $V(\theta_{pr}) \to \infty$ as $|\theta_{pr}| \to \overline{\theta}_{pr}$.

Consider the control laws

$$\dot{\theta}_{pr} = \frac{-\dot{f}^{-1}_{\dot{v}_{y,ss}}(0)\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)}{\left(f^{-1}_{\dot{v}_{y,ss}}(0)\theta_{pr} - \overline{\theta}_p^2\right)} - K_r\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(0)\right)\left(f^{-1}_{\dot{v}_{y,ss}}(0)\theta_{pr} - \overline{\theta}_p^2\right) + \frac{\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)^2 K_v f_{\dot{v}_{y,ss}}(\theta_{pr})(v_{yr} - v_y)}{\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(0)\right)\left(f^{-1}_{\dot{v}_{y,ss}}(0)\theta_{pr} - \overline{\theta}_p^2\right)} \tag{5.3.14}$$

$$\dot{w}_1 = -K_{w_1} w_1 + K_v(v_{xr} - v_x)\left(\overline{w}_1^2 - w_1^2\right)^2 \tag{5.3.15}$$

$$\dot{w}_2 = -K_{w_2} w_2 + K_{\dot{\phi}}(\dot{\phi}_r - \dot{\phi})\left(\overline{w}_2^2 - w_2^2\right)^2 \tag{5.3.16}$$

Substituting (5.3.14)-(5.3.16) into (5.3.12) yields

$$\dot{V}_\Sigma = \frac{-K_r\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(0)\right)^2\left(f^{-1}_{\dot{v}_{y,ss}}(0)\theta_{pr} - \overline{\theta}_p^2\right)^2}{\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)^2} + \frac{-K_{w_1} w_1^2}{\left(\overline{w}_1^2 - w_1^2\right)^2} + \frac{-K_{w_2} w_2^2}{\left(\overline{w}_2^2 - w_2^2\right)^2} \tag{5.3.17}$$

from which it is clear that $\dot{V}_\Sigma \leq 0$ for $(K_r, K_{w_1}, K_{w_2}) > 0$, thus proving closed-loop stability. This stability proof does, however, require $\{v_{xr}, v_{yr}, \dot{\phi}_r\} \in \mathcal{A}_{ss}$, $x_0 \in \mathcal{A}_{ss}$, and $x \in \left\{\mathcal{A}_{ss} \ : |\theta_p| \leq \overline{\theta}_p\right\} \ \forall \ t$. While the first two conditions can be trivially ensured, the latter cannot be guaranteed, as any overshoot when approaching references that require $\theta_{p,ss}$ to lie close to $\overline{\theta}_p$ could violate this condition. This can be addressed by bounding the solution to $f^{-1}_{\dot{v}_{y,ss}}(0)$ to lie within constraint bounds. Using LaSalle's invariance principle it is apparent that

$$\lim_{t\to\infty} \begin{cases} \theta_{pr} = f^{-1}_{\dot{v}_{y,ss}}(0) \implies v_y = v_{yr} \\ w_1 = 0 \implies v_x = v_{xr} \\ w_2 = 0 \implies \dot{\phi} = \dot{\phi}_r \end{cases} \tag{5.3.18}$$

thus guaranteeing asymptotic convergence to the desired references.

The dynamics of the controller can be tuned by modification of the 'damping' terms K_r, K_{w_1}, and K_{w_2}, and 'proportional' terms K_v and $K_{\dot{\phi}}$. Convergence of the expression

$$\lim_{\theta_{pr}\to f^{-1}_{\dot{v}_{y,ss}}(0)} \frac{f_{\dot{v}_{y,ss}}(\theta_{pr})}{\left(f^{-1}_{\dot{v}_{y,ss}}(0) - \theta_{pr}\right)} \tag{5.3.19}$$

in the latter term of (5.3.14) cannot be directly determined, as performing the substitution $\theta_{pr} = f^{-1}_{\dot{v}_{y,ss}}(0)$ yields the indeterminate expression

$$\lim_{\theta_{pr}\to f^{-1}_{\dot{v}_{y,ss}}(0)} \frac{f_{\dot{v}_{y,ss}}(\theta_{pr})}{\left(f^{-1}_{\dot{v}_{y,ss}}(0) - \theta_{pr}\right)} = \frac{f_{\dot{v}_{y,ss}}(f^{-1}_{\dot{v}_{y,ss}}(0))}{\left(f^{-1}_{\dot{v}_{y,ss}}(0) - f^{-1}_{\dot{v}_{y,ss}}(0)\right)} = \frac{0}{0} \tag{5.3.20}$$

However, as these functions are known to be continuously differentiable within the region of interest, convergence can instead be determined by L'Hôpital's rule as

$$\lim_{\theta_{pr}\to f^{-1}_{\dot{v}_{y,ss}}(0)} \frac{f_{\dot{v}_{y,ss}}(\theta_{pr})}{\left(f^{-1}_{\dot{v}_{y,ss}}(0) - \theta_{pr}\right)} = \lim_{\theta_{pr}\to f^{-1}_{\dot{v}_{y,ss}}(0)} \frac{\left(\frac{df_{\dot{v}_{y,ss}}(\theta_{pr})}{d\theta_{pr}}\right)}{\left(\frac{d\left(f^{-1}_{\dot{v}_{y,ss}}(0) - \theta_{pr}\right)}{d\theta_{pr}}\right)} \tag{5.3.21}$$

The inverse function $f^{-1}_{\dot{v}_{y,ss}}(0)$ is independent of θ_{pr} as $f^{-1}_{\dot{v}_{y,ss}}(0) = \theta_{p,ss}$ and $\theta_{pr} \not\equiv$

$\theta_{p,ss}$, so $\frac{df^{-1}_{\dot{v}_{y,ss}}(0)}{d\theta_{pr}} = 0$, and clearly $-\frac{d\theta_{pr}}{d\theta_{pr}} = -1$, therefore

$$\begin{aligned}\lim_{\theta_{pr}\to f^{-1}_{\dot{v}_{y,ss}}(0)} \frac{\left(\frac{df_{\dot{v}_{y,ss}}(\theta_{pr})}{d\theta_{pr}}\right)}{\left(\frac{d\left(f^{-1}_{\dot{v}_{y,ss}}(0)-\theta_{pr}\right)}{d\theta_{pr}}\right)} &= \lim_{\theta_{pr}\to f^{-1}_{\dot{v}_{y,ss}}(0)} -\frac{df_{\dot{v}_{y,ss}}(\theta_{pr})}{d\theta_{pr}} \\ &= -\frac{df_{\dot{v}_{y,ss}}(\theta_{pr})}{d\theta_{pr}}\bigg|_{\theta_{pr}=f^{-1}_{\dot{v}_{y,ss}}(0)}\end{aligned} \tag{5.3.22}$$

taking the derivative of (5.3.9) w.r.t. θ_p yields

$$\begin{aligned}\frac{df_{\dot{v}_{y,ss}}(\theta_{pr})}{d\theta_{pr}} = -\frac{g-\dot{\phi}^2 e+\cos(\theta_{pr})\left(gi-\dot{\phi}^2 ei\right)+\sin(\theta_{pr})\left(dv_y-\dot{\phi}hv_x\right)}{(i+\cos(\theta_{pr}))^2} \\ -b\dot{\phi}^2\cos(\theta_{pr})\end{aligned} \tag{5.3.23}$$

which is clearly a smooth function over the usual interval $\theta_{pr} \in \big(-\cos^{-1}(-i), \cos^{-1}(-i)\big)$, therefore (5.3.19) is convergent as $\theta_{pr} \to f^{-1}_{\dot{v}_{y,ss}}(0)$, and thus the control law (5.3.14) remains defined.

Finally, all that remains to be proven is that (5.3.19) does not converge to zero within the operating region of interest, as if this were the case the third term of (5.3.14) would vanish at $\theta_{pr} = f^{-1}_{\dot{v}_{y,ss}}(0)$, even if $v_y \neq v_{yr}$. As the first term of this control law vanishes when $w_1 = w_2 = 0$, and the second term also vanishes when $\theta_{pr} = f^{-1}_{\dot{v}_{y,ss}}(0)$, this would force $\dot{\theta}_{pr} = 0 \;\forall\; t \to \infty$, and thus prevent any further control action even when $v_y \neq v_{yr}$.

Ideally this would be performed by calculating the Hessian matrix of (5.3.23) w.r.t. $\begin{bmatrix}\theta_{pr} & v_x & v_y & \dot{\phi}\end{bmatrix}^T$, then by examining the eigenvalues of this matrix proving that (5.3.23) is either positive or negative definite. This would ensure that (5.3.23) never evaluates to zero, avoiding the above problem.

Unfortunately, these eigenvalues when represented symbolically are too complex for this analysis to be performed, so a less rigorous approach must be used. For the $v_x = v_y = \dot{\phi} = 0$ case (5.3.23) simplifies to

$$\frac{df_{\dot{v}_{y,ss}}(\theta_{pr})}{d\theta_{pr}} = -\frac{g(1+i\cos(\theta_{pr})}{(i+\cos(\theta_{pr}))^2} \tag{5.3.24}$$

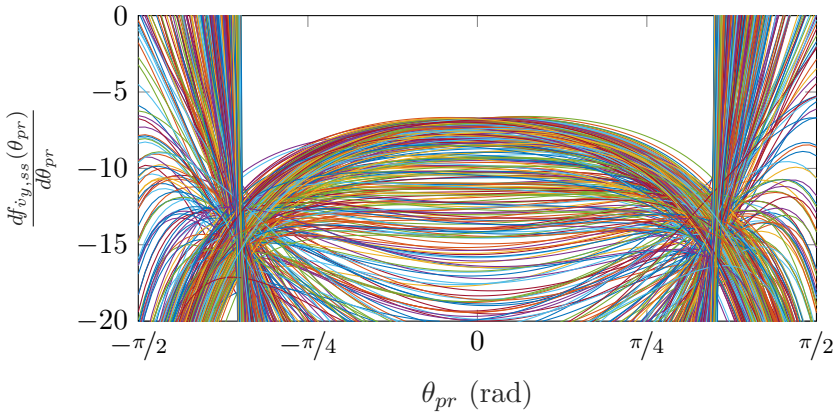


Figure 5.5: Plot of (5.3.23) over $\theta_{pr} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ for 1000 uniformly random samples $\{v_x, v_y, \dot{\phi}\} \in \mathcal{A}_{ss}$. This shows that in practice this function appears to be negative definite for $|\theta_p| \lessapprox 1.2\,\text{rad},\ \{v_x, v_y, \dot{\phi}\} \in \mathcal{A}_{ss}$

which is clearly negative definite for $\theta_{pr} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, indicating that the origin is contained within the unknown set of $\{v_x, v_y, \dot{\phi}\}$ for which this condition holds. The effect of nonzero v_x, v_y, and $\dot{\phi}$ is examined by evaluating (5.3.23) over $\theta_{pr} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ for 1000 uniformly random samples $\{v_x, v_y, \dot{\phi}\} \in \mathcal{A}_{ss}$, shown in Figure 5.5. This shows that in practice this function appears to be negative definite for $|\theta_p| \lessapprox 1.2\,\text{rad}$. This controller therefore remains well defined for all $\{v_x, v_y, \dot{\phi}\} \in \mathcal{A}_{ss}$, $|\theta_p| \lessapprox 1.2\,\text{rad}$. This is a tighter bound on θ_p than found previously, but still far larger than is expected to be attained in practice.

As this stability proof relies on the assumption of prior convergence of the inner $\theta_p \to \theta_{pr}$ control loop, update of the control law (5.3.14) should be avoided when $|\theta_p - \theta_{pr}| \gg 0$. This can be achieved by multiplication of (5.3.14) by the expression

$$e^{-K|\theta_p - \theta_{pr}|} \tag{5.3.25}$$

where $K \gg 1$, and where e is redefined as the natural logarithm. This prevents substantial change of θ_{pr} when the inner loop is still converging.

Figure 5.6 shows the phase portrait of (5.3.14), showing convergence from a number of initial states to the desired v_{yr} reference and non-zero solution to $f^{-1}_{\dot{v}_{y,ss}}(0)$, whilst satisfying the $|\theta_p| < \overline{\theta}_p$ constraint.

Remark 4. *Boundedness of internal dynamics*

From (5.1.11) z_6 *is a linear combination of states* x_5 *to* x_8*. As these states are*

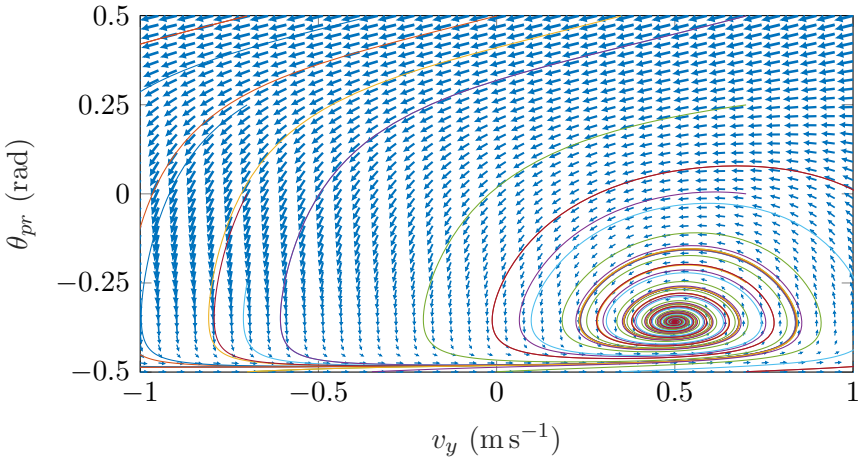


Figure 5.6: A phase portrait of the controller (5.3.14) for $v_x = v_{xr} = 1$, $\dot{\phi} = \dot{\phi}_r = 3$, $v_{yr} = 0.5$, $K_r = 3$, $K_v = 1$, $\overline{\theta}_p = 0.5$. This shows convergence of $v_y \to v_{yr}$ and $\theta_{pr} \to f^{-1}_{\dot{v}_{y,ss}}(0)$ from all states, whilst enforcing the constraint $|\theta_p| < \overline{\theta}_p$.

controlled they remain bounded, therefore z_6 remains bounded. z_1 and z_2 grow linearly and unboundedly, but as this is a velocity controller this is to be expected.

Figure 5.7 shows the simulated response of the prototype system with this controller to a reference $(v_{xr}, v_{yr}, \dot{\phi}_r) = (1, 1, 4)$, initialised at the origin. This shows asymptotic convergence to the reference whilst satisfying θ_p, w_1, and w_2 constraints, with θ_p correctly converging to the required steady state $\theta_p = \theta_{p,ss} = f^{-1}_{\dot{v}_{yss}}(0)$. The error $\theta_p - \theta_{pr}$ remains small, indicating that (5.3.25) functions as intended and thus the assumption of convergence of this inner loop holds, with full convergence achieved in steady state.

Figure 5.8 shows the experimental response of the prototype to a reference $(v_{xr}, v_{yr}, \dot{\phi}_r) = (0, 1, 2)$, again initialised at the origin. A more conservative reference is chosen than that used in simulation, as wheel slip is found to occur before the more aggressive reference can be reached. This results in the system following a circular trajectory whilst leaning in towards the center, demonstrated in Figure 5.9 using a strobe-effect image.

A small steady state tracking error, though hard to discern in this figure, is present in the linear $\theta_p \to \theta_{pr}$ controller. This is to be expected, as no model can perfectly describe the behaviour of a real-world system due to parameter uncertainty and unmodelled dynamics, meaning a model-derived feedback linearisation

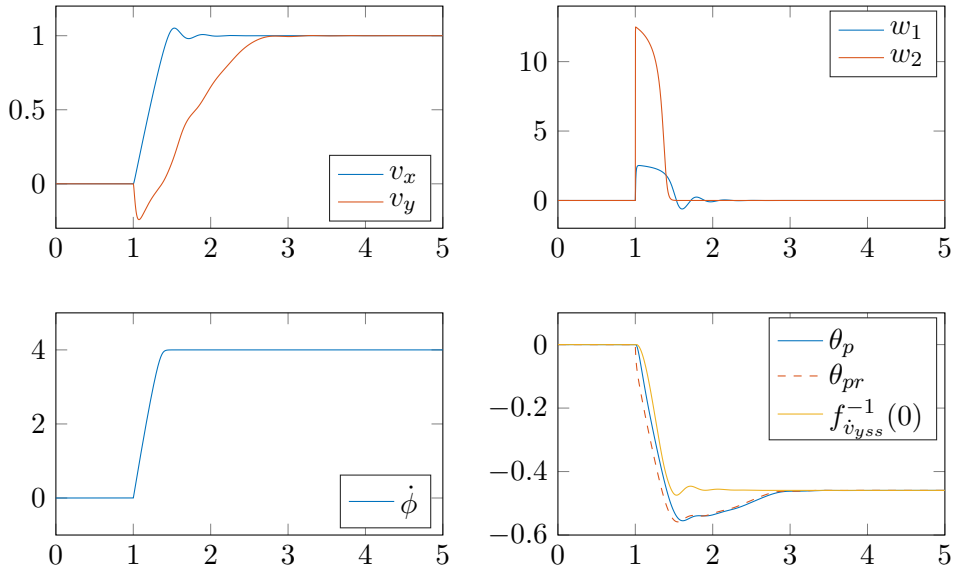


Figure 5.7: Simulated system state trajectories for a reference $(v_{xr}, v_{yr}, \dot{\phi}_r) = (1, 1, 4)$, initialised at the origin with $\overline{\theta}_p = 0.6$, $\overline{w}_1 = 2$, and $\overline{w}_2 = 4$. This shows asymptotic convergence to the reference whilst satisfying θ_p, w_1, and w_2 constraints, with θ_p correctly converging to the required steady state $\theta_p = \theta_{p,ss} = f^{-1}_{\dot{v}_{yss}}(0)$. The error $\theta_p - \theta_{pr}$ remains small, indicating that (5.3.25) functions as intended and thus the assumption of convergence of this inner loop holds, with full convergence achieved in steady state.

will always be imperfect and therefore not fully converge. This manifests as a steady state tracking error θ_e, i.e. $\theta_p \to \theta_{pr} + \theta_e$, which due to the proportional feedback term in this controller results in a non-zero steady state w_3, i.e. $w_3 \to K_{\theta_p}\theta_e \neq 0$. This invalidates the assumption in (5.3.9), yielding the steady state bias in the solution to $f^{-1}_{\dot{v}_{y,ss}}$, visible in this figure. A significant steady-state tracking error is visible in the $v_x \to v_{xr}$ controller, and $w_1 \not\to 0$. This again indicates an error in the feedback linearisation, as while $w_1 \neq 0$ the velocity v_x reaches a steady state. This could be addressed by improved friction modelling in the underlying model, as to predict this force resisting w_1 in steady state, or by some form of integral action.

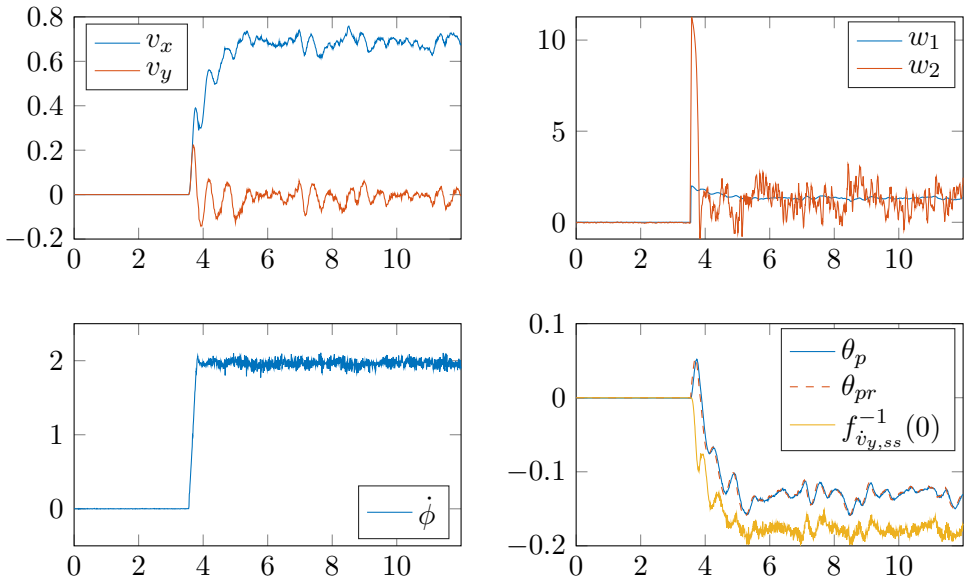


Figure 5.8: Experimental system state trajectories for a reference $(v_{xr}, v_{yr}, \dot{\phi}_r) = (1, 0, 2)$, initialised at the origin with $\overline{\theta}_p = 0.4$, $\overline{w}_1 = 3$, and $\overline{w}_2 = 15$. This shows good tracking of $\theta_p \to \theta_{pr}$, however, now $\theta_{pr} \not\to f^{-1}_{\dot{v}_{y,ss}}(0)$. This is found to be due to imperfect tracking within the inner loop yielding a steady state bias in w_3, which is in turn due to imperfect feedback linearisation. A combination of this and further model error yields a steady state tracking error of the v_{xr} and v_{yr} references, though in reality this is visually imperceptible.

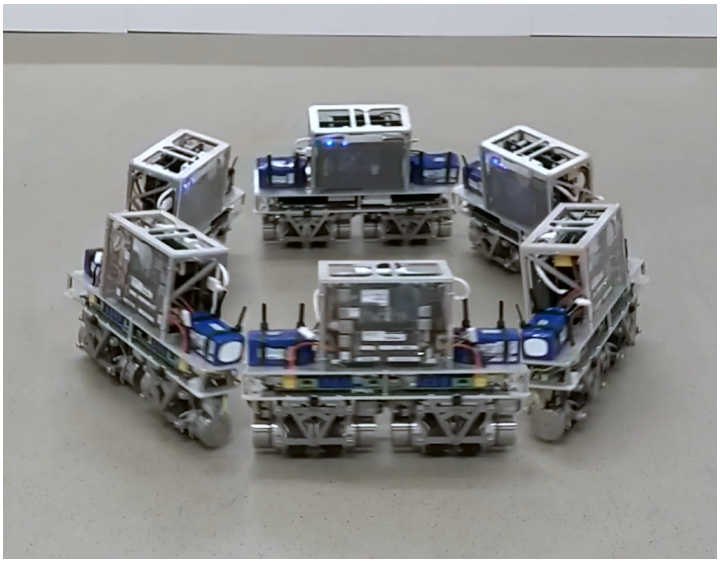


Figure 5.9: A strobe-effect image of the trajectory shown in Figure 5.8.

5.4 Backstepping Inertial Frame Velocity Control

Control of inertial frame velocities $\dot{x}$ and $\dot{y}$ is more useful in applications that involve the autonomous navigation of an environment. The desired steady state body accelerations are now defined as $\dot{v}_x = \dot{\phi} v_y$ and $\dot{v}_y = -\dot{\phi} v_x$, representing unforced body acceleration due to the mapping of inertial frame velocities into the rotating local frame.

Remark 5. *Inertial frame velocity equilibria*

For an inertial frame velocity controller it is desired that the inertial frame body velocities $\{\dot{x}, \dot{y}, \dot{\phi}\}$ asymptotically converge to the references $\{\dot{x}_r, \dot{y}_r, \dot{\phi}_r\}$ within finite time. In steady state the local frame body accelerations must therefore be purely that due to rotation of inertial frame velocities into the local body frame, i.e. $\dot{v}_x = \dot{\phi} v_y$, $\dot{v}_y = -\dot{\phi} v_x$, and local frame body velocities must be simply a rotation of the time invariant inertial frame velocity reference into the local frame, so

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = R_{EB}^T \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos(\phi)\dot{x} + \sin(\phi)\dot{y} \\ -\sin(\phi)\dot{x} + \cos(\phi)\dot{y} \end{bmatrix} \tag{5.4.1}$$

In steady state it is therefore required that $\dot{x} = \dot{x}_r$ and $\dot{y} = \dot{y}_r$, which for time invariant references implies $(\ddot{x}, \ddot{y}) \to 0$. It is also required that $\dot{\phi} = \dot{\phi}_r$, so $\phi = \dot{\phi}_r t + \phi_0$ in steady state.

Expressing inertial frame body accelerations in terms of inertial frame velocities and local body frame acceleration $\dot{v}_y$, and performing the above substitutions, yields

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\sin(\dot{\phi}_r t)(\dot{v}_y + \dot{\phi}_r \dot{x}_r \cos(\dot{\phi}_r t) + \dot{\phi}_r \dot{y}_r \sin(\dot{\phi}_r t)) \\ \cos(\dot{\phi}_r t)(\dot{v}_y + \dot{\phi}_r \dot{x}_r \cos(\dot{\phi}_r t) + \dot{\phi}_r \dot{y}_r \sin(\dot{\phi}_r t)) \end{bmatrix} \tag{5.4.2}$$

Substituting $\dot{v}_y$ in (5.4.2) *with* (5.3.3)*, solving again for $\begin{bmatrix} \ddot{x} & \ddot{y} \end{bmatrix}^T$, and equating to zero as required in steady state yields*

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{-\sin(\dot{\phi}_r t)\Gamma(t)}{i + \cos(\theta_p)} \\ \frac{\cos(\dot{\phi}_r t)\Gamma(t)}{i + \cos(\theta_p)} \end{bmatrix} \tag{5.4.3}$$

where

$$\Gamma(t) = w_3\left(c + ai + a\cos(\theta_p)\right) - \sin(\theta_p)\Big(g - f\dot{\theta}_p^2 + \dot{\phi}_r^2\left(e - bi - b\cos(\theta_p)\right)\Big) + \dot{\phi}_r\dot{y}_r h\sin(\dot{\phi}_r t) - \dot{y}_r d\cos(\dot{\phi}_r t) + \dot{\phi}_r\dot{x}_r h\cos(\dot{\phi}_r t) + \dot{x}_r d\sin(\dot{\phi}_r t) \quad (5.4.4)$$

For $\dot{\phi}_r \neq 0$ *these equalities clearly require* $\Gamma(t) = 0$*, which allows a unique solution for the required* w_3 *dynamics as*

$$w_3 = \frac{1}{c + a\cos(\theta_p) + ai}\Bigg(\sin(\theta_p)\Big(g + f\dot{\theta}_p^2 + \dot{\phi}_r^2\left(bi - e + b\cos(\theta_p)\right)\Big) + \cos(\dot{\phi}_r t)\Big(-\dot{\phi}_r\dot{x}_r h + \dot{y}_r d\Big) + \sin(\dot{\phi}_r t)\Big(-\dot{x}_r d - \dot{\phi}_r\dot{y}_r h\Big)\Bigg) \quad (5.4.5)$$

For the parameter choices in this thesis $(bi - e) \gg 0$*, so*

$$\frac{\sin(\theta_p)\Big(g + \dot{\phi}_r^2 b\cos(\theta_p) + f\dot{\theta}_p^2 + \dot{\phi}_r^2(bi - e)\Big)}{c + a\cos(\theta_p) + ai} \quad (5.4.6)$$

is an odd function within a neighbourhood of the origin for up to much larger values of $\dot{\phi}_r$ *than are expected to be encountered. Any deviation of* θ_p *from* 0 *will result in a similarly signed* w_3*, making* $\theta_p = 0$ *an unstable equilibrium for this part of* (5.4.5)*. The* $\cos(\dot{\phi}_r t)\Big(-\dot{\phi}_r\dot{x}_r h + \dot{y}_r d\Big) + \sin(\dot{\phi}_r t)\Big(-\dot{x}_r d - \dot{\phi}_r\dot{y}_r h\Big)$ *expression is time varying when* $\dot{\phi}_r \neq 0$*, which acts to perturb* (5.4.6)*.*

These dynamics can therefore only be stabilised if this is achieved by the

$$\cos(\dot{\phi}_r t)\Big(-\dot{\phi}_r\dot{x}_r h + \dot{y}_r d\Big) + \sin(\dot{\phi}_r t)\Big(-\dot{x}_r d - \dot{\phi}_r\dot{y}_r h\Big)$$

term, which given that this expression is invariant in θ_p *cannot be the case. An unstable equilibrium can be achieved for the* $\dot{\phi}_r = 0$ *case, as this makes the latter expression time invariant, allowing a time invariant solution for* w_3*. The overall dynamics are therefore unstable for* $\theta_p \in [-\frac{\pi}{2}, \frac{\pi}{2}]$*, meaning that for* $\ddot{x} = \ddot{y} = 0$ *to be maintained when* $\dot{\phi}_r \neq 0$ *the system is is forced to violate the constraint* $|\theta_p| < \frac{\pi}{2}$*, making the asymptotic tracking of constant* $\dot{x}_r$ *and* $\dot{y}_r$ *trajectories with a time varying heading impossible.*

As from Remark 5 no states satisfying $\|v\|\dot{\phi} \neq 0$ represent equilibria, asymp-

totic tracking of constant inertial frame velocity references is not possible, and thus a degree of tracking error is to be expected. As in the body frame velocity controller it is desired that $\theta_p \to f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})$, so a similar energy function to that in (5.3.11) can be used, though now this will never be perfectly tracked in steady state. Quadratic energy functions are defined with unique minimums at $\dot{x} = \dot{x}_r$ and $\dot{y} = \dot{y}_r$. However, as these minimums no longer represent equilibria of the system it is expected that the controlled system will follow a periodic trajectory about these references in steady state, with the characteristics of this limit cycle tunable by manipulation of control gains. Identical energy functions to that in (5.3.11) are used to describe a quadratic cost on $|\dot{\phi} - \dot{\phi}_r|$ and a barrier function on w_2. A similar barrier function is used to constrain w_1. As the purpose of this barrier is to constrain wheel torque demands, it makes more sense in this application to only apply the barrier to forced body acceleration, rather than also constraining acceleration due to rotation of inertial frame velocities into the local body frame. The barrier function is therefore chosen to instead enforce $\left|w_1 - v_y\dot{\phi}\right| < \overline{v}_{xf}$.

These new constraints and quadratic reference tracking costs can be captured by the Lyapunov function candidate

$$V_\Sigma = \frac{\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\right)^2}{2\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)} + \frac{K_v\left((\dot{x}_r - \dot{x})^2 + (\dot{y}_r - \dot{y})^2\right)}{2} + \frac{K_{\dot{\phi}}(\dot{\phi}_r - \dot{\phi})^2}{2} + \frac{(w_1 - \dot{\phi}v_y)^2}{2\left(\overline{\dot{v}}_{xf}^2 - (w_1 - \dot{\phi}v_y)^2\right)} + \frac{1}{2\left(\overline{w}_2^2 - w_2^2\right)} \quad (5.4.7)$$

with time derivative

$$\dot{V}_\Sigma = \frac{\left(\dot{f}^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})(\overline{\theta}_p^2 - \theta_{pr}^2) + \dot{\theta}_{pr}\left(f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\theta_{pr} - \overline{\theta}_p^2\right)\right)\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\right)}{\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)^2} - K_v\big(\ddot{x}(\dot{x}_r - \dot{x}) + \ddot{y}(\dot{y}_r - \dot{y})\big) - K_{\dot{\phi}}w_2(\dot{\phi}_r - \dot{\phi}) + \frac{\overline{\dot{v}}_{xf}^2(\dot{\phi}v_y - w_1)\left(\frac{d\dot{\phi}v_y}{dt} - \dot{w}_1\right)}{\left(\overline{\dot{v}}_{xf} + \dot{\phi}v_y - w_1\right)^2\left(\overline{\dot{v}}_{xf} - \dot{\phi}v_y + w_1\right)^2} + \frac{w_2\dot{w}_2}{\left(\overline{\dot{\phi}}^2 - w_2^2\right)^2} \quad (5.4.8)$$

Substituting inertial frame accelerations $(\ddot{x}, \ddot{y})$ and velocities $(\dot{x}, \dot{y})$ for their body frame equivalents using

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \end{bmatrix} = R_{EB} \begin{bmatrix} v_{xr} \\ v_{yr} \end{bmatrix} = \begin{bmatrix} \cos(\phi)v_{xr} - \sin(\phi)v_{yr} \\ \sin(\phi)v_{xr} + \cos(\phi)v_{yr} \end{bmatrix} \tag{5.4.9}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = R_{EB} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \cos(\phi)v_x - \sin(\phi)v_y \\ \sin(\phi)v_x + \cos(\phi)v_y \end{bmatrix} \tag{5.4.10}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \frac{d}{dt}\left(R_{EB} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \right) = \begin{bmatrix} (\dot{v}_x - v_y\dot{\phi})\cos(\phi) - (\dot{v}_y + v_x\dot{\phi})\sin(\phi) \\ (\dot{v}_x - v_y\dot{\phi})\sin(\phi) - (\dot{v}_y + v_x\dot{\phi})\cos(\phi) \end{bmatrix} \tag{5.4.11}$$

allows (5.4.8) to be rewritten as

$$\begin{aligned} \dot{V}_\Sigma = & \frac{\left(\dot{f}^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})(\overline{\theta}_p^2 - \theta_{pr}^2) + \dot{\theta}_{pr}\left(f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\theta_{pr} - \overline{\theta}_p^2\right)\right)\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\right)}{\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)^2} \\ & + \frac{w_2\dot{w}_2}{\left(\overline{w}_2^2 - w_2^2\right)^2} - K_v\big((w_1 - v_y\dot{\phi})(v_{xr} - v_x) + (f_{\dot{v}_{y,ss}}(\theta_{pr}) + v_x\dot{\phi})(v_{yr} - v_y)\big) \\ & - K_{\dot{\phi}}w_2(\dot{\phi}_r - \dot{\phi}) + \frac{\overline{\dot{v}}_{xf}^2(\dot{\phi}v_y - w_1)\left(\frac{d\dot{\phi}v_y}{dt} - \dot{w}_1\right)}{\left(\overline{\dot{v}}_{x,f} + \dot{\phi}v_y - w_1\right)^2\left(\overline{\dot{v}}_{x,f} - \dot{\phi}v_y + w_1\right)^2} \end{aligned} \tag{5.4.12}$$

Substituting the control laws

$$\begin{aligned} \dot{\theta}_{pr} = & \frac{-\dot{f}^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)}{\left(f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\theta_{pr} - \overline{\theta}_p^2\right)} \\ & - K_r\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\right)\left(f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\theta_{pr} - \overline{\theta}_p^2\right) \\ & + \frac{\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)^2 K_v(f_{\dot{v}_{yss}}(\theta_{pr}) + v_x\dot{\phi})(v_{yr} - v_y)}{\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\right)\left(f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\theta_{pr} - \overline{\theta}_p^2\right)} \end{aligned} \tag{5.4.13}$$

$$\dot{w}_1 = \dot{\phi} f_{\dot{v}_{yss}}(\theta_{pr}) + v_y w_2 - K_{w_1}(w_1 - \dot{\phi} v_y) + \frac{K_v(v_{xr} - v_x)\left(\overline{\dot{v}}_{xf} - \dot{\phi} v_y + w_1\right)^2 \left(\overline{\dot{v}}_{xf} + \dot{\phi} v_y - w_1\right)^2}{\overline{\dot{v}}_{xf}^2} \tag{5.4.14}$$

$$\dot{w}_2 = -K_{w_2} w_2 + K_{\dot{\phi}}(\dot{\phi}_r - \dot{\phi})\left(\overline{w}_2^2 - w_2^2\right)^2 \tag{5.4.15}$$

into (5.4.12) yields

$$\dot{V}_\Sigma = \frac{-K_r\left(\theta_{pr} - f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\right)^2 \left(f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})\theta_{pr} - \overline{\theta}_p^2\right)^2}{\left(\overline{\theta}_p^2 - \theta_{pr}^2\right)^2} + \frac{-K_{w_2} w_2^2}{\left(\overline{w}_2^2 - w_2^2\right)^2} + \frac{-K_{w_1}\overline{\dot{v}}_{xf}^2 (w_1 - \dot{\phi} v_y)^2}{\left(\overline{\dot{v}}_{xf} + \dot{\phi} v_y - w_1\right)^2 \left(\overline{\dot{v}}_{xf} - \dot{\phi} v_y + w_1\right)^2} \tag{5.4.16}$$

from which it is clear that $\dot{V}_\Sigma \leq 0 \ \forall \ \{K_r, K_{w_1}, K_{w_2}\} > 0$, thus proving stability under the assumption that non-zero inertial frame velocities are attainable in steady state while $\dot{\phi} \neq 0$. As from Remark 5 this is not possible, this stability proof is invalidated, though as the necessary resulting limit cycle is expected to be small it is assumed that this stability proof is still relevant to some degree. Control gains are tuned as to achieve a desirable trade-off between control performance and minimisation of this periodic error trajectory.

As in the body velocity controller this controller also relies on the assumption $\theta_p = \theta_{pr}$, so update of the control is slowed by multiplying the second and third terms of (5.4.13) by

$$e^{-K|\theta_{pr} - \theta_p|} \qquad K \gg 1 \tag{5.4.17}$$

such that the control law is slowed when the inner loop has not converged, but without affecting the first term of (5.4.13) that is required to feedforward a necessary variation in θ_p due to rotation of inertial frame velocities into the local body frame.

Remark 6. *Boundedness of internal dynamics*

As in Remark 4, by controlling states x_5 to x_8, z_6, which is a linear combination of these, remains bounded. Again, z_1 and z_2 are not controlled and are expected to grow unboundedly.

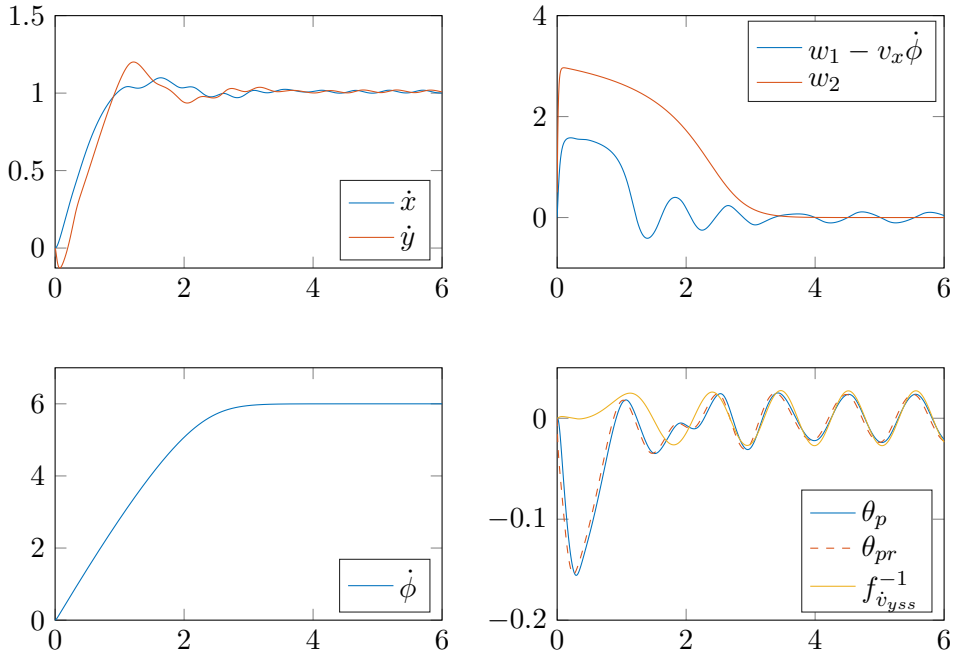


Figure 5.10: Simulated system state trajectories for a reference $(\dot{x}_r, \dot{y}_r, \dot{\phi}_r) = (1, 1, 6)$, with the system initialised at the origin and with $\overline{\theta}_p = 0.4$, $\overline{\dot{v}}_{xf} = 2$, and $\overline{w}_2 = 4$. $w_1 - v_x\dot{\phi}$ is shown rather than w_1, as this represents acceleration in the v_x subsystem not due to rotation, and is the value that is constrained. θ_p is seen to converge towards $f^{-1}_{\dot{v}_{yss}}(0)$, though a small tracking error is now observed. This is due to the infeasibility of asymptotically tracking inertial velocity references, as well as the now invalid assumption of θ_p converging to a constant value, i.e. now $\dot{\theta}_p \neq 0$, $w_3 \neq 0$ in steady state. $\dot{x}$ and $\dot{y}$ are seen to converge to a small limit cycle about the target reference.

Figure 5.10 shows the simulated response of the controlled system to a reference $(\dot{x}_r, \dot{y}_r, \dot{\phi}_r) = (1, 1, 6)$ with constraints $\overline{\theta}_p = 0.4$, $\overline{\dot{v}}_{xf} = 2$, and $\overline{w}_2 = 4$, with the system initialised at the origin. This demonstrates convergence to an acceptable velocity trajectory limit cycle with an RMS error of 4.1%, and satisfaction of the constraints $|\theta_{pr}| < \overline{\theta}_{pr}$, $|w_1 - v_x\dot{\phi}| < \overline{\dot{v}}_{xf}$, and $|w_2| < \overline{w}_2$. In steady state θ_p is seen to closely track $f^{-1}_{\dot{v}_{y,ss}}(-v_x\dot{\phi})$. Controller parameters are selected to best demonstrate the controller; more aggressive gains can obtain faster tracking without significantly altering the limit cycle.

Figure 5.11 shows the experimental response of the prototype to a reference $(\dot{x}_r, \dot{y}_r, \dot{\phi}_r) = (0, 1, 3)$. Wheel traction limitations necessitate a less aggressive

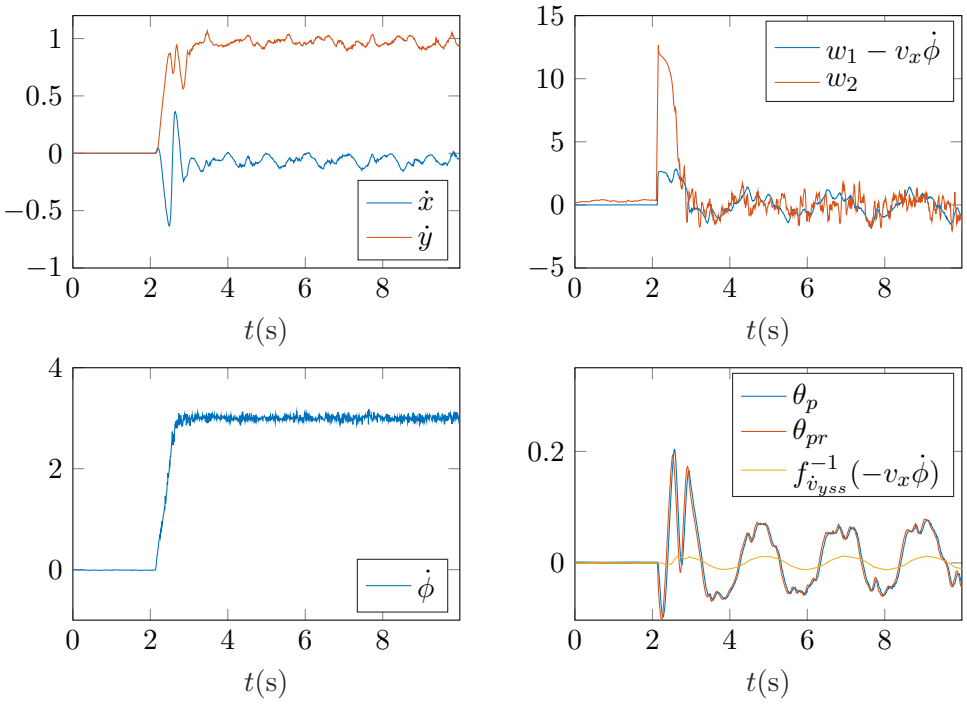


Figure 5.11: Experimental system state trajectories for a reference $(\dot{x}_r, \dot{y}_r, \dot{\phi}_r) = (0, 1, 3)$, with the system initialised at the origin and with $\overline{\theta}_p = 0.4$, $\overline{\dot{v}}_{xf} = 2$, and $\overline{w}_2 = 15$. $w_1 - v_x\dot{\phi}$ is shown rather than w_1, as this represents acceleration in the v_x subsystem not due to rotation, and is the value that is constrained. RMS $\dot{x}$ and $\dot{y}$ tracking errors of 4.2% and 7.1% are visible, due to a combination of the infeasibility of perfect tracking, imperfect feedback linearisation, and modelling error in the outer control laws.

reference than that in Figure 5.10. This experiment highlights a weakness in this controller; just as selection of controller gains affects the system's resulting limit cycle, this is also influenced by imperfect feedback linearisation and modelling error in the control laws, yielding larger periodic velocity tracking errors, with RMS errors of 10.8% and 7.8% respectively. From observation it is believed that the main influencing unmodelled dynamic is related to friction in the Mecanum wheel rollers, which in practise will not be perfectly modelled by the linear friction models used in this thesis. Figure 5.12 uses a long exposure image to demonstrate this experiment.

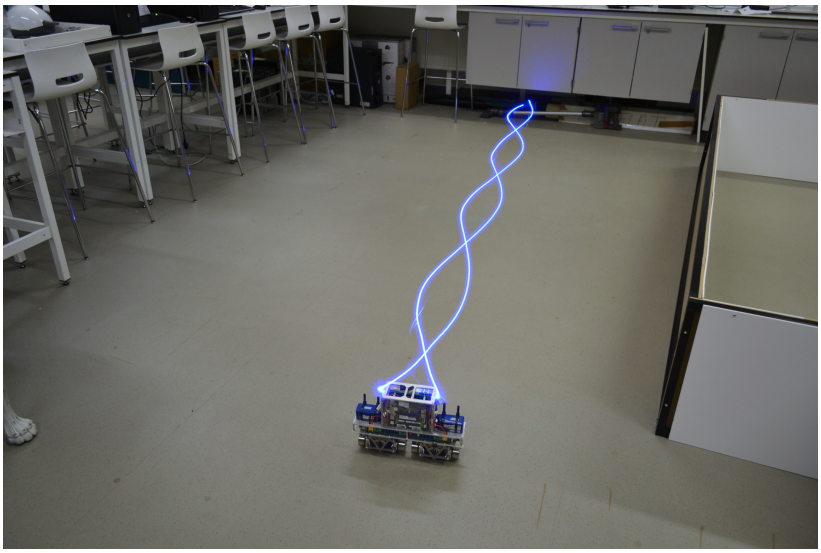


Figure 5.12: A long exposure image of the trajectory in Figure 5.11, in which two blue LEDs are used to capture the experimentally tracked path.

5.5 Backstepping Global Position Control

With system inertial frame velocities successfully controlled it is relatively straightforward to design a controller capable of generating $(\dot{x}_r, \dot{y}_r, \dot{\phi}_r)$ reference trajectories that drive the system to some arbitrary position in the inertial frame $(p_{x_r}, p_{y_r}, p_{\phi_r})$. This must be performed whilst enforcing a velocity constraint in order to bound the system's kinetic energy as to generate safe velocity trajectories. Such a controller is significant, as this allows the system to perform point-to-point translations in its environment, and is therefore a prerequisite for autonomous navigation between waypoints.

Consider the candidate Lyapunov function

$$V_\Sigma = \frac{K_p\left((p_{x_r} - x)^2 + (p_{y_r} - y)^2\right)}{2} + \frac{K_\phi\left(p_{\phi_r} - \phi\right)^2}{2} + \frac{1}{2(\bar{v}^2 - \dot{x}_r^2 - \dot{y}_r^2)} + \frac{1}{2(\bar{\dot{\phi}}^2 - \dot{\phi}_r^2)} \quad (5.5.1)$$

with time derivative

$$\dot{V}_\Sigma = -K_p\left(\dot{x}_r(p_{x_r} - x) + \dot{y}_r(p_{y_r} - y)\right) + \frac{\dot{x}_r\ddot{x}_r + \dot{y}\ddot{y}_r}{\left(\overline{v}^2 - \dot{x}_r^2 - \dot{y}_r^2\right)^2} \\ - K_\phi\dot{\phi}_r(p_{\phi_r} - \phi) + \frac{\dot{\phi}_r\ddot{\phi}_r}{(\dot{\phi}_r^2 - \dot{\overline{\phi}}_r^2)^2} \quad (5.5.2)$$

in which convergence of the lower velocity controller is assumed such that $\dot{x} = \dot{x}_r$, $\dot{y} = \dot{y}_r$, $\dot{\phi} = \dot{\phi}_r$. The first two terms of (5.5.1) define a cost quadratic in position error, the third term forms a barrier function enforcing the constraint $\dot{x}_r^2 + \dot{y}_r^2 < \overline{v}^2$, with a unique minimum at $\dot{x}_r = \dot{y}_r = 0$, and the last term enforces $|\dot{\phi}| < \dot{\overline{\phi}}_r$ with a unique minimum at $\dot{\phi}_r = 0$.

Substituting the control laws

$$\ddot{x}_r = -K_{v_r}\dot{x}_r + \left(\overline{v}^2 - \dot{x}_r^2 - \dot{y}_r^2\right)^2 K_p(p_{x_r} - x) \quad (5.5.3)$$

$$\ddot{y}_r = -K_{v_r}\dot{y}_r + \left(\overline{v}^2 - \dot{x}_r^2 - \dot{y}_r^2\right)^2 K_p(p_{y_r} - y) \quad (5.5.4)$$

$$\ddot{\phi}_r = -K_{\dot{\phi}}\dot{\phi}_r + (\dot{\phi}_r^2 - \dot{\overline{\phi}}_r^2)^2 K_\phi(p_{\phi_r} - \phi) \quad (5.5.5)$$

into (5.5.2) yields

$$\dot{V}_\Sigma = \frac{-K_{v_r}\left(\dot{x}_r^2 + \dot{y}_r^2\right)}{\left(\overline{v}^2 - \dot{x}_r^2 - \dot{y}_r^2\right)^2} + \frac{-K_{\dot{\phi}_r}\dot{\phi}_r^2}{\left(\dot{\phi}_r^2 - \dot{\overline{\phi}}_r^2\right)^2} \quad (5.5.6)$$

from which is it clear that $\dot{V}_\Sigma \leq 0 \;\forall\; \{K_{v_r}, K_{\dot{\phi}_r}\} > 0$, and that $\dot{V}_\Sigma = 0$ has a unique solution at the desired steady state, proving stability. Similar to the velocity controller, update of each control law is slowed by multiplication with terms of the form

$$e^{-K|\dot{x}_r - \dot{x}|} \quad K \gg 1 \quad (5.5.7)$$

so that the assumption of lower loop convergence holds.

Figure 5.13 shows the simulated response of the above controller to the reference $(p_{x_r}, p_{y_r}, p_{\phi_r}) = (2, 2, 2\pi)$, with the system initialised at the origin and with $\overline{\theta}_p = 0.6$, $\overline{v} = 1$, $\overline{v}_{xf} = 2$, and $\overline{w}_2 = 4$. $w_1 - v_x\dot{\phi}$ is shown rather than w_1, as this represents acceleration in the v_x subsystem exclusive of that due to rotation; it is this value that is constrained by the lower velocity controller. This demonstrates asymptotic position reference tracking with minimal overshoot, and

sensible smooth velocity trajectories that satisfy constraints.

Figure 5.14 shows the experimental response of the prototype to exactly the same reference trajectories with identical control gains. This results in the system tracking a nearly identical position trajectory, though now there is more disturbance in the $\dot{x}$ and $\dot{y}$ states due to $\|v\|\dot{\phi} \neq 0$. Similarly, a more aggressive θ_p trajectory is required than in simulation to counter this deviation from the velocity reference. Again, all constraints are satisfied, and all state and input trajectories evolve as expected. A long exposure image demonstrating this controller is shown in Figure 5.15, in which two LEDs are used to capture the resulting tracked path.

5.6 Conclusion

This chapter has demonstrated a novel partial feedback linearisation of the CMD, transforming its dynamics from a system of six nonlinear and two linear ODEs to a system of three nonlinear and five linear ODEs. Backstepping control design is then used to create novel constrained local frame body velocity, inertial frame body velocity, and inertial position controllers, all with Lyapunov derived stability guarantees. While these stability proofs are not robust to model uncertainty, in practise this controller is found to exhibit significantly robust behaviour, and so it is expected that with more work a robust controller and stability proof could also be derived. These controllers have been demonstrated both in simulation and experimentally on a CMD prototype, with their performance in each evaluated.

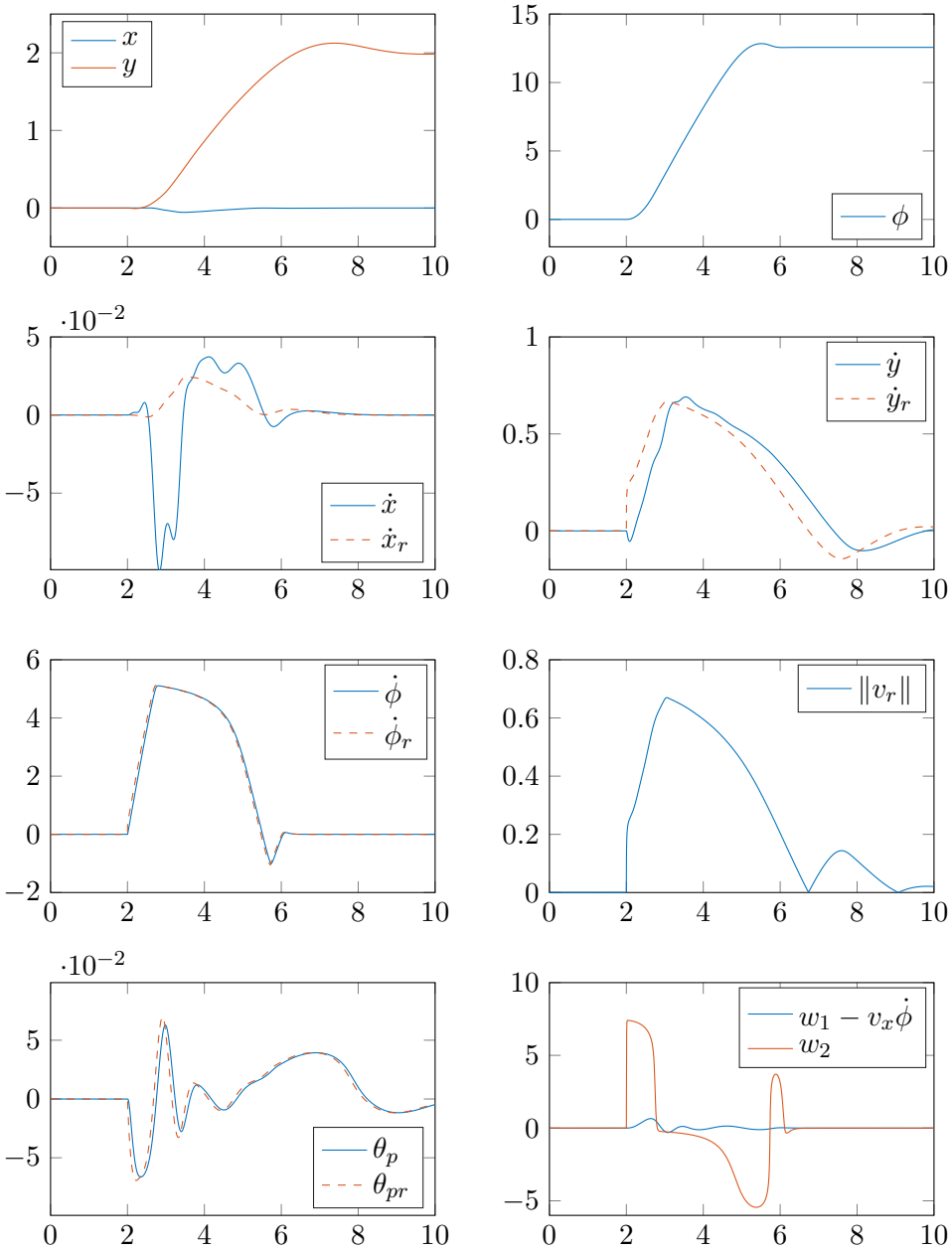


Figure 5.13: Simulated system state trajectories for a reference $(x_r, y_r, \phi_r) = (2, 2, 2\pi)$, initialised at the origin with $\overline{\theta}_p = 0.6$, $\overline{v} = 1$, $\overline{\dot{v}}_{xf} = 3$, and $\overline{w}_2 = 15$. $w_1 - v_x\dot{\phi}$ is shown rather than w_1, as this represents acceleration in the v_x subsystem exclusive of that due to rotation of the body frame, and is the value that is constrained. Convergence is slowed to better demonstrate asymptotic stability.

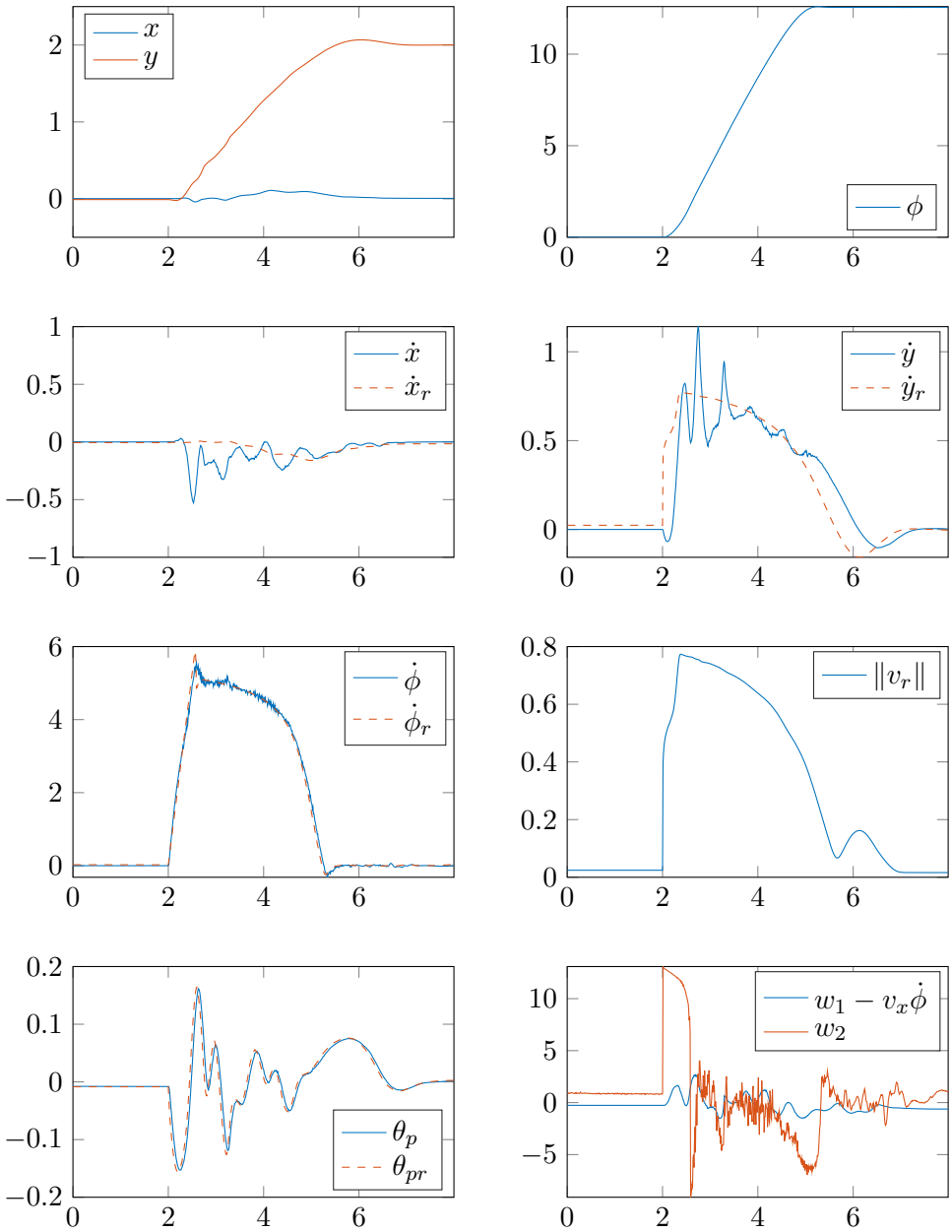


Figure 5.14: Experimental system state trajectories for reference $(x_r, y_r, \phi_r) = (0, 2, 4\pi)$, initialised at the origin with $\overline{\theta}_p = 0.4$, $\overline{v} = 1$, $\overline{\dot{v}}_{xf} = 3$, $\overline{w}_2 = 15$, $\overline{v} = 1$, and $\overline{\dot{\phi}} = 6$. $w_1 - v_x\dot{\phi}$ is shown rather than w_1, as this represents acceleration in the v_x subsystem not due to rotation, and is the value that is constrained. Small steady state errors in the tracking of $\dot{x}_r$ and $\dot{y}_r$ are due to the presence of static friction in the real-world system and a lack of integral action in the controller.

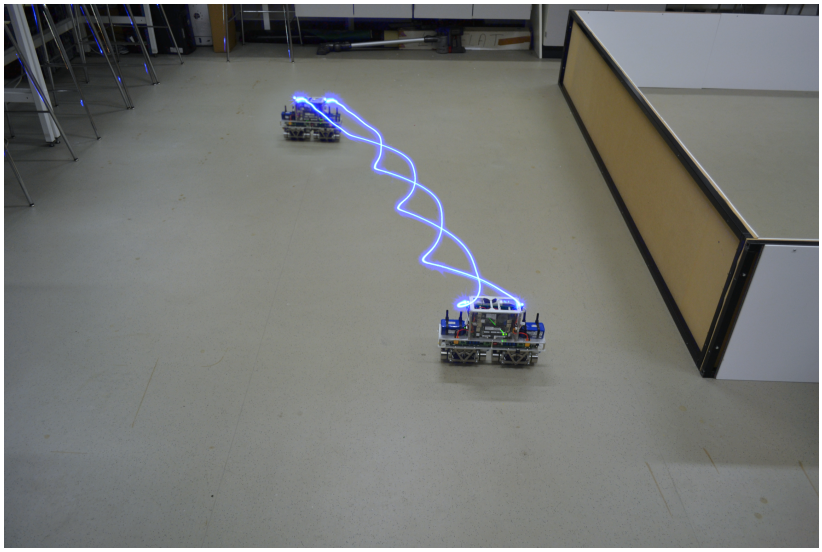


Figure 5.15: A long exposure image capturing a trajectory from the origin to $(x_r, y_r, \phi_r) = (1, 2, 4\pi)$, in which two blue LEDs are used to capture the experimentally tracked path, progressing from the bottom to top of the figure.

Chapter 6

Model Predictive Control

This chapter describes the development of the second CMD controller presented in this thesis, an input and output constrained dual-mode model predictive position controller. Like the backstepping controllers developed in the previous chapter, this controller is able to enforce lean angle and velocity constraints, whilst guaranteeing stability and convergence. However, this controller enforces these constraints in an optimal rather than asymptotic fashion, allowing for the generation of closer to time-optimal trajectories. This controller is also able to optimally enforce wheel torque constraints, providing a guarantee of wheel slip prevention for a given degree of traction whilst maintaining stability during actuator saturation. In a further advantage over the previous chapter, this controller is able to make optimal use of advance reference trajectory knowledge, allowing for improved tracking of complex Cartesian paths. The controller is demonstrated in both simulation and on a CMD experimental prototype, and its performance and shortcomings are evaluated. This chapter marks the first implementation of such a controller on a mobile balancing robot of any wheel configuration, and is therefore novel in a more general sense than just to the CMD on which it is applied.

As shown in Chapter 3, a Collinear Mecanum Drive requires larger wheel torques to generate motion than a similar TWIP, especially when accelerating along the $\hat{b}_x$ direction, due to both a required cancelling of body forces parallel to $\hat{b}_y$ and the addition of substantial friction in the wheel roller bearings. In practice, this means that typical desired accelerations along $\hat{b}_x$ often require wheel torques that are close to overcoming the tractive friction force between the roller

surface and the ground, i.e. torques that could cause wheel slip, especially when simultaneously accelerating in other dimensions. As slip is likely to result in instability or loss of control, a hard constraint must be observed on the magnitude of tractive force required at the roller-ground interface. Assuming a constant coefficient of static friction, this can be achieved by enforcing a hard constraint on the torque generated by each wheel motor. Here a 'hard' constraint refers to a constraint for which no violation is acceptable, as opposed to a 'soft' constraint, which may be violated during disturbance, and after which the controller is required to re-establish constraint satisfaction and resume tracking. While the backstepping controllers developed in Chapter 5 incorporated hard constraints on body accelerations $\dot{v}_x$, $\ddot{\phi}$, and $\ddot{\theta}_p$, these only act to approximately constrain wheel torques, and therefore in order to avoid wheel slip these constraints must be sized such that slip does not occur for the worst case combination of constraint satisfying accelerations. These constraints are therefore conservative for the vast majority of the constrained acceleration set. Furthermore, the enforcement of acceleration and velocity constraints in Chapter 5 is performed in an asymptotic rather than exact manner, meaning the resulting state trajectories are always suboptimal w.r.t. the constraints. In practice it would instead be desirable if the system's inputs could be saturated for long durations in order to sustain the greatest accelerations possible, and if the system could make maximum use of its constrained accessible velocity set in order to achieve the quickest possible translations. This must all be achieved without detrimentally affecting system stability, as would often occur in a rudimentary controller when its actuators saturate.

Rather than attempting to define a fixed control law such that the system's input and state trajectories iteratively evolve in a desirable manner, as demonstrated in Chapter 5, these trajectories can instead be calculated in an optimal fashion by minimisation of a suitable cost function subject to some constraints, an approach referred to as model predictive control (MPC).

Model predictive control typically uses a model of the plant to predict the system's response to a piecewise constant sequence of future control moves over a receding horizon, which can be optimally chosen to reduce a cost function that captures the controller's desired performance over a receding or infinite prediction horizon. This online optimisation allows for the systematic and exact handling

of constraints, making MPC well suited to this application. However, these approaches are computationally demanding due to their need to numerically solve a constrained minimisation problem for every control iteration, making their real-time implementation on systems with fast unstable dynamics such as the TWIP or CMD challenging. The complexity of this minimisation problem depends heavily on the prediction model and constraint structure; those with linear prediction models and constraints can usually be formulated as convex quadratic programs (QPs), which can be efficiently solved to global optimality. Those with nonlinear prediction models and constraints require the solution of a nonlinear program (NLP), typically a more computationally expensive undertaking. Nonlinear MPCs often possess nonconvex cost functions and constraints, meaning a guarantee of convergence to global optimality may not exist. Despite this, NMPC has been successfully applied to the cart-pole inverted pendulum [112]; however, this system possesses simpler and substantially slower dynamics than that in this thesis, and a fixed control delay is used to allow for a lengthy computation time, at a cost of a significant reduction in high frequency disturbance rejection. Complexity also varies greatly with choice of discretisation period, prediction and control horizons, and constraint quantity, meaning there is often value in tailoring each MPC implementation to its particular plant. Linear MPC formulations exist which allow for the straightforward derivation of proofs of convergence and stability, though similar techniques can sometimes also be applied to NMPC. This is typically achieved by minimising the cost function over an infinite rather than receding horizon, in which the control actions are used to drive the state into an invariant terminal constraint set, after which a fixed closed-loop control is implemented over the infinite horizon. These methods are often referred to dual-mode MPC, and are the de-facto standard for MPC implementations in which convergence and stability guarantees are required [113, 114].

Model predictive controllers with linear prediction models and constraints have been well studied in the context of cart-and-pole inverted pendulums [112, 115–117], but their application to TWIPs has been limited. Dini [60] implements a finite horizon MPC, but only for the control of the yaw and pitch states. Hirose [61] and Yue [59] also control a TWIP's forward velocity, incorporating both pitch and velocity constraints. However, both rely on an externally generated reference velocity trajectory in order to avoid infeasibility when performing

point-to-point manoeuvres, meaning the controller has no intrinsic guarantee of reference feasibility. Ohhira [118] comes close to implementing a full-state MPC position controller, in which a stabilising inner LQR is used to provide a closed-loop system that is then augmented by an outer optimal predictive controller, in a similar manner to dual-mode MPC. However, the inner loop is calculated with a 0.01 s sample time, whilst the outer optimisation is computed with a 0.08 s sample time, meaning constraint satisfaction can only be guaranteed when the inner and outer loop sampling instants coincide. This intermittent enforcement of the hard input constraints could allow constraint violation during the 70 ms gap between calculations of the augmenting input, inducing wheel slip and loss of control. Currently no successful implementations of a constrained MPC position controller for a TWIP exist in the literature.

As this controller is to be used to directly calculate desired wheel torque set-points, minimal computation time is critical. While sensitivity to control delay could be minimised by optimising a delayed input so as to allow a fixed time period for computation [112], as this controller is to be responsible for high frequency disturbance rejection this is not a suitable solution.

The goal of this chapter is therefore to develop a fast MPC capable of tracking both continuously varying and discontinuous position reference trajectories, whilst optimally enforcing hard wheel torque constraints $|\tau_i| \leq \overline{\tau},\ i \in [1\ ..\ 4]$, and softened lean angle and velocity constraints $|\theta_p| \leq \overline{\theta}_p$ and $v_x^2 + v_y^2 \leq \overline{v}^2$. For sake of safety, guarantees of stability and convergence should be provided.

6.1 Controller Derivation

The Jacobian linearisation of (3.2.19) about the stationary upright $\phi = 0$ position yields a linear state space prediction model suitable for the development of a linear model predictive controller in the form

$$x_{k+1} = Ax_k + Bu_k, \qquad y_k = Cx_k \tag{6.1.1}$$

where the output is chosen as $C = \begin{bmatrix} I_{3\times 3} & 0_{3\times 5} \end{bmatrix}$. This linearisation ignores the nonlinearity that is inherent in the integration of local body frame velocities to global inertial frame positions with a time varying ϕ; correction of this error is assumed to be provided in the external generation of reference trajectories. These

can be mapped into body frame position trajectories at the start of every control iteration, and replanned if excessive deviation occurs.

A dual-mode controller is chosen for its systematic design approach, its *a priori* stability guarantee, and its improved numerical conditioning in the prediction of open-loop unstable plants [114]. This controller uses the control law

$$\begin{aligned} u_{k+i} &= -Kx_{k+i} + c_{k+i} \qquad i \leq n_c \\ u_{k+i} &= -Kx_{k+i} \qquad\qquad\quad i > n_c \end{aligned} \tag{6.1.2}$$

for prediction timestep $i \geq 0$, i.e. $i = 0$ denotes the state at the current timestep k, K represents an unconstrained optimal feedback, and c_k represents the first element of an optimised sequence of n_c future perturbations from the unconstrained optimal, denoted $\underset{\rightarrow k}{c}$. The notation $\underset{\rightarrow k}{c}$ represents the column vector formed by stacking future values of c_j for $j = [k \;..\; k + n_c - 1]$, i.e. $\underset{\rightarrow k}{c} = \begin{bmatrix} c_k & c_{k+1} & \dots & c_{k+n_c-1} \end{bmatrix}^T$.

6.1.1 Reference Tracking

In order to track a varying reference r_k the cost function evaluated at an upright unstable equilibrium with $y_k = r_k$ must propose no change to the input. To achieve this the model must be redefined in terms of deviations $\hat{x}_k$ and $\hat{u}_k$ from the desired steady states $x_{ss|k}$ and $u_{ss|k}$ at step k as $x_k = \hat{x}_k + x_{ss|k}$ and $u_k = \hat{u}_k + u_{ss|k}$. The steady state values are calculated by the simultaneous equations $y_{ss|k} = Cx_{ss|k}$, $x_{ss|k} = Ax_{ss|k} + Bu_{ss|k}$, arranged to solve for $x_{ss|k}, u_{ss|k}$ in matrix form with $y_{ss|k} = r_k$ as

$$\begin{bmatrix} C & 0 \\ A - I & B \end{bmatrix}^{+} \begin{bmatrix} r \\ 0 \end{bmatrix} = \begin{bmatrix} x_{ss|k} \\ u_{ss|k} \end{bmatrix} = \begin{bmatrix} M_x \\ M_u \end{bmatrix} \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{6.1.3}$$

allowing $x_{ss|k}$ and $u_{ss|k}$ to be defined as

$$x_{ss|k} = M_x r_k \qquad u_{ss|k} = M_u r_k \tag{6.1.4}$$

where it is found that

$$M_x = C^+ = \begin{bmatrix} I_{3\times3} & 0_{3\times5} \end{bmatrix}^T \tag{6.1.5}$$

$$M_u = B^{-1}(A - I)C^+ = 0_{4\times3} \tag{6.1.6}$$

Substituting (6.1.2) into (6.1.1) and applying the above change of variables gives

$$\begin{aligned} x_{k+1} - x_{ss|k+1} &= \Phi(x_k - x_{ss|k}) + Bc_k \\ u_k - u_{ss|k} &= -K(x_k - x_{ss|k}) + c_k \end{aligned} \tag{6.1.7}$$

where $\Phi = A - BK$, which can then be substituted with (6.1.4) to give new closed-loop prediction model and control law

$$x_{k+1} = \Phi x_k + (I - \Phi)M_x r_{k+1} + Bc_k \tag{6.1.8}$$

$$u_k = -Kx_k + (KM_x + M_u)r_{k+1} + c_k \tag{6.1.9}$$

6.1.2 Autonomous Model Formulation and Reference Previewing

Instead of continuing with two separate control laws as in (6.1.2), analysis can be simplified by the formation of an all-encompassing autonomous prediction model of the form $Z_{k+1} = \Psi Z_k$. This allows constraints of the form $Gx \leq f$ to be applied to predicted step Z_{k+n} as $G\Psi^n Z_k \leq f$.

Additionally, advanced knowledge of n_r future reference inputs can be incorporated by the inclusion of $\underrightarrow{r}_{k+1} = \begin{bmatrix} r_{k+1} & r_{k+2} & \dots & r_{k+n_r+1} \end{bmatrix}^T$ into the autonomous model state. Typically $n_r \leq n_c$, as designing a controller to optimise its trajectory against a longer reference previewing horizon than its control horizon can introduce a transient tracking error, yield undesirable performance, and result in infeasibility of the QP [114]. In this thesis it is assumed that $n_r = n_c$, providing the controller with the maximum possible horizon of advanced reference knowledge. The change in advanced knowledge available to the controller over

the prediction horizon $k+i$ can be defined using a shift matrix D_r as

$$\underbrace{\begin{bmatrix} r_{k+2} \\ r_{k+3} \\ \vdots \\ r_{k+n_r+1} \\ r_{k+n_r+1} \end{bmatrix}}_{\underrightarrow{r}_{k+2}} = \underbrace{\begin{bmatrix} 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & I \end{bmatrix}}_{D_r} \underbrace{\begin{bmatrix} r_{k+1} \\ r_{k+2} \\ \vdots \\ r_{k+n_r+1} \end{bmatrix}}_{\underrightarrow{r}_{k+1}} \tag{6.1.10}$$

in which at $k > n_r$ the controller's final steady state reference is assumed to be equal to r_{k+n_r+1}.

Similarly, the matrix D_c captures the change in the future perturbation vector $\underrightarrow{c}_k$ over one prediction step

$$\underbrace{\begin{bmatrix} c_{k+1} \\ \vdots \\ c_{k+n_c} \\ 0 \end{bmatrix}}_{\underrightarrow{c}_{k+1}} = \underbrace{\begin{bmatrix} 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & I \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{D_c} \underbrace{\begin{bmatrix} c_k \\ c_{k+1} \\ \vdots \\ c_{k+n_c} \end{bmatrix}}_{\underrightarrow{c}_k} \tag{6.1.11}$$

in which no perturbation occurs for $k+i > n_c$.

This allows the definition of the autonomous prediction model $Z_{k+1} = \Psi Z_k$ as

$$\underbrace{\begin{bmatrix} x_{k+1} \\ \underrightarrow{c}_{k+1} \\ \underrightarrow{r}_{k+2} \end{bmatrix}}_{Z_{k+1}} = \underbrace{\begin{bmatrix} \Phi & BD_{c_k} & (\Phi - I)M_x D_{r_{k+1}} \\ 0 & D_c & 0 \\ 0 & 0 & D_r \end{bmatrix}}_{\Psi} \underbrace{\begin{bmatrix} x_k \\ \underrightarrow{c}_k \\ \underrightarrow{r}_{k+1} \end{bmatrix}}_{Z_k} \tag{6.1.12}$$

where D_{c_k} and $D_{r_{k+1}}$ select the first elements of $\underrightarrow{c}_k$ and $\underrightarrow{r}_{k+1}$ respectively.

6.1.3 Cost Function Derivation

A quadratic cost function that drives the state and input towards their desired steady state values can be defined as

$$J = \sum_{i=1}^{\infty} (x_{k+i} - x_{ss|k+i})^T Q (x_{k+i} - x_{ss|k+i}) + (u_{k+i-1} - u_{ss|k+i-1})^T R (u_{k+i-1} - u_{ss|k+i-1}) \quad (6.1.13)$$

where Q and R are diagonal matrices scaling the quadratic cost of each state and input error where $Q \succeq 0$, $R \succ 0$.

The terms $x_{k+i} - x_{ss|k+i}$ and $u_{k+i-1} - u_{ss|k+i-1}$ can be redefined in terms of the autonomous model state Z_k as

$$x_{k+i} - x_{ss|k+i} = \underbrace{\begin{bmatrix} I & 0 & -M_x D_{r_{k+1}} \end{bmatrix}}_{K_{xss}} Z_{k+i-1} \quad (6.1.14)$$

$$u_{k+i-1} - u_{ss|k+i-1} = \underbrace{\begin{bmatrix} -K \\ D_{c_k} \\ -M_u D_{r_{k+1}} \end{bmatrix}^T}_{K_{uss}} Z_{k+i-1} \quad (6.1.15)$$

allowing (6.1.13) to be rewritten in terms of K_{xss}, K_{uss}, and Z_k as

$$J = \sum_{i=1}^{\infty} (K_{xss} Z_{k+i})^T Q (K_{xss} Z_{k+i}) + (K_{uss} Z_{k+i-1})^T R (K_{uss} Z_{k+i-1}) \quad (6.1.16)$$

which can be rewritten as

$$J = \sum_{i=0}^{\infty} (K_{xss} Z_{k+i+1})^T Q (K_{xss} Z_{k+i+1}) + (K_{uss} Z_{k+i})^T R (K_{uss} Z_{k+i}) \quad (6.1.17)$$

It is then possible to substitute for Z_{k+i+1} using $Z_{k+i+1} = \Psi Z_{k+i}$

$$J = \sum_{i=0}^{\infty} Z_{k+i}^T \left[(K_{xss}\Psi)^T Q (K_{xss}\Psi) + (K_{uss})^T R (K_{uss}) \right] Z_{k+i} \quad (6.1.18)$$

From the autonomous model it is then seen that $Z_{k+i} = \Psi^i Z_k$, allowing substi-

tution of Z_{k+i} and factorisation of the now constant Z_k term

$$J = Z_k^T \left\{ \sum_{i=0}^{\infty} (\Psi^i)^T \Big(K_{xss}^T \Psi^T Q K_{xss} \Psi + K_{uss}^T R K_{uss} \Big) \Psi^i \right\} Z_k \tag{6.1.19}$$

The discrete Lyapunov equation can then be used to convert this convergent infinite series into the form

$$J = Z_k^T S Z_k = \begin{bmatrix} x_k \\ \underset{\rightarrow}{c}_k \\ \underset{\rightarrow}{r}_{k+1} \end{bmatrix}^T \begin{bmatrix} S_x & S_{xc} & S_{xr} \\ S_{xc}^T & S_c & S_{cr} \\ S_{xr}^T & S_{cr}^T & S_r \end{bmatrix} \begin{bmatrix} x_k \\ \underset{\rightarrow}{c}_k \\ \underset{\rightarrow}{r}_{k+1} \end{bmatrix} \tag{6.1.20}$$

This can be minimised by the solution of

$$\frac{\partial J}{\partial \underset{\rightarrow}{c}_k} = 0 \tag{6.1.21}$$

to define the unconstrained optimal perturbation required to incorporate advanced reference knowledge

$$\underset{\rightarrow}{c}_k = -S_c^{-1} \left(S_{xc} x_k + S_{cr} \underset{\rightarrow}{r}_{k+1} \right) \tag{6.1.22}$$

As expected, it is found that $\underset{\rightarrow}{c}_k = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}$ for $\underset{\rightarrow}{r}_{k+1} = \begin{bmatrix} r_{k+1} & r_{k+1} & \dots & r_{k+1} \end{bmatrix}^T$, and $S_{xc} \equiv 0$ and can therefore be omitted [114].

It is therefore clear that $\underset{\rightarrow}{c}_k \neq 0$ if any advanced reference knowledge is included, even for the unconstrained case. This goes against the dual-mode paradigm, in that for the unconstrained case the optimal trajectory should be fully captured by the underlying control law, meaning that in the absence of constraints it is desired that $\underset{\rightarrow}{c}_k = \begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}^T \; \forall \; \underset{\rightarrow}{r}_{k+1}$. To reinstate this intrinsic unconstrained optimality, the perturbation term can be redefined as $\underset{\rightarrow}{c}_k = \underset{\rightarrow}{\hat{c}}_k + -S_c^{-1} S_{cr} \underset{\rightarrow}{r}_{k+1}$, now instead optimising for $\underset{\rightarrow}{\hat{c}}_k$. The elements not dependant on the DoF $\underset{\rightarrow}{\hat{c}}_k$ can then be removed, allowing the redefinition of the QP and control law (6.1.8) as

$$\min_{\underset{\rightarrow k}{\hat{c}}} J = \left[\underset{\rightarrow k}{\hat{c}} - S_c^{-1} S_{cr} \underset{\rightarrow k+1}{r}\right]^T S_c \left[\underset{\rightarrow k}{\hat{c}} - S_c^{-1} S_{cr} \underset{\rightarrow k+1}{r}\right] + 2\left[\underset{\rightarrow k}{\hat{c}} - S_c^{-1} S_{cr} \underset{\rightarrow k+1}{r}\right]^T S_{cr} \underset{\rightarrow k+1}{r} \quad (6.1.23)$$

$$u_k = -Kx_k + \left[(KM_x + M_u) D_{r_{k+1}} - D_{c_k} S_c^{-1} S_{cr}\right] \underset{\rightarrow k+1}{r} + \hat{c}_k \quad (6.1.24)$$

Since the unconstrained optimal perturbation is known to be $\underset{\rightarrow k}{\hat{c}} = 0$, the performance index must be purely quadratic [119], allowing (6.1.23) to be simplified to

$$\min_{\underset{\rightarrow k}{\hat{c}}} J = \underset{\rightarrow k}{\hat{c}}^T S_c \underset{\rightarrow k}{\hat{c}} \quad (6.1.25)$$

The autonomous model (6.1.12) can be redefined to incorporate this reference previewing feedforward as

$$\underbrace{\begin{bmatrix} x_{k+1} \\ \underset{\rightarrow k+1}{c} \\ \underset{\rightarrow k+2}{r} \end{bmatrix}}_{Z_{k+1}} = \underbrace{\begin{bmatrix} \Phi & BD_{c_k} & \Gamma \\ 0 & D_c & 0 \\ 0 & 0 & D_r \end{bmatrix}}_{\Psi} \underbrace{\begin{bmatrix} x_k \\ \underset{\rightarrow k}{c} \\ \underset{\rightarrow k+1}{r} \end{bmatrix}}_{Z_k} \quad (6.1.26)$$

where $\Gamma = -BD_{c_k} S_c^{-1} S_{cr} + (I - \Phi) M_x D_{r_{k+1}}$.

6.1.4 Infeasible Reference Tracking

For this controller to be practically useful it must be able to drive the platform to any reference position and heading $\{x_r, y_r, \phi_r\} \in \mathbb{R}^3$ from any initial position. However, the nature of the dual mode controller means that there must exist a feasible perturbation sequence $\underset{\rightarrow k}{c}$ that takes the platform into the set of states from which the closed-loop system at $k > n_c$ will not violate any constraints over the infinite horizon, referred to as the maximal admissible set (MAS, $\mathcal{S}_{MAS}$) [114]. This is an invariant set, meaning once the underlying closed-loop system enters this set it will never leave, and will therefore never violate constraints. This lies within a set of states from which there exists a sequence of feasible control moves $\underset{\rightarrow k}{c}$ that drive the initial state into the MAS without constraint violation, referred to as the maximal controlled admissible set (MCAS, $\mathcal{S}_{MCAS}$), $\mathcal{S}_{MAS} \in \mathcal{S}_{MCAS}$. For

any initial state outside $\mathcal{S}_{MCAS}$ there does not exist a sequence of control moves that can guide the state into $\mathcal{S}_{MAS}$ within $k \leq n_c$ without violating constraints, rendering the QP infeasible. For this application this limits step translations to approximately 0.1m for $n_c = 10$, dependant on initial state and choice of quadratic cost function matrices Q and R, rendering the controller impractical for real-world implementation.

Multiple approaches exist to address this problem. Simon [120] replaces r with a pseudo-reference $\tilde{r}$ as an additional degree of freedom, penalising deviation of this from the true reference, making the QP feasible for all possible references. Dughman [121] introduces an extra perturbation term c_∞ to the end of the control sequence $\underset{\rightarrow k}{c}$, which acts as a constant perturbation to the input for $k > n_c$. This constant perturbation has the same effect as a pseudo-reference, with the equivalent pseudo-reference calculable as $\tilde{r} = C(I - \Phi)^{-1}Bc_\infty + r$. These methods ensure that $\mathcal{S}_{MCAS}$ spans all $\{x, y, \phi\} \in \mathbb{R}^3$.

Here the latter approach is taken, introducing a c_∞ term for $k > n_c$. The opportunity is also taken to introduce a vector of slack variables $\underset{\rightarrow k}{s}$ that will be later used to soften the controller's output constraints, giving an updated autonomous model Ψ and state Z

$$\underbrace{\begin{bmatrix} x_{k+1} \\ \underset{\rightarrow k+1}{c} \\ c_\infty \\ \underset{\rightarrow k+2}{r} \\ \underset{\rightarrow k+1}{s} \end{bmatrix}}_{Z_{k+1}} = \underbrace{\begin{bmatrix} \Phi & BD_{c_k} & 0 & \Gamma & 0 \\ 0 & D_c & E_c & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & D_r & 0 \\ 0 & 0 & 0 & 0 & D_s \end{bmatrix}}_{\Psi} \underbrace{\begin{bmatrix} x_k \\ \underset{\rightarrow k}{c} \\ c_\infty \\ \underset{\rightarrow k+1}{r} \\ \underset{\rightarrow k}{s} \end{bmatrix}}_{Z_k} \tag{6.1.27}$$

where E_c is used to replace the last value of $\underset{\rightarrow k+1}{c}$ with c_∞ as $E_c = \begin{bmatrix} 0 & 0 & \dots & 0 & I \end{bmatrix}^T$, and where D_s represents a shift matrix with a similar structure as D_c.

The QP (6.1.25) can now be redefined to additionally minimise c_∞^2 and $\underset{\rightarrow k}{s}^2$ as

$$\min_{\left\{\underset{\rightarrow k}{\hat{c}}, c_\infty, \underset{\rightarrow k}{s}\right\}} J = \begin{bmatrix} \underset{\rightarrow k}{\hat{c}} \\ c_\infty \\ \underset{\rightarrow k}{s} \end{bmatrix}^T \begin{bmatrix} S_c & 0 & 0 \\ 0 & WS_{c_\infty} & 0 \\ 0 & 0 & S_s \end{bmatrix} \begin{bmatrix} \underset{\rightarrow k}{\hat{c}} \\ c_\infty \\ \underset{\rightarrow k}{s} \end{bmatrix} \tag{6.1.28}$$

S_{c_∞} is scaled by a very small factor W so that choice of c_∞ has a minimal

effect on $\underrightarrow{c}_k$, whilst ensuring $c_\infty \to 0$ as $k \to \infty$ and feasibility $\forall \underrightarrow{r}_{k+1}$. S_s is a diagonal matrix of large slack weights used to heavily penalise deviation of $\underrightarrow{s}_k$ from 0, such that its elements are only optimised to be substantially larger than zero if feasibility of the QP would be otherwise lost. This means that the output constraints are only significantly violated if this is necessary to maintain feasibility of the QP.

6.1.5 Quadratic Cost Weights and Discretisation Period

The discretisation period T_s must be chosen as a trade-off between computational simplicity and prediction model accuracy, with the latter requirement related to the expected magnitude of control inputs. This is chosen through trial and error as $T_s = 35\,\text{ms}$. This does not dictate the control update rate - this can be performed over a much shorter period, only limited by the time required to solve each QP.

As for this application the controller is required to track reference body positions and heading, Q is initially set to $Q = \texttt{diag}\left(\begin{bmatrix}1 & 1 & 1 & 0_{1\times5}\end{bmatrix}\right)$. R must be chosen as a trade-off between control performance and avoiding the excitation of unmodelled dynamics, so a value of $R = 0.1I_{4\times4}$ is chosen.

6.1.6 Constraint Derivation

Hard input wheel torque constraints $|\tau| \leq \overline{\tau}$ are required in order to avoid wheel slip. Output constraints are required on the θ_p state in order to prevent the controller from attempting to translate using an unrealistic lean angle, as well as to keep the system near the model operating point. The v_x, v_y, and $\dot{\phi}$ states must be constrained in order to maintain a safe margin from the edge of $\mathcal{S}_{MCAS}$ for the controller to be able to handle disturbances, as well as to bound the system's kinetic energy as to ensure the controller generates safe and sensible velocity profiles.

Hard constraints on the input $|u_k| \leq \overline{u}$ and softened output constraints $|x_k| \leq$

$\overline{x} + s_k$ at timestep k can be represented in the form $GZ_k \leq f$ as

$$\underbrace{\begin{bmatrix} -K & D_{c_k} & 0 & P & 0 \\ K & -D_{c_k} & 0 & -P & 0 \\ C & 0 & 0 & 0 & -D_{sk} \\ -C & 0 & 0 & 0 & -D_{sk} \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} x_k \\ \underrightarrow{c}_k \\ c_\infty \\ \underrightarrow{r}_{k+1} \\ \underrightarrow{s}_k \end{bmatrix}}_{Z_k} \leq \underbrace{\begin{bmatrix} \overline{u} \\ \overline{u} \\ \overline{x} \\ \overline{x} \end{bmatrix}}_{f} \tag{6.1.29}$$

where $P = -D_{c_k} S_c^{-1} S_{cr} + (KM_x + M_u)\, D_{r_k}$.

These constraints can be projected over a $k + n_{con}$ constraint horizon by use of the autonomous model as[1]

$$\underbrace{\begin{bmatrix} G \\ G\Psi \\ G\Psi^2 \\ \vdots \\ G\Psi^{n_{con}} \end{bmatrix}}_{F} Z_k \leq \underbrace{\begin{bmatrix} f \\ f \\ f \\ \vdots \\ f \end{bmatrix}}_{t} \tag{6.1.30}$$

where $n_{con} > n_c$, and n_{con} is sufficiently large as to fully define $\mathcal{S}_{MAS}$, as from Section 6.1.4 membership of $\mathcal{S}_{MAS}$ at $k + n_c + 1$ is necessary to guarantee constraint satisfaction over the infinite horizon. However, this approach to defining $\mathcal{S}_{MAS}$ can result in redundant constraints, and provides no systematic method for selection of n_{con}.

Fortunately, algorithms for deriving the minimal set of constraints required to define $\mathcal{S}_{MAS}$ are well studied in the literature [122, 123]. Generally these function as follows. First, the initial set $S_0 = \{Z_k : GZ_k - t \leq 0\}$ is defined. Linear programming is then used to maximise each row of $GZ_k - t \leq 0$ subject to the other remaining constraints. If any row can be made to be larger than zero, then there exists a Z_k such that $Z_k \in S_0$ but $\Psi Z_{k+1} \notin S_0$, meaning S_0 is not invariant. The next set

$$S_1 = \{Z_k : \begin{bmatrix} GZ_k - t \\ G\Psi Z_k - t \end{bmatrix} \leq 0\} \tag{6.1.31}$$

[1] Note the rows of the G entry in F that constrain x_k serve no purpose and can be removed.

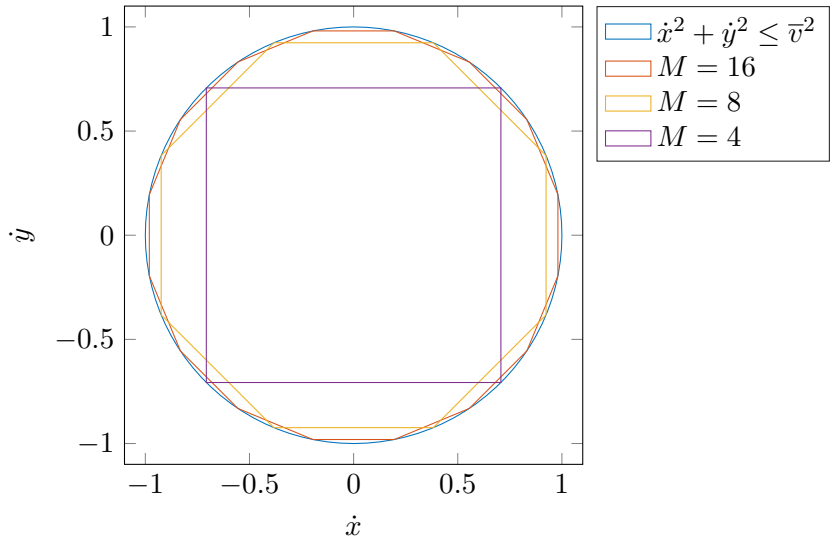


Figure 6.1: Polytope approximations of the quadratic velocity constraint $\dot{x}^2 + \dot{y}^2 \leq \overline{v}^2$ with $\overline{v} = \overline{v}_x = \overline{v}_y = 1$.

is then defined, and the above procedure is repeated until there exists no choice of Z_k that is able to force a constraint violation, so $Z_k \in S_n \implies Z_k \in S_{n+1}$ and thus S_n is invariant. This procedure always converges if the state asymptote lies within $\mathcal{S}_{MAS}$, i.e. $\lim_{i\to\infty} Z_{k+i} \in G\Psi^i Z_k \leq t$ [114]. Typically a number of redundant constraints will be introduced, which can be eliminated by various pruning methods.

6.1.7 Velocity Constraint Approximation

Whilst exact constraint satisfaction is achieved by enforcing $\dot{x}^2 + \dot{y}^2 \leq \overline{v}^2$, this forms a quadratic constraint, and therefore cannot be implemented using quadratic programming[2]. This can be avoided by approximating this constraint

[2]Quadratic constraints must be implemented using second-order cone programming if applicable, otherwise using semidefinite programming. These are both significantly more computationally expensive than quadratic programming.

using a convex polytope [124], defined by the linear inequalities

$$\left(\dot{y} - \overline{v}\sin\left(\frac{2\pi m}{M}\right)\right)\left(\cos\left(\frac{2\pi(m+1)}{M}\right) - \cos\left(\frac{2\pi m}{M}\right)\right) - \left(\dot{x} - \overline{v}\cos\left(\frac{2\pi m}{M}\right)\right)\left(\sin\left(\frac{2\pi(m+1)}{M}\right) - \sin\left(\frac{2\pi m}{M}\right)\right) \geq 0 \quad (6.1.32)$$

where $M \in [3 \,..\, \infty]$, $m \in [1 \,..\, M]$. Choices of $M = [4, 8, 16]$ are shown in Figure 6.1. Here $M = 8$ is chosen, yielding only a minor suboptimality w.r.t. the exact velocity constraint, whilst avoiding excessive growth of G.

6.1.8 Slack Variables and their Distribution

For the prototype used in this thesis with $n_c = 10$, $\mathcal{S}_{MAS}$ can be defined using 65 individual output constraints. Softening each of these constraints individually requires an equal number of slack variables, increasing the QP DoF from 44 to 109, a significant increase in complexity. To lessen this, the number of slack variables can be reduced by the sharing of slack variables across multiple timesteps through a redefinition of F_s and removal of diagonal entries from S_s, provided this does not reintroduce the possibility of output constraint infeasibility. Multiple slack variable sharing approaches were considered:

1. One slack variable per constraint per timestep.

2. One slack variable per constraint per timestep for $k \leq n_c$, with no constraint softening for $k > n_c$.

3. Sharing a single slack variable per constraint over the entire horizon.

4. Sharing a single slack variable per constraint for $k \leq n_c$, and sharing a single slack variable per constraint for $k > n_c$

5. Allocating one slack variable per constraint per timestep for $k \leq n_c$, with a single slack variable per constraint shared for $k > n_c$.

6. Allocating one slack per constraint per n_{block} timesteps, with a single slack per constraint shared for $k > n_c$.

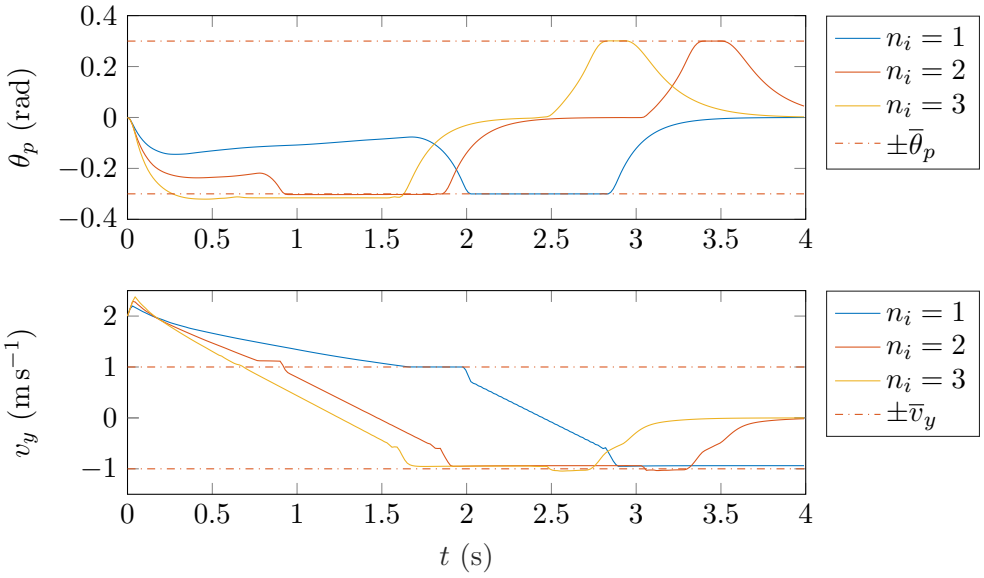


Figure 6.2: Comparison of controller response in the regulation case to an initial constraint violation of $v_y = 2\overline{v}_y$ for $n_i = [1, 2, 3]$

7. Allocating one slack variable per constraint per timestep for $k \leq n_i$, and a single shared slack variable per constraint for $k > n_i$, with n_i chosen by trial and error.

8. Allocating one slack per constraint per timestep for $k \leq n_i$, one slack per constraint per n_{block} timesteps for $n_i < k \leq n_c$, and a single slack variable per constraint shared for $k > n_c$.

Methods 2, 3, 4, and 6 showed eventual infeasibility during Monte Carlo analysis, whereas methods 1, 5, 7, and 8 all maintained feasibility. Of the feasible methods method 7 with $n_i = 1$ for $\dot{x}$ and $\dot{\phi}$ constraints and $n_i = 3$ for θ_p and $\dot{y}$ constraints allowed for the minimum number of slack variables, reducing n_s from the initial $n_s \approx 65$ down to $n_s = 12$. While this method with $n_i = 1$ for θ_p and $\dot{y}$ constraints was also able to maintain feasibility, the controller had insufficient degrees of freedom to quickly address constraint violations. A comparison of controller response for $n_i = [1, 2, 3]$ in the regulation case with an initial condition of $\dot{y} = 2\overline{v}_y$ is shown in Figure 6.2, in which a large variation in the time taken to re-establish output constraint satisfaction following a constraint violating disturbance is visible between different values of n_i.

Slack weights in (6.1.28) for non-shared slack variables are set to near zero to allow the controller to worsen constraint violation over a short horizon to improve the rate of convergence to S_O over the infinite horizon. For example, for an initially upright system with $|v_y| > 2\overline{v}_y$, the non-minimum phase nature of this system means that the desired control response is for the controller to first briefly increase v_y, driving $\theta_p < 0$ in order to start generating sustained deceleration $\dot{v}_y < 0$. A large slack weight for the slacks at $i \leq 3$ penalises this type of quick correction, preventing the system from correcting the violation as aggressively as is desired. The infinite horizon slack weights are made very large to ensure that all resulting trajectories bring the state towards S_O as quickly as possible. The cost associated with non-zero s_∞ must also be much greater than that practically encountered due to tracking error, otherwise a distant reference trajectory will cause the controller to purposefully generate an undesirable constraint violation.

6.1.9 The Effect of Input Constraints on Feasibility

While the output constraints enforced on v_x, v_y, $\dot{\phi}$, and θ_p are softened to maintain feasibility during disturbance, the hard input constraints applied to $\underset{\rightarrow k}{u}$ effectively apply a second set of output constraints, albeit with a larger constrained range in this particular application. This is due to the requirement for the controller to direct the state into $\mathcal{S}_{MAS}$ at $k+n_c+1$, in which the underlying control law must not violate the hard input constraints as $i \to \infty$. Given the range of output constraints acting on this system, this could take a reasonably long time from some choices of $x_k \in \mathcal{S}_O$; for example, a system initialised with $v_y = \overline{v}_y$ and $\theta_p = \overline{\theta}_p$ with $x_k \notin \mathcal{S}_{MAS}$ will require a large control horizon to give the controller enough time to manipulate the plant toward the origin and into $\mathcal{S}_{MAS}$ using its constrained input. This can lead to parts of the output constraint set $\mathcal{S}_O$ being infeasible, despite using output constraint softening. Care must therefore be taken to ensure n_c and $\overline{u}$ are sufficiently large when specifying $\mathcal{S}_O$ in order to ensure $\mathcal{S}_O \subseteq \mathcal{S}_{MCAS}$. Additionally, a large margin must be allowed due to the use of softened output constraints, otherwise an anticipated constraint violation could result in infeasibility. For this application the maximum possible v_y and θ_p values for which the controller must remain feasible are expected to be in the region of $\pm 3\,\mathrm{m\,s^{-1}}$ and $\pm 0.5\,\mathrm{rad}$ respectively, with minimal disturbance expected in the open-loop stable v_x and $\dot{\phi}$ states.

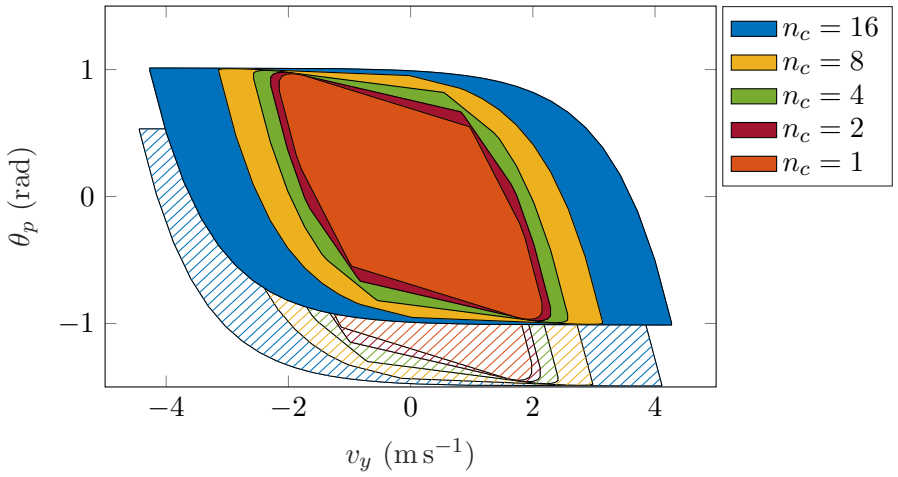


Figure 6.3: A cross section of $\mathcal{S}_{MCAS}$ through $\theta_{p|k}$ and $v_{y|k}$, taken through the origin (solid, front) and through $\dot{\theta}_{p|k} = 4\,\mathrm{rad\,s^{-1}}$ (hatched, back), with varying control horizon n_c and $Q = \mathtt{diag}\,(1_{1\times 3}, 0_{1\times 5})$.

The boundary of $\mathcal{S}_{MCAS}$ along each individual state can be examined by fixing all other elements and solving a linear program to find the maximum possible constrained value of this state. For example, the maximum initial stationary lean angle can be found by

$$\underbrace{\max}_{\{\theta_{p|k},\underrightarrow{c}_k,c_\infty\}} \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix} \begin{bmatrix}\theta_{p|k}\\ \underrightarrow{c}_k\\ c_\infty\end{bmatrix} \quad s.t. \quad F \begin{bmatrix}\theta_{p|k}\\ \underrightarrow{c}_k\\ c_\infty\end{bmatrix} \leq t \tag{6.1.33}$$

where F and t are redefined to include the now fixed elements of x_k.

Analysing a 2D cross-section of $\mathcal{S}_{MCAS}$ at the origin across $v_{y|k}$ and $\theta_{p|k}$ shows the relationship between feasibility and these two states, shown in Figure 6.3. Interestingly, for the QP to remain feasible for the desired output constraint set and anticipated magnitude of constraint violation a control horizon of $n_c \geq 8$ is required, placing a lower limit on n_c for this set of controller parameters. This analysis ignores the effect of the other remaining states on feasibility, with the hatched areas in Figure 6.3 showing a cross section of $\mathcal{S}_{MCAS}$ through $\dot{\theta}_{p|k} = 4\,\mathrm{rad\,s^{-1}}$, a more realistic representation of a large external impulse disturbance along the b_y axis. This emphasises the importance of a large control horizon and therefore large $\mathcal{S}_{MCAS}$ if a combination of equally signed disturbances in the

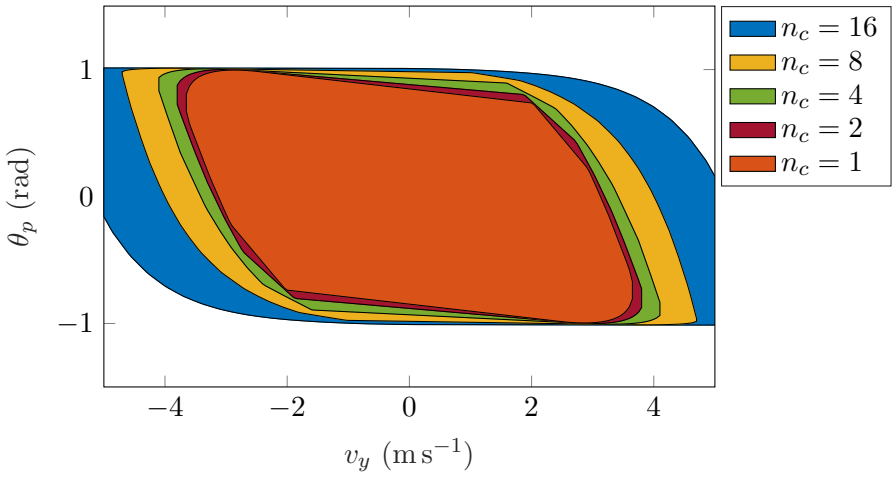


Figure 6.4: A cross section of $\mathcal{S}_{MCAS}$ through $\theta_{p|k}$ and $v_{y|k}$ for varying control horizon n_c and $Q = \texttt{diag}\left(\begin{bmatrix}1_{1\times 3} & 0_{1\times 4} & 0.01\end{bmatrix}\right)$.

v_y, θ_p, and $\dot{\theta}_p$ states are to not result in infeasibility. The relationship between choice of n_c and $\overline{u}$ also extends to the x and ϕ subsystems, but neither require a long control horizon in order for the MCAS to encompass the desired output constraints and their anticipated violations.

These figures demonstrates one of the main disadvantages of the dual-mode approach to predictive control, in that for a given cost function a long control horizon may be required in order to ensure a sufficiently large $\mathcal{S}_{MCAS}$ relative to the soft output constraint set. For the same cost function and constraints this can only be addressed by increasing n_c at a cost of execution time, or by increasing T_s at a cost of control performance and prediction model accuracy.

Alternatively, a modification to the cost function can be used to define an alternative unconstrained feedback K such that a larger MCAS can be obtained for the same choices of n_c and T_s. Q is therefore modified to $Q = \texttt{diag}\left(\begin{bmatrix}1_{1\times 3} & 0_{1\times 4} & 0.01\end{bmatrix}\right)$, with a cross-section of the new $\mathcal{S}_{MCAS}$ shown in Figure 6.4. This shows that a small control horizon is now able to fully access the anticipated constraint violation set, at a cost of less aggressive control of θ_p and v_y. A similar effect could be achieved through control move blocking [125], however, this method does not allow the embedding of the reference previewing feedforward term into the unconstrained control law as in (6.1.24).

6.2 Results

This section now aims to demonstrate the efficacy and efficiency of the proposed controller in simulation and on an experimental prototype. The quadratic program is solved using qpOASES [126], an online active set solver, implemented on an Intel i7-4720HQ processor for simulated results and an Intel i7-8650U for experimental results. In single threaded benchmarks the latter achieves 17% greater performance than the former. Simulation is performed by numerical integration of the continuous time nonlinear model using MATLAB's `ode45`. In simulation the controller is updated at a rate of $1/T_s$, whereas on the experimental prototype it is updated as continuously as execution time allows in order to improve disturbance rejection, with a typical control update rate of around 500 Hz.

6.2.1 Step Reference Tracking - Forward Translation

First, a step reference of $r_y = [0, 1]$ is used to demonstrate the step response of the (y, θ_p) subsystem, with the simulated response shown in Figure 6.5. This shows a response with asymptotic convergence, minimal overshoot, and sensible preemption of the reference change. The infinite horizon perturbation c_∞ correctly increases at the moment of the reference step to maintain feasibility, tending to zero once $x \in \mathcal{S}_{MCAS}$. The perturbation terms $c_{0,i}$ show a similar response, tending to zero once $x \in \mathcal{S}_{MAS}$. The controller shows satisfaction of all constraints, saturating the input for a number of samples and producing toward minimum-time v_y and θ_p trajectories. A small amount of $|v_y| \leq \overline{v}_y$ constraint violation is observed; this is to be expected, as this is related to the low cost associated with the softening of this constraint for $k < n_i$. Execution time peaks at the instant of the reference step to 3.5 ms, an acceptable control delay, quickly dropping below 2 ms for the remainder of the trajectory, and computing nearly instantly once $x \in \mathcal{S}_{MAS}$. The cost J is observed to be monotonic from the first knowledge of the impending reference change onwards, indicating good problem formulation.

Figure 6.6 shows the response of the experimental prototype to this step reference. This shows a strong similarity to that predicted in Figure 6.5, with only minor increases in constraint violation due to external disturbance. Execution time is found to be slightly faster, demonstrating the real-world feasibility of

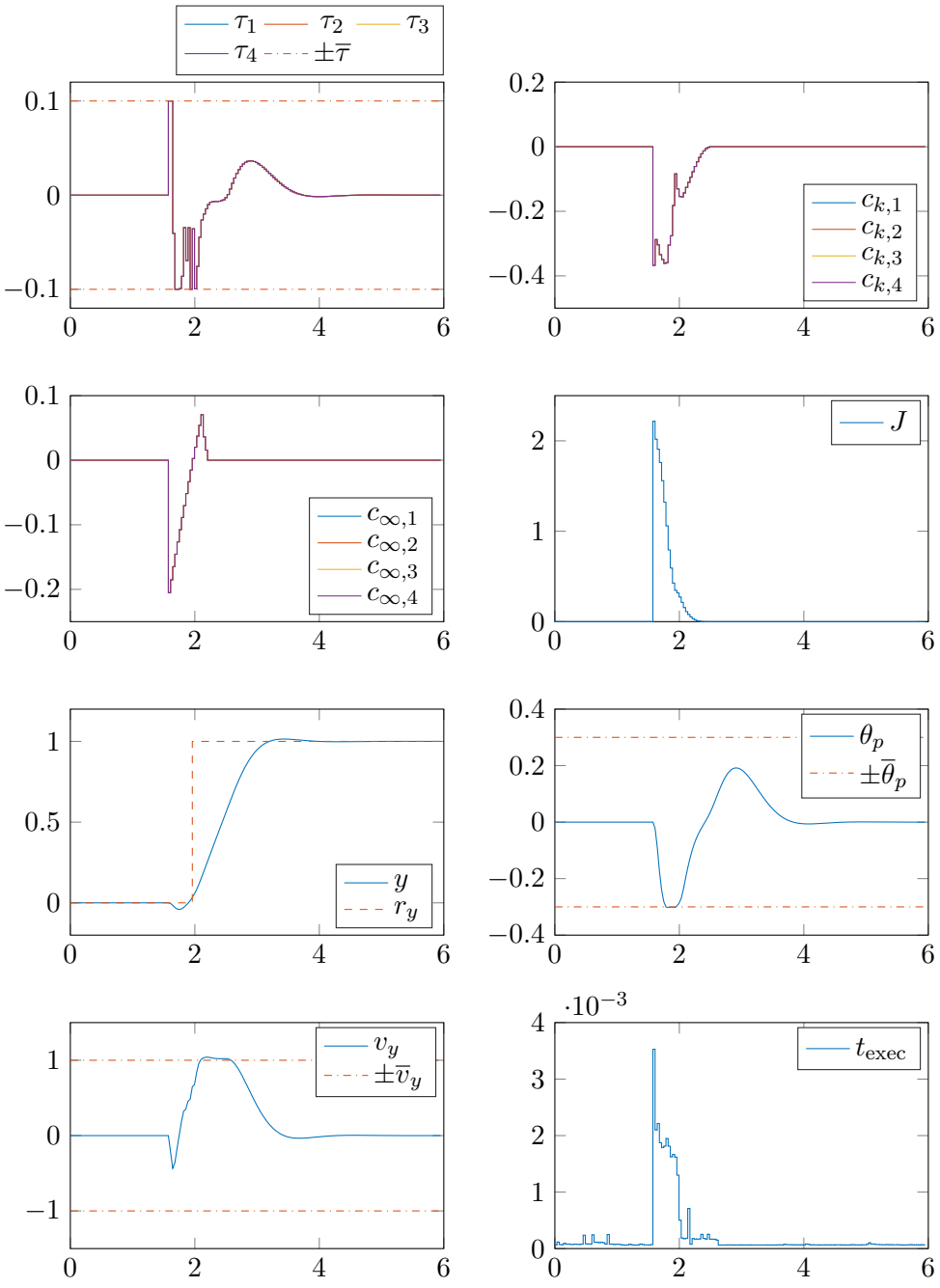


Figure 6.5: Simulated MPC response to a step reference of $y = [0, 1]m$, showing asymptotic convergence, minimal overshoot, and sensible preemption of the reference change. The $i = 0$ perturbation $c_{0,i}$ and infinite horizon perturbation c_∞ terms correctly increase at the moment of the reference step to maintain feasibility, and tend to zero as $x \in \mathcal{S}_{MAS}$ and $x \in \mathcal{S}_{MCAS}$ respectively. The controller shows satisfaction of all constraints, saturating the input for a number of samples and producing toward minimum-time v_y and θ_p trajectories.

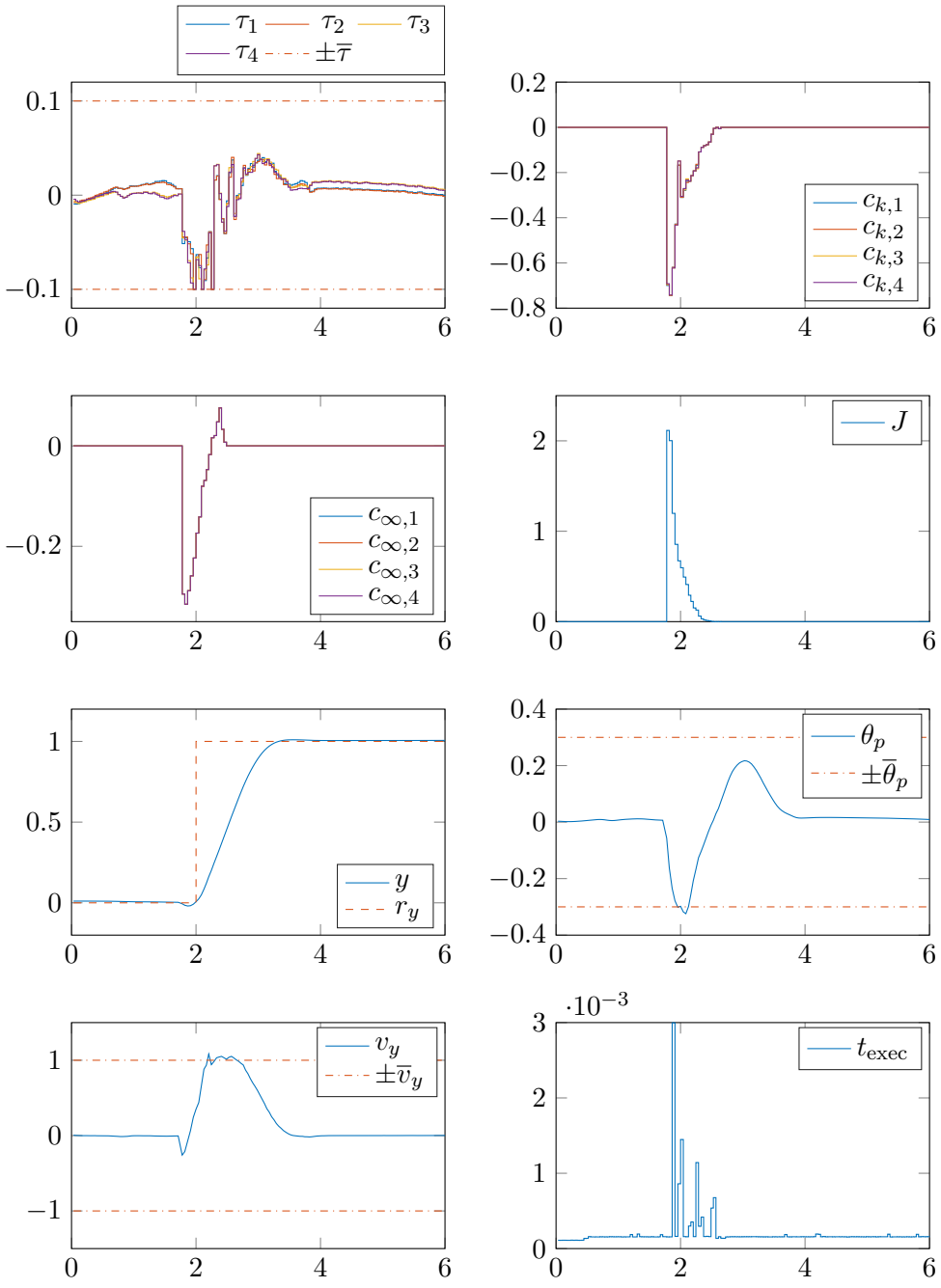


Figure 6.6: Experimental response to a step reference of $r_y = [0, 1]$m. This shows a strong similarity to that predicted in Figure 6.5, with only minor increases in constraint violation due to external disturbance. Execution time is found to be slightly faster, demonstrating the real-world feasibility of applying online constrained optimal control to this highly dynamic system.

applying online constrained optimal control to this highly dynamic system.

6.2.2 Step Reference Tracking - Lateral Translation

Figure 6.7 shows the system's response to an $r_x = [0, 1]$ step reference input, showing close to bang-bang control without any constraint violation. This demonstrates both the advantageous ability of this optimal control approach to produce close to minimum-time trajectories in the presence of constraints, and how by directly controlling wheel torques this controller is able to optimally saturate the input for a significant duration without any negative impact on closed-loop stability.

Figure 6.8 shows the response of the experimental system to this reference, again showing a very similar response to that predicted in simulation. This response does exhibit a small steady-state error, believed to be due to static friction in the Mecanum wheel roller bearings. This could be addressed in future work by the incorporation of integral action into the controller, likely through an output disturbance observer [114], or by an improved discontinuous form of friction compensation. Also shown in this figure are the tracked θ_p, v_y, and y trajectories, as in practice lateral motion generates large disturbances in these states. This shows peak deviations of $\theta_p = 0.027\,\text{rad}$ and $y = 1.5\,\text{cm}$, indicating that these disturbances are mostly successfully rejected, especially given that through the system's non-minimum phase property it is impossible to drive this error to zero in the presence of disturbance. This control performance is achieved whilst maintaining $t_{\text{exec}} < 2\,\text{ms}$.

As expected, a step reference applied to r_ϕ generates very similar trajectories to that in Figures 6.7 and 6.8, and is therefore omitted.

6.2.3 Constraint-Violating Disturbance Handling

Figure 6.9 demonstrates the controller's handling of large constraint-violating disturbances. For this scenario the reference is kept at the origin and the system is given an initial forward velocity $v_y = 2\overline{v}_y = 2\,\text{m}\,\text{s}^{-1}$, whilst maintaining all other initial states the origin. As expected, the controller is seen to briefly worsen constraint violation by increasing v_y to lean the platform towards the origin, maintaining $\theta_p \approx \overline{\theta}_p$ to decelerate as quickly as constraints allow, before maintaining $v_y \approx -\overline{v}_y$ until $y = 0$, with minimal overshoot.

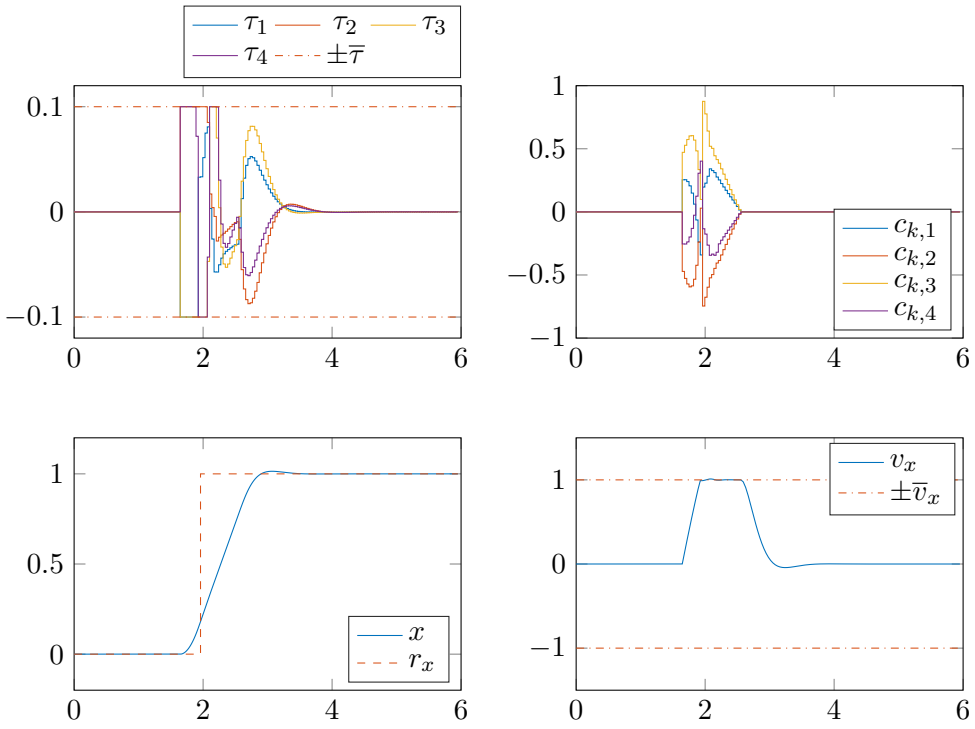


Figure 6.7: Simulated MPC response for a step reference of $r_x = [0, 1]$m. This shows close to bang-bang control without, any constraint violation. This demonstrates both the advantageous ability of this optimal control approach to produce close to minimum-time trajectories in the presence of constraints, and how by directly controlling wheel torques this controller is able to optimally saturate all four inputs for a large number of timesteps without any negative impact on closed-loop stability.

Figure 6.10 shows a similar scenario, though real-world initial conditions are somewhat harder to control, and so contain some visible error. Despite this, a very similar response is observed, again showing fast re-establishment and maintenance of constraint satisfaction, before asymptotically re-converging to the origin.

6.2.4 Time-Varying Trajectory Tracking

This controller provides no mechanism for directly incorporating feedforward velocity reference information as would be performed for a simple full-state feedback controller. The improved steady state tracking of a time-varying reference that

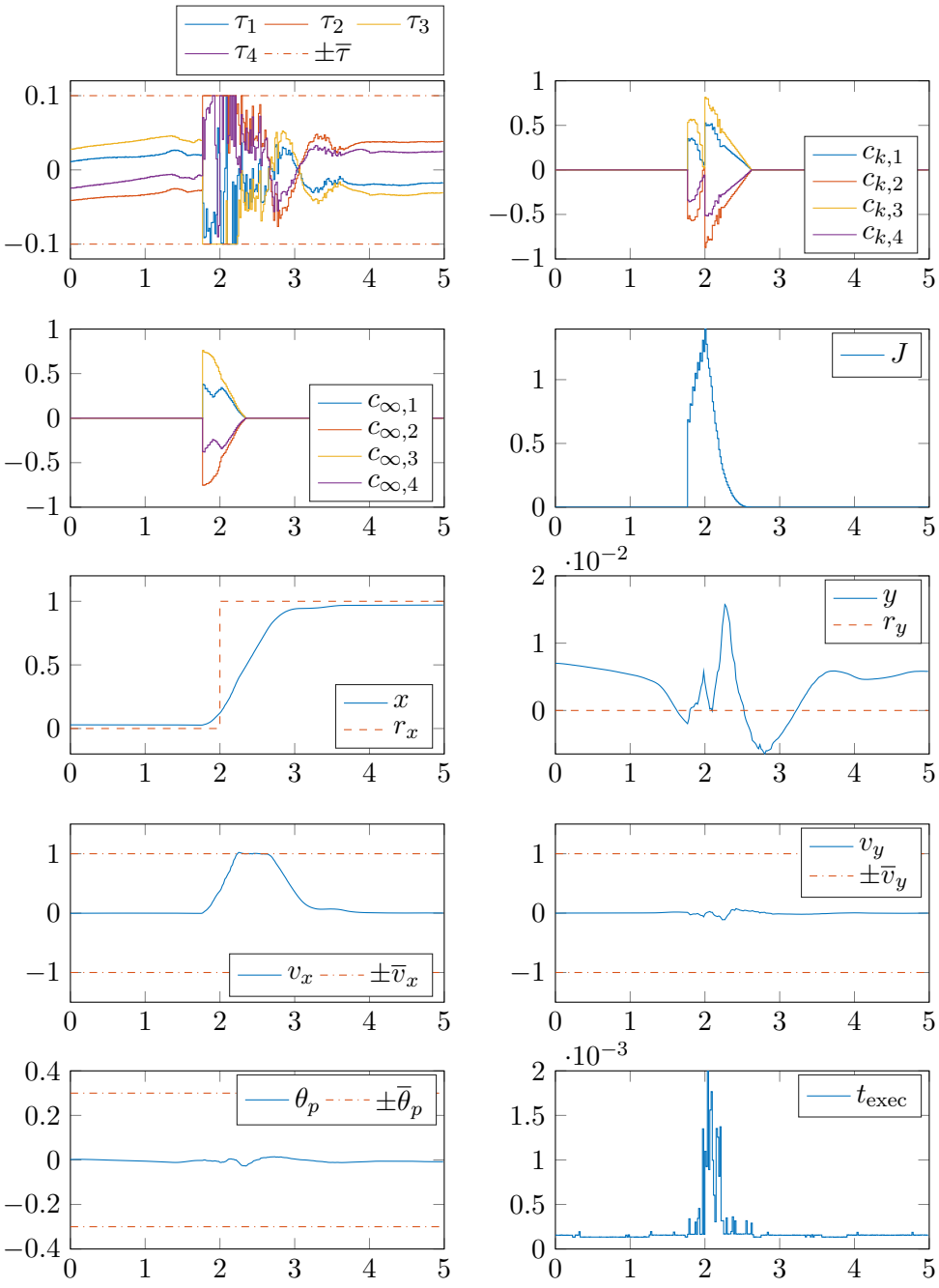


Figure 6.8: Experimental response to a step reference of $r_x = [0, 1]$m, showing a very similar response to that predicted in simulation in Figure 6.7. This response does now exhibit a small steady-state error, believed to be due to static friction in the Mecanum wheel roller bearings. The θ_p and y states show peak deviations of $\theta_p = 0.027\,\text{rad}$ and $y = 1.5\,\text{cm}$, indicating disturbance in these states generated during lateral motion is mostly rejected, especially given that through the system's non-minimum phase property it is impossible to drive this error to zero, even with perfect control.

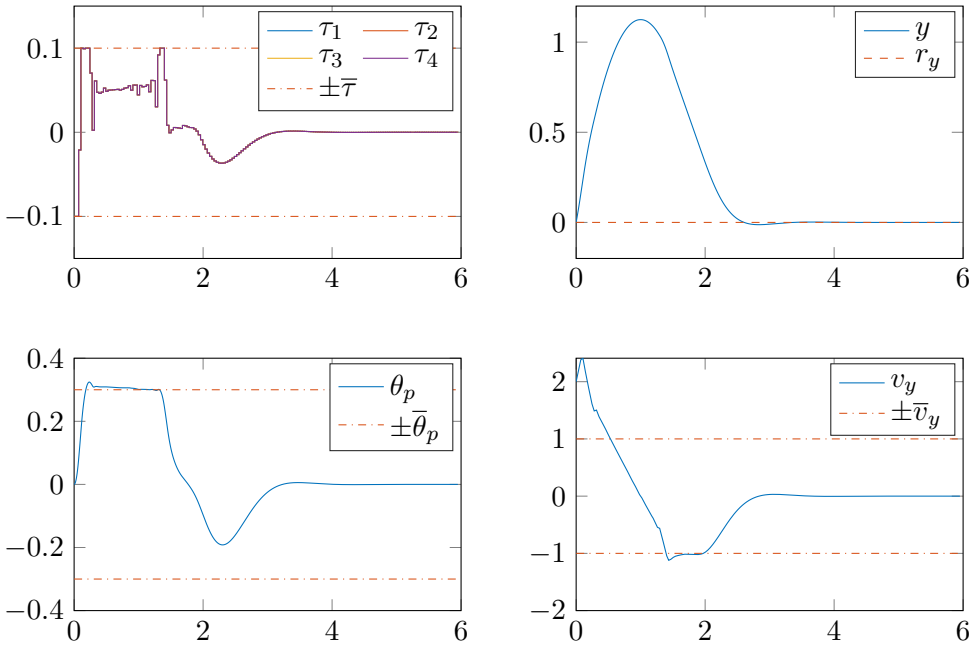


Figure 6.9: Simulated controller response in regulation scenario to an initial disturbance of $v_y = 2\overline{v}_y = 2\,\mathrm{m\,s^{-1}}$, demonstrating the controller's handling of large constraint-violating disturbances. For this scenario the reference is kept at the origin and the system is given an initial forward velocity $v_y = 2\overline{v}_y = 2\,\mathrm{m\,s^{-1}}$, whilst maintaining all other initial states at the origin. As expected, the controller is seen to briefly worsen constraint violation by increasing v_y to lean the platform towards the origin, maintaining $\theta_p \approx \overline{\theta}_p$ to decelerate as quickly as constraints allow, before maintaining $v_y \approx -\overline{v}_y$ until $y = 0$, with minimal overshoot.

this would bring is instead incorporated via the reference previewing mechanism, meaning tracking performance is related to choice of n_r. Figure 6.11 shows simulated position tracking error for a ramping y reference signal over increasing n_r. This shows a steady state tracking error for short reference previewing horizons, with this error vanishing around $n_r \approx 18$. This effect highlights a drawback of the dual-mode MPC approach; the use of a closed-loop prediction model means that when given a previewed section of a continuously changing reference the controller is optimising its future trajectory to come to rest at r_{k+n_c+1}, meaning it must plan for its state at $k = n_c + 1$ to lie within $\mathcal{S}_{MAS}$. In practise this means that the system can only increase v_y to a value from which it can enter $\mathcal{S}_{MAS}$ in n_c timesteps, which for the value of n_c used in the previous figures is

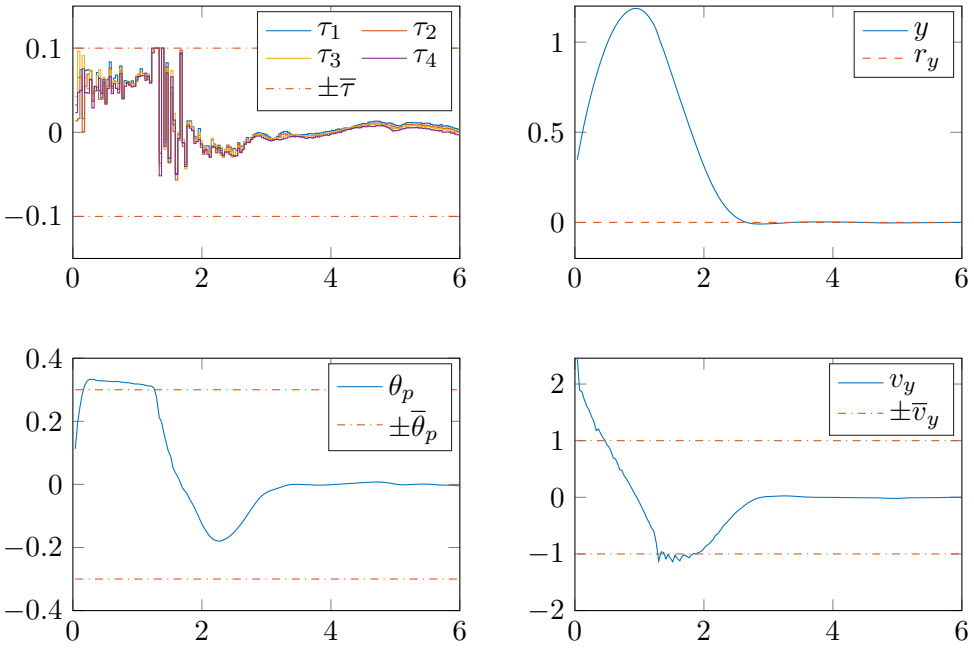


Figure 6.10: Experimental response in the regulation case to an initial velocity disturbance of approximately $v_y = 2\overline{v}_y = 2\,\mathrm{m\,s^{-1}}$. This shows a very similar response to that predicted by simulation in Figure 6.10, again showing fast re-establishment and maintenance of constraint satisfaction, before asymptotically re-converging to the origin.

insufficient to correctly track the given ramping reference trajectory. This results in the system lagging behind the desired trajectory, which in turn increases the effective stopping distance from the target steady state that the controller is able to use to enter $\mathcal{S}_{MAS}$ at $k = n_c + 1$. This position lag increases until the system has accumulated sufficient distance from r_{k+n_c+1} to be able to safely reach the velocity required to keep up with the moving reference.

This tracking error can be addressed by ensuring a reference previewing period that is of sufficient length to fully capture the transition of the system from any state within $\mathcal{S}_O$ into $\mathcal{S}_{MAS}$ without constraint violation. This value of n_c can be approximated by examining the system response in the regulation case to an initial forward velocity of $v_y = \overline{v}_y$, which indicates a stopping distance of 0.53 m, taking 0.56 s for the system to enter $\mathcal{S}_{MAS}$. Two approaches exist to ensure this; the reference previewing and control horizons can be increased to match the stopping time at $n_r = n_c = 28$, at a cost of greater computational

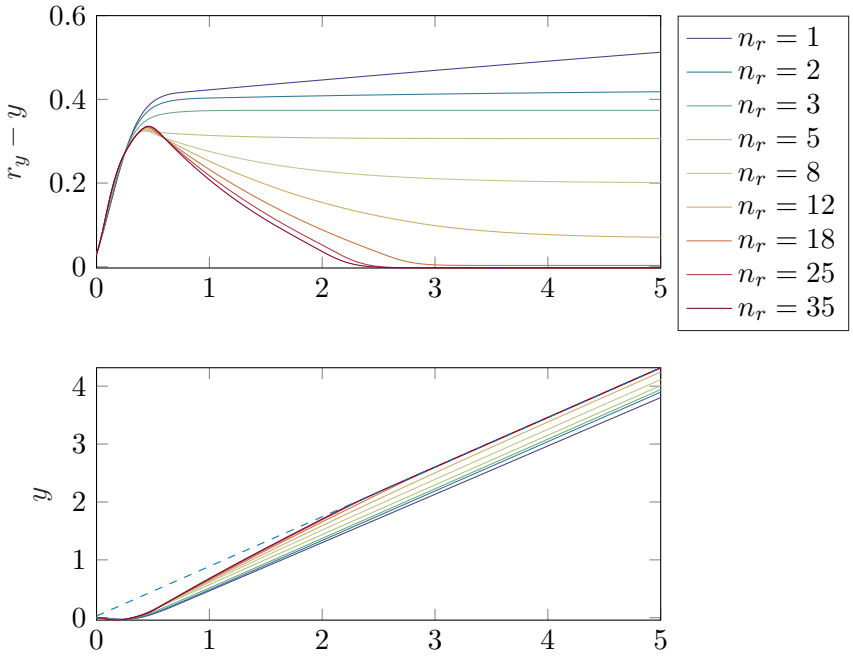


Figure 6.11: Simulated MPC r_y tracking error (top) for a linearly ramping position setpoint over varying n_r, with corresponding r_y reference (dashed) and state trajectories (bottom). This demonstrates how an insufficient reference previewing horizon can prevent asymptotic tracking of a time-varying r_y, with convergence only achieved for $n_c \gtrapprox 18$.

complexity, or the underlying gain K can be increased and the output constraint set enlarged in order to allow the system to reach the required steady state in less time. However, increasing K in turn decreases the size of $\mathcal{S}_{MCAS}$, as discussed in Section 6.1.9, as well as increasing sensitivity to noise and the excitation of unmodelled dynamics. Enlarging the output constraint set is also ineffective, as the same problem occurs, only at a larger velocity. For demonstration n_c and n_r are increased to $n_c = n_r = 28$, with the response to the same figure-of-eight trajectory also shown in Figure 6.12. This also allows a reduction of R in contrast to Section 6.1.9, as this larger control horizon already enlarges $\mathcal{S}_{MCAS}$ to encompass $\mathcal{S}_O$ with sufficient margin. However, this increase in control horizon increases worst case execution time to $t_{exec} = 6.4\,\text{ms}$, which in practise is close to being too slow for the real-time control of this particular system. The control horizon is therefore left unchanged, and a small amount of steady state tracking

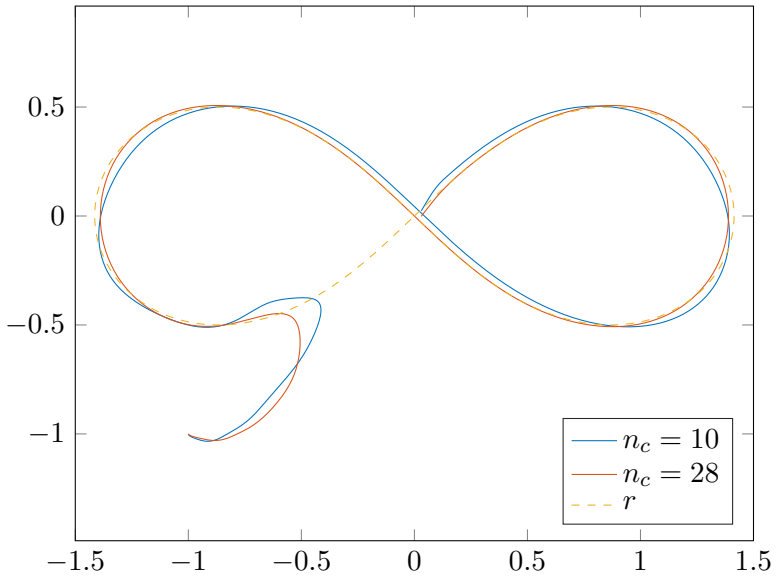


Figure 6.12: Simulated MPC state trajectories with $n_c = 10$ (blue) and $n_c = 28$ (red) for a figure-of-eight reference of 10 s duration and constant ϕ (yellow, dashed), starting at the origin with an initial direction of down and left. The system is initialised in a stationary pose at $(-1, -1)$, a tracking error that would result in constraint violation and wheel slip for a rudimentary controller.

error at large constant velocities is accepted.

6.2.5 Complex Trajectory Tracking

Finally, a figure-of-eight trajectory of 10 s duration with a large initial tracking error is used to demonstrate the ability of this controller to track complex trajectories whilst maintaining stability and feasibility for large reference deviations. In Figures 6.12 and 6.13 this is shown in simulation for both $n_c = 10$ and $n_c = 28$, though state trajectories are only plotted for the $n_c = 10$ case. In the $n_c = 10$ case a r_y tracking error as in Figure 6.11 is observed, resulting in the system following a distorted figure-of-eight. As expected, increasing n_c to $n_c = 28$ eliminates this error. A small r_x tracking error is also observed at the extremes of this reference. This is due to the controller optimising a trade-off between perfect tracking and the increased control effort required to achieve this, and is therefore an expected behaviour, and can be lessened by a reduction of R or increase of $Q_{1,1}$.

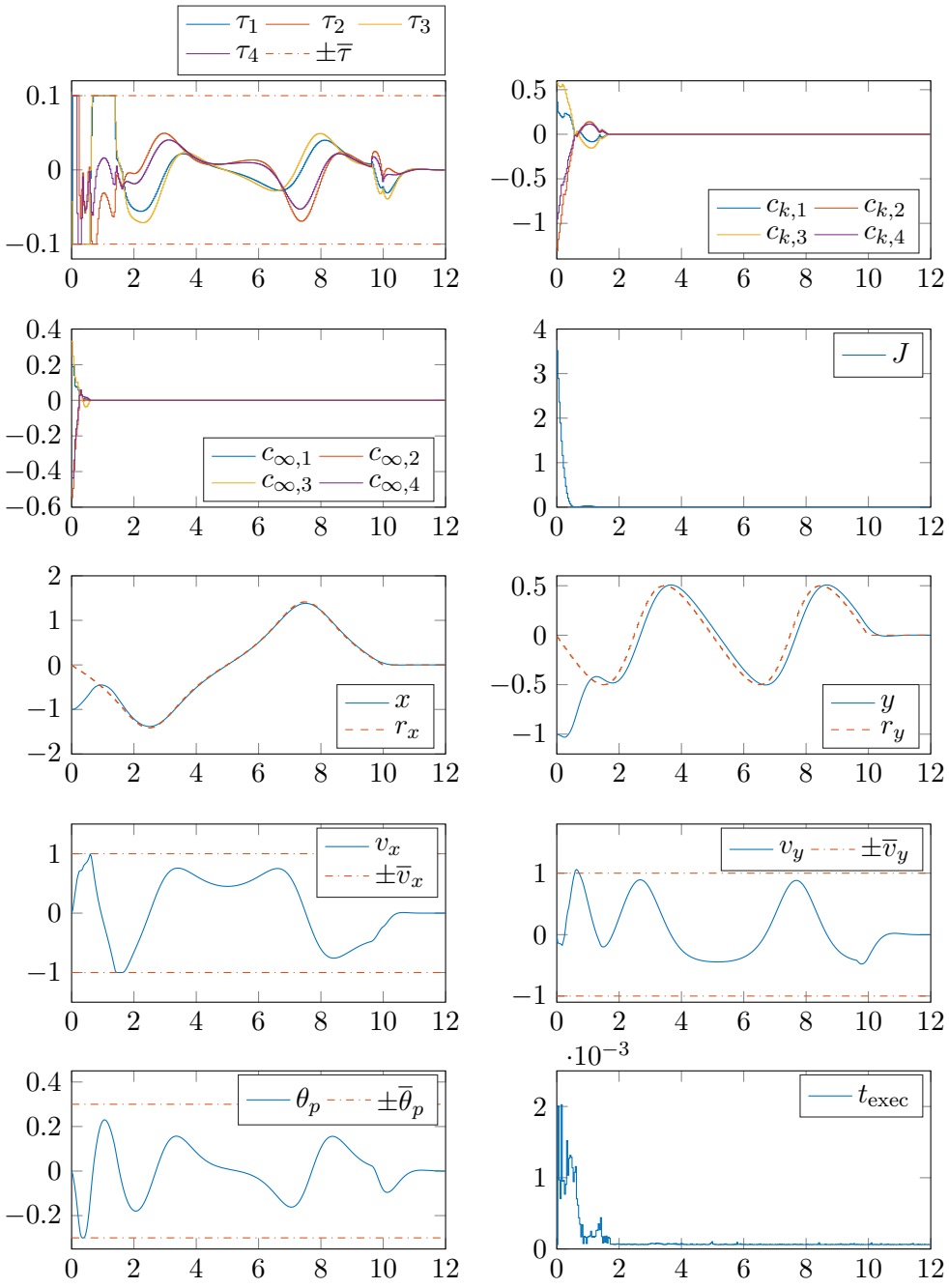


Figure 6.13: State trajectories for the figure-of-eight trajectory in Figure 6.12 with $n_c = 10$. An r_y tracking error is observed as in Figure 6.11, resulting in the system following a distorted figure-of-eight path. As expected, increasing n_c to $n_c = 28$ eliminates this error. A small r_x tracking error is also observed at the extremes of this reference due to the controller optimising a trade-off between perfect tracking and the increased control effort required to achieve this.

Figures 6.12 and 6.15 shows the experimental response to the same figure-of-eight trajectory with $n_c = 12$. This shows the same expected phase lag in the y state as in simulation, along with the same x error as in Figure 6.8, now also presenting as a lag behind the desired trajectory. A large spike in computation time occurs at the start of the experiment to $t_{\text{exec}} = 20\,\text{ms}$. This is due to the significant change in optimal solution that occurs at this point, requiring a large number of solver iterations to re-converge. Once this is solved subsequent control iterations are 'warm started', meaning the solver is initialised with the solution to the previous control iteration. Increasing the control horizon to $n_c = 28$ as in simulation in practice makes this initial computation time too large, causing instability, and hence this is not shown.

This scenario demonstrates the combined advantages of this MPC approach, in that both constraint satisfaction and stability are maintained throughout the correction of the large initial tracking error, where a rudimentary controller would yield significant constraint violation and loss of control, before continuing to yield good tracking of time varying references.

6.3 Conclusion

This chapter has demonstrated the first successful implementation of a real-time constrained model predictive controller for position and heading control of a Collinear Mecanum Drive. Through a redefinition of the nonholonomic constraints (3.1.5)-(3.1.7) and a reduction of the state vector this controller could be simplified to be applicable to a two-wheeled inverted pendulum, for which such a controller has not been successfully demonstrated either. This work is therefore novel in a more general sense than in the control of specifically just a Collinear Mecanum Drive.

Given the good experimental performance demonstrated by this controller on this small and highly dynamic prototype, future work would explore application of this controller to a larger system with slower dynamics. This would allow for longer optimisation execution times, therefore allowing a larger control horizon to be used to eliminate the tracking error demonstrated in Figures 6.11 and 6.15. Future work will also explore introducing move blocking as discussed in Section 6.1.9 to allow for a larger n_r and higher gain feedback for the same number of

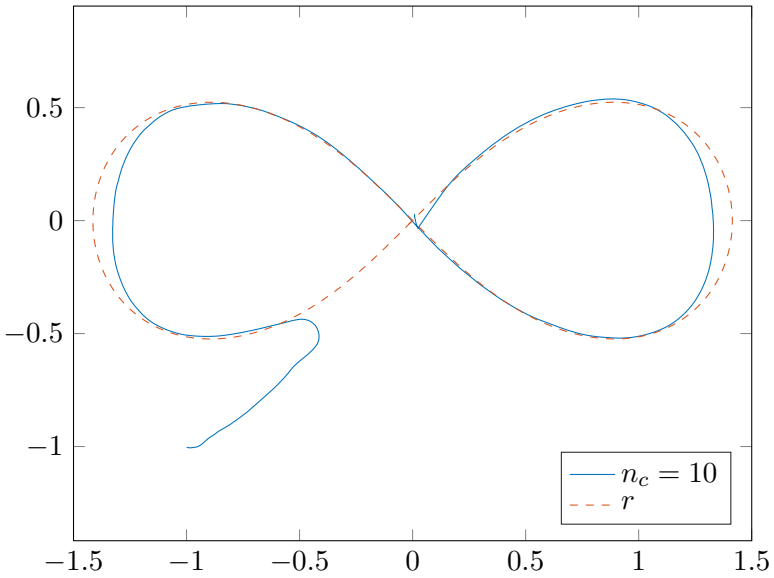


Figure 6.14: Experimental response to the same 10 s figure-of-eight trajectory as in Figures 6.13 and 6.12 with $n_c = 12$. This shows convergence to and tracking of the reference trajectory from a large initial tracking error. A small r_x remains at the extremes of the path, believed to be due to static friction effects in the Mecanum wheel rollers.

decision variables, along with exploring the size and distribution of these blocks.

The development of a high level planner capable of generating reference trajectories for this controller to follow would enable the navigation of a known map. This reference could be a series of discontinuous position waypoints, or it could be a 2D path returned by a sampling based planner such as RRT*, with some simple heuristic-derived timing law. In contrast with polynomial-based trajectory generation methods [73] this removes the requirement for the outer planner to specify a dynamically feasible and constraint satisfying trajectory between waypoints; here the dynamically feasible and constraint satisfying state and input trajectories are instead derived iteratively by the MPC. The MPC demonstrated here is better suited to this task than existing TWIP MPC implementations, as by maintaining feasibility across the full reference set waypoints can be placed at arbitrarily sparse intervals, rather than needing to consider the controller's feasible reference set. The embedding of input and output constraint satisfaction into the low level controller also improves safety and robustness in the event of delay

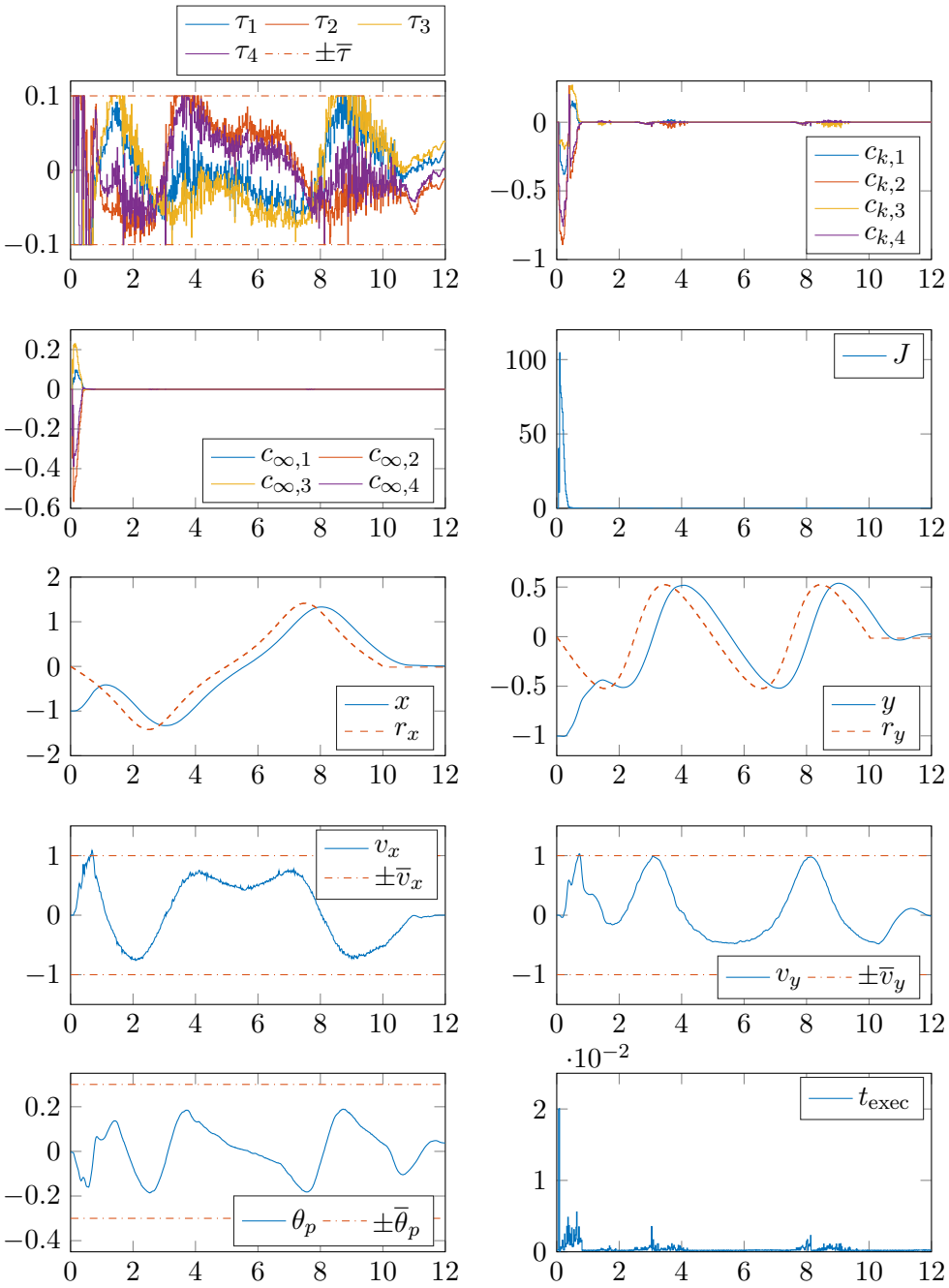


Figure 6.15: Experimental response to the same 10 s figure-of-eight trajectory as in Figures 6.13 and 6.12 with $n_c = 12$. This shows convergence toward and tracking of the reference trajectory from an initially large tracking error. Both r_x and r_y tracking errors are visible, believed to be due to static friction effects in the Mecanum wheel rollers and an insufficient control horizon.

or error in the high level planner, with the system guaranteed to safely come to rest at the end of the last specified reference. This also improves recovery in the event of significant disturbance, with tracking safely resumed whilst satisfying constraints, as opposed to relying on the high level planner to quickly generate a suitable 'recovery' trajectory. This is all achieved with a proof of convergence and stability for the linearised model, which given the small degree of nonlinearity over the constrained operating region can reasonably be assumed to extend to the full nonlinear system.

Chapter 7

Fast Online Trajectory Optimisation

This chapter demonstrates the final control approach of this thesis, in which a concept known as differential flatness is used to generate dynamically feasible state trajectories from optimally smooth geometric paths between waypoints. A differentially flat model of the CMD is derived, and the suitability of this method for the generation of state trajectories between waypoints is demonstrated. A novel approach to the generation of toward minimum-time velocity constrained trajectories is then developed, allowing the generation of dynamically feasible and velocity constrained state trajectories through multiple waypoints. This is again demonstrated in both simulation and on an experimental CMD prototype. Finally, a novel approach to polynomial trajectory optimisation is demonstrated, in which a concept known as sum-of-squares programming is used to yield an average 40% *reduction in constrained polynomial optimisation time over the state-of-the-art.*

The ability for dynamically balancing omnidirectional robots to plan dynamically feasible state trajectories through multiple temporally free position waypoints is a key precursor to enabling the autonomous navigation of an environment. This is a more challenging task than the simple kinematic planning that is commonly used to generate trajectories for statically-stable holonomic wheeled vehicles, as the underactuation property of dynamically balancing robots means they cannot be commanded to follow arbitrary trajectories through configuration space. The generated trajectories must therefore meet the desired navigation goals and constraints, whilst also evolving exactly on the surface of feasible trajectories satisfying one or more second-order nonholonomic constraints.

Trajectory optimisation methods for general constrained nonlinear systems can

typically be grouped into two categories: shooting methods, and transcription methods [31, 83, 127]. The single shooting method is the simplest form of trajectory optimisation, in which at each optimisation iteration a parametrised input function $u(t)$ and trajectory duration t_e are defined to allow the continuous Runge-Kutta integration of the full nonlinear dynamics over $t \in [0, t_e]$. Some suitable cost function can then be evaluated along the resulting state and input trajectories, and satisfaction of constraints can be evaluated. By numerically calculating gradients along each optimisation variable a search direction can be defined, allowing the iterative improvement of $u(t)$ and t_e toward a minimum. While these methods are suitable for systems with simple dynamics and with a good initial guess, when applied to highly nonlinear and unstable systems such as the CMD a small variation in the optimisation variables can yield vastly different state trajectories, making such an optimisation ill-conditioned. This can be greatly improved using multiple shooting methods, in which the trajectory is split into multiple sections, with equality constraints used to enforce continuity at segment boundaries. While reducing solution time, this method is still computationally expensive even for simple systems.

For complex nonlinear systems, transcription methods typically yield faster solutions [127]. These methods split the trajectory into multiple sections as in multiple shooting, but instead of simulating the full nonlinear dynamics between nodes the state and input dynamics are instead approximated using polynomial splines. These techniques have been applied to complex dynamic systems such as that in this thesis, with solution times in the region of hundreds of milliseconds for simple trajectories [83]. However, complexity and solve time rapidly increases for trajectories through multiple waypoints, still limiting the suitability of these methods for real-time planning in this application.

Trajectory optimisation can also be approached by extension of existing sampling based kinematic planners commonly used in ground vehicle and manipulator trajectory planning to incorporate dynamic constraint satisfaction, referred to as kinodynamic planning, of which kinodynamic RRT* is the most commonly encountered example [128, 129]. This method builds a random tree of trajectories in the system's configuration space, rooted at the system's initial state. By integrating a linearisation of the system dynamics between nodes all resulting trajectories through the tree are approximately dynamically feasible. However,

this too is a computationally expensive method, taking seconds to minutes to plan simple trajectories between waypoints for similar nonlinear systems [129].

An approach used in trajectory generation for ball-balancing robots optimises lean angle trajectories comprised of piecewise summed parametrised hyperbolic secant and cubic spline functions [74], forming a single shooting trajectory optimisation problem. However, this choice of basis function requires the solution of a relatively complex constrained nonlinear program, again at high computational cost, and through this basis can only optimise over a limited set of solutions. Nagarajan presents a further method for the planning of ball-balancing robot trajectories in which desired Cartesian trajectories in the external variables are defined using high-order polynomials $p(t)$. In recognising that in a neighbourhood of the origin there exists a linear mapping of steady state external variable accelerations $\ddot{p}(t)$ to the shape variables $q(t)$, i.e. the body lean angles, a linear mapping $\ddot{p}(t) = K_1 q(t)$, $\dddot{p}(t) = K_2 \dot{q}(t)$, $\ddddot{p}(t) = K_3 \ddot{q}(t)$ can be optimised to minimise the error $J = \|\ddot{p}(t) - \ddot{p}^*(t)\|_2^2$, where $p^*(t)$ represents the external variable acceleration trajectory derived from evaluation of the second order nonholonomic constraint imposed by the system's underactuation along $q(t)$ [82]. This optimisation forms a nonlinear program, with computation times of around 1 s for short trajectories, so can feasibly be implemented in real-time on slow moving systems. This method yields exactly dynamically feasible trajectories, at the expense of a small deviation from the infeasible desired Cartesian trajectory. As this linear map only exists in a neighbourhood of the origin it will decrease in validity for larger lean angles. This method is also applied to a system with a pair of 2-DOF arms, with minimal increase in problem complexity [130].

Despite this plethora of trajectory optimisation methods, none are able to provide sufficiently fast execution times so as to allow for the real-time planning required in this application, preventing fast reaction to dynamic obstacles and large disturbances.

A less computationally demanding alternative applies the concept of differential flatness to the planning problem, a technique that allows for state trajectories to be calculated algebraically from sufficiently continuously differentiable geometric trajectories in some possibly fictitious system outputs [20]. This is similar to Nagarajan's shape-space planner, but instead of optimising a linear mapping of external to shape variables to minimise tracking error, this map is instead model-

derived, constant, and potentially nonlinear. This allows the planning problem to be addressed in a top down manner by optimising trajectories in the system's outputs rather than its inputs, yielding less computationally demanding problem formulations than shooting or transcription methods. This subset of the trajectory planning literature mainly focuses on quadrotors, which are naturally differentially flat systems [22, 23], and therefore well suited to this type of planning, though some research has been undertaken into applying these techniques to ball-balancing robots [73, 81], in which smooth trajectories between position waypoints are generated for a single planar direction of a ballbot.

Differential flatness based trajectory planning techniques are therefore more suitable for this application. As no differential model of a CMD exists in the literature this is first derived.

7.1 Differentially Flat Model Derivation

From Chapter 3 the dynamic model of a Collinear Mecanum Drive can be described in terms of inertial frame positions p and local frame body velocities v in the form

$$M(p)\dot{v} + C(p, v)v + G(p) + Fv = B\tau \tag{7.1.1}$$

By treating these as four simultaneous equations, equation 4 of these can be divided by r_w and summed with equation 2 to isolate the system's internal dynamics by elimination of τ, yielding an expression of the form

$$f(\theta_p, \dot{\theta}_p, \ddot{\theta}_p, v_x, v_y, \dot{v}_y, \dot{\phi}) = 0 \tag{7.1.2}$$

This contains multiple orders of derivative of θ_p, and therefore does not allow an expression of θ_p as an algebraic function of the system's Cartesian positions and their derivatives, meaning the CMD's Cartesian positions and heading are not differentially flat outputs.

Shomin [73] showed that a single planar axis of a ball-balancing robot can be approximately differentially flattened by a combination of model simplification and a flat output of the form $S = y + \lambda\theta_p$. This technique can be applied to a CMD and extended to the fully coupled nonlinear model by choosing the system's

flat outputs as

$$S_1 = x - \sin(S_3)\lambda\theta_p \tag{7.1.3a}$$
$$S_2 = y + \cos(S_3)\lambda\theta_p \tag{7.1.3b}$$
$$S_3 = \phi \tag{7.1.3c}$$

with some time invariant λ. The system states x, y, v_x, v_y, $\dot{v}_y$, and all derivatives of ϕ can be trivially derived by differentiation and rotation of the flat outputs as

$$x = S_1 + \sin(S_3)\lambda\theta_p \tag{7.1.4a}$$
$$y = S_2 - \cos(S_3)\lambda\theta_p \tag{7.1.4b}$$
$$\phi = S_3 \tag{7.1.4c}$$
$$v_x = \cos(S_3)\dot{S}_1 + \sin(S_3)\dot{S}_2 + \lambda\theta_p\dot{S}_3 \tag{7.1.4d}$$
$$v_y = \cos(S_3)\dot{S}_2 - \sin(S_3)\dot{S}_1 - \lambda\dot{\theta}_p \tag{7.1.4e}$$
$$\dot{v}_y = -\cos(S_3)\dot{S}_1\dot{S}_3 - \sin(S_3)\dot{S}_2\dot{S}_3 - \sin(S_3)\ddot{S}_1 + \cos(S_3)\ddot{S}_2 - \lambda\ddot{\theta}_p \tag{7.1.4f}$$
$$\dot{\phi} = \dot{S}_3 \tag{7.1.4g}$$
$$\ddot{\phi} = \ddot{S}_3 \tag{7.1.4h}$$

Substituting (7.1.4a)-(7.1.4h) into (7.1.2) and performing a small angles approximation of $\sin(\theta_p)$ and $\cos(\theta_p)$ about $\theta_p = 0$ yields the expression

$$\begin{aligned} 0 = {} & \theta_p\left(a + e\dot{S}_3^2 + f\lambda\dot{S}_3^2\right) + c\theta_p\dot{\theta}_p^2 - b\dot{\theta}_p\lambda - \ddot{\theta}_p\left(g + d\lambda\right) \\ & - \sin(S_3)\left(b\dot{S}_1 + d\ddot{S}_1 + h\dot{S}_2\dot{S}_3\right) + \cos(S_3)\left(b\dot{S}_2 + d\ddot{S}_2 - h\dot{S}_1\dot{S}_3\right) \end{aligned} \tag{7.1.5}$$

where a to h are time invariant positive constants[1]. $\ddot{\theta}_p$ can be eliminated from this expression by selection of $\lambda = -g/d$. It is a safe assumption that the centripetal force acting on the θ_p generalised coordinate due to $\dot{\theta}_p$ will always be small relative to other forces, so the term $c\theta_p\dot{\theta}_p^2$ can be omitted. Finally, the term $-b\dot{\theta}_p\lambda$ is found to vanish when rolling friction is not considered, i.e. $v_{rw} = 0$, so from the knowledge that rolling friction has only a small effect on system dynamics it is assumed that this term can be omitted without significant impact

[1] redefined from the definitions in previous chapters

on the accuracy of the differentially flat model. This allows for solution of θ_p as

$$\theta_p = \frac{\sin(S_3)\left(b\dot{S}_1 + d\ddot{S}_1 + h\dot{S}_2\dot{S}_3\right) - \cos(S_3)\left(b\dot{S}_2 + d\ddot{S}_2 - h\dot{S}_1\dot{S}_3\right)}{a + e\dot{S}_3^2 - f\,(g/d)\,\dot{S}_3^2} \tag{7.1.6}$$

in which all differentials of θ_p have been eliminated, converting this from an ODE to a simple algebraic equation.

$\dot{\theta}_p$ and $\ddot{\theta}_p$ can then be determined by differentiation of θ_p, thus algebraically defining all system states in terms of the flat outputs $(S_1, S_2) \in \mathcal{C}^4$ and $S_3 \in \mathcal{C}^3$. Finally, as state trajectories θ_p and $\dot{\theta}_p$ are now known to evolve with approximate satisfaction of the system's nonholonomic dynamic constraint, the associated input trajectories τ can be derived by inversion of the system dynamics in (7.1.1) using the pseudo-inverse B^+ and substitution with states in terms of S. This inversion only exists for feasible accelerations $\dot{v}$, but as these accelerations are to be derived from the differentially flattened model it is assumed that they will be sufficiently close to the feasible set for this assumption to be valid, and thus the inverse exists. The resulting expression is too large to be printed here and is therefore omitted.

Note that this method of differential flattening introduces a singularity at

$$\dot{S}_3 = \pm\sqrt{\frac{-a}{e - f\,(g/d)}} \tag{7.1.7}$$

and is divergent as $\dot{S}_3 \to \pm\sqrt{\frac{-a}{e-f(g/d)}}$, which for the parameters in Table 4.2 occur at $\dot{S}_3 = \pm 23.8\,\mathrm{rad\,s^{-1}}$. Fortunately this rate of rotation is unlikely to occur in any real-world scenario.

This model is demonstrated in Figure 7.1, in which corresponding state and input trajectories are derived using the differentially flat model from a trajectory in S_2 representing a translation of $S_2 = 0$ to $S_2 = 1$ in 2 s, with the first to fourth derivatives of S_2 constrained to zero at $t = 0$ and $t = 2$. Note how despite the flat output being monotonic, the y state trajectory correctly first translates away from the target in order to start leaning towards the target, and likewise at the end of the trajectory, thus capturing the non-minimum phase behaviour present in this system.

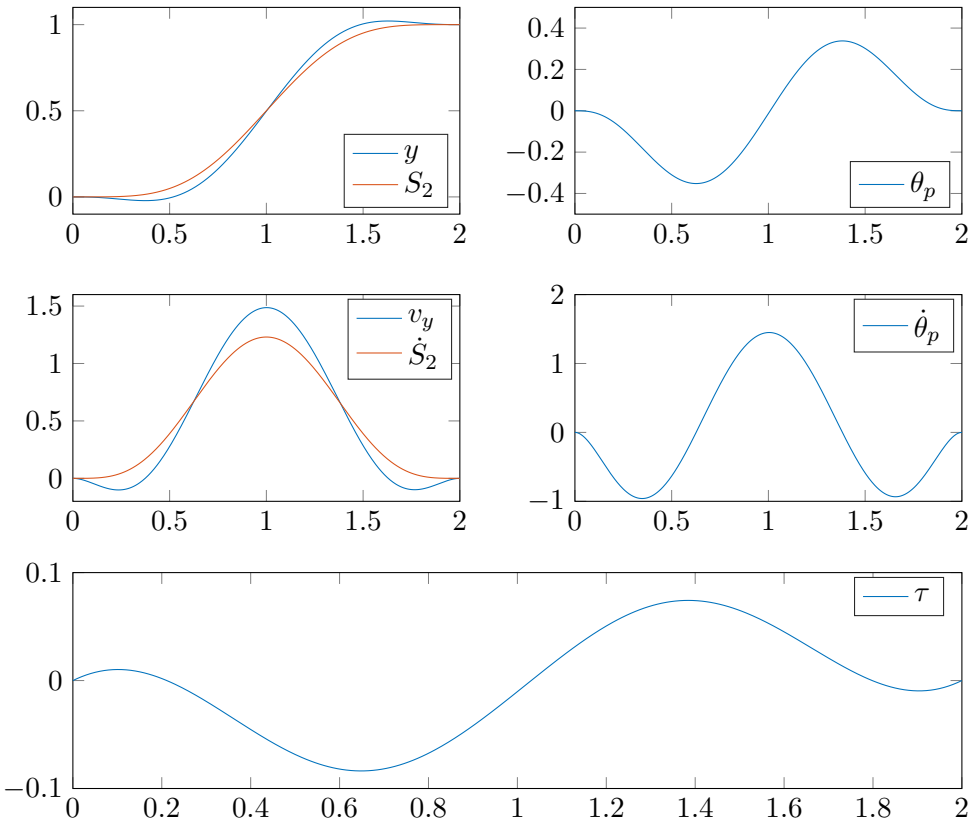


Figure 7.1: A trajectory in S_2 for a translation from $S_2 = 0$ to $S_2 = 1$ in 2 s, with corresponding state and input trajectories derived from the differentially flat model. The first to fourth derivatives of S_2 are constrained to zero at $t = 0$ and $t = 2$ so that this represents a rest-to-rest trajectory. The other two flat outputs remain at rest. All resulting τ_i trajectories are identical, so only one is shown.

7.2 Closed Loop Trajectory Tracking

While perfectly dynamically feasible state and input trajectories can in theory be tracked in open-loop, in reality the system will quickly deviate from the target trajectory. This is due to model parameter uncertainty, unmodelled dynamics, external disturbances, and in this case the simplifications required in the derivation of the differentially flat model, all compounded by the unstable dynamics of the open-loop system about the upright equilibrium. A fast tracking controller is therefore required to regulate the error between the current state and time

varying optimal state trajectories to zero. As asymptotic convergence can only be achieved with a perfect error model, integral action is incorporated to ensure convergence.

Consider the error dynamics obtained by rotating Cartesian position error and its integral into the local body frame

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \\ e_{10} \\ e_{11} \end{bmatrix} = \begin{bmatrix} R_{EB}^T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & R_{EB}^T & 0 \\ 0 & 0 & 0 & I_{6\times 6} \end{bmatrix} \begin{bmatrix} \int (x_r - x)\ dt \\ \int (y_r - y)\ dt \\ \int (\phi_r - \phi)\ dt \\ x_r - x \\ y_r - y \\ \phi_r - \phi \\ \theta_{pr} - \theta_p \\ v_{xr} - v_x \\ v_{yr} - v_y \\ \dot{\phi}_r - \dot{\phi} \\ \dot{\theta}_{pr} - \dot{\theta}_p \end{bmatrix} \tag{7.2.1}$$

with derivative

$$\dot{e} = \begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \\ \dot{e}_4 \\ \dot{e}_5 \\ \dot{e}_6 \\ \dot{e}_7 \\ \dot{e}_8 \\ \dot{e}_9 \\ \dot{e}_{10} \\ \dot{e}_{11} \end{bmatrix} = \begin{bmatrix} e_2\dot{\phi} + e_4 \\ -e_1\dot{\phi} + e_5 \\ e_6 \\ e_5\dot{\phi} + v_{xr} - v_x \\ -e_4\dot{\phi} + v_{yr} - v_y \\ \dot{\phi}_r - \dot{\phi} \\ \dot{\theta}_{pr} - \dot{\theta}_p \\ \dot{v}_{xr} - \dot{v}_x \\ \dot{v}_{yr} - \dot{v}_y \\ \ddot{\phi}_r - \ddot{\phi} \\ \ddot{\theta}_{pr} - \ddot{\theta}_p \end{bmatrix} \tag{7.2.2}$$

Substituting system dynamics of the form $\dot{x}_i = f_i(x, u),\ i \in [1\ldots 8]$ yields the

error dynamics

$$\dot{e}_1 = e_2 f_3(x,u) + e_4 \tag{7.2.3a}$$

$$\dot{e}_2 = -e_1 f_3(x,u) + e_5 \tag{7.2.3b}$$

$$\dot{e}_3 = e_6 \tag{7.2.3c}$$

$$\dot{e}_4 = e_5 f_3(x,u) + \cos(\phi) f_1(x_r,u_r) + \sin(\phi) f_2(x_r,u_r) - \cos(\phi)(f_1(x,u) - \sin(\phi) f_2(x,u) \tag{7.2.3d}$$

$$\dot{e}_5 = -e_4 f_3(x,u) - \sin(\phi) f_1(x_r,u_r) + \cos(\phi) f_2(x_r,u_r) + \sin(\phi)(f_1(x,u) - \cos(\phi) f_2(x,u) \tag{7.2.3e}$$

$$\dot{e}_j = f_i(x_r,u_r) - f_i(x,u), \qquad i = [3\,..\,8], \qquad j = i+3 \tag{7.2.3f}$$

in which substituting $x = x_r - e$ and $u = u_r - e_u$ yields an expression of the form

$$\dot{e} = F(x_r, u_r, e, e_u, e_i) \tag{7.2.4}$$

where

$$e_i = \int_0^t \begin{bmatrix} x_r - x \\ y_r - y \\ \phi_r - \phi \end{bmatrix} dt \tag{7.2.5}$$

Linearising the error dynamics about $e = 0$, $e_u = 0$, and $e_i = 0$ yields the time varying linear error model

$$\dot{e} = A(x_r,u_r)e + B(x_r,u_r)e_u \tag{7.2.6}$$

in which the error dynamics are linearised about the desired trajectory.

The goal of the controller is to regulate $e \to 0$, $e_i \to 0$, $e_u \to 0$. As $x_r(t)$ and $u_r(t)$ are to be derived from the differentially flat model they can be assumed to be dynamically feasible, meaning $e = 0$ can be maintained in steady state and therefore represents an equilibrium. This allows the design of a time-varying infinite-horizon LQR, and thus the optimal algebraic definition of K in the control law $u = Ke + u_r$. While this linearisation decreases in accuracy with deviation from the desired trajectory, it is assumed that any substantial deviation would

be accounted for by online trajectory replanning, as significant deviations may result in violation of constraints that would have otherwise been satisfied had the system correctly followed the desired trajectory.

7.3 Trajectory Generation using Optimally Smooth Polynomials

With a differentially flat model, it is now possible to calculate feasible state trajectories from any geometric paths in the flat outputs satisfying $(S_1, S_2) \in \mathcal{C}^4$, $S_3 \in \mathcal{C}^3$. This is typically achieved by representing the transition of a flat output from one state to another using parametrised univariate polynomial basis functions, such as representing a path between two waypoints or a change in velocity. While from the unisolvence theorem [131] it is known that there always exists a unique polynomial of at most degree $n_w - 1$ that can be used to describe a path between n_w waypoints in the absence of constraints, the large monomial powers required for paths through more than a handful of waypoints quickly lead to poor numerical conditioning. It is therefore better to describe trajectories through multiple waypoints in a piecewise continuous fashion using multiple chained polynomials of lower degree, with equality constraints enforced on a finite number of derivatives at the boundaries between these polynomial segments.

Trajectories in each flat output with $j \in [1 \,..\, n_w]$ waypoints w_j are therefore defined as piecewise functions using $i \in [1 \,..\, n_w - 1]$ univariate polynomials $p_i(t)$ each defined over $t \in [0, T_i]$, with constraints on the initial and terminal values of the zeroth to n_d derivatives enforced at $t = 0$ and $t = T_i$. Segment transitions are constrained to occur at w_i, and continuity of segment derivatives up to order n_d are enforced at segment transitions. The first to $n_d th$ derivatives at w_1 are constrained in order for the trajectory to evolve from the system's initial state, and the zeroth to $n_d th$ derivatives at w_{n_w} are constrained to zero

for the trajectory to end with the system at rest.

$$p_1(0)^{(m)} = w_1^{(m)} \qquad \text{for } m = [1, n_d] \tag{7.3.1a}$$

$$p_{n_w-1}(T_{n_w-1})^{(m)} = w_n^{(m)} \qquad \text{for } m = [1, n_d] \tag{7.3.1b}$$

$$p_i(T_i)^{(m)} = p_{i+1}(0)^{(m)} \qquad \text{for } i = [1, n_w - 2], \quad m = [2, n_d] \tag{7.3.1c}$$

$$p_i(T_i) = w_i \qquad \text{for } i = [1, n_w - 1] \tag{7.3.1d}$$

Work applying differentially flat trajectory generation to quadrotors [22, 23] argues that desirable smooth trajectories can be generated by minimising the 2-norm of the kth derivative of a system of order $k-1$, as this acts to penalise changes in control action. This is important for quadrotors, as their thrust response to setpoint changes is not instantaneous due to propeller inertia. This yields the cost function for a single flat output

$$J = \sum_{i=1}^{n_w-1} \int_0^{T_i} \left(\frac{d^k p_i(t)}{dt^k} \right)^2 dt \tag{7.3.2}$$

Segment continuity at boundaries is enforced for the zeroth to $n_d th$ derivatives, where n_d should be sufficiently large so as to enforce piecewise continuity of the resulting input trajectories, and is therefore typically chosen to equal the order of the highest order flat output differential in the differentially flat input function.

7.3.1 Polynomial Cost and Piecewise Smoothness Requirements

In examining the differentially flattened inverse dynamics function $\tau = f(S)$ it is apparent that S_1 and S_2 appear up to their fourth derivatives, the snap of the outputs, and S_3 appears up to its third derivative, the jerk of the output. Following the argument used in the quadrotor literature $k = 5$ is selected for S_1 and S_2, minimising the crackle of the output, and constraints are enforced up to $n_d = 4$. Similarly, for S_3 $k = 4$ and $n_d = 3$ are selected. However, in this application it could be argued that due to wheel traction limitations and a desire to minimise energy usage it may be more sensical to minimise the kth derivative of a system of order k, as while making the input trajectory less smooth, this will act to minimise $\|\tau\|_2^2$ to a greater degree than the previous scheme. Also, as the flat expression for $\ddot{\theta}_p$ contains coefficients of $S_1^{(4)}$ and $S_2^{(4)}$, and as it is known

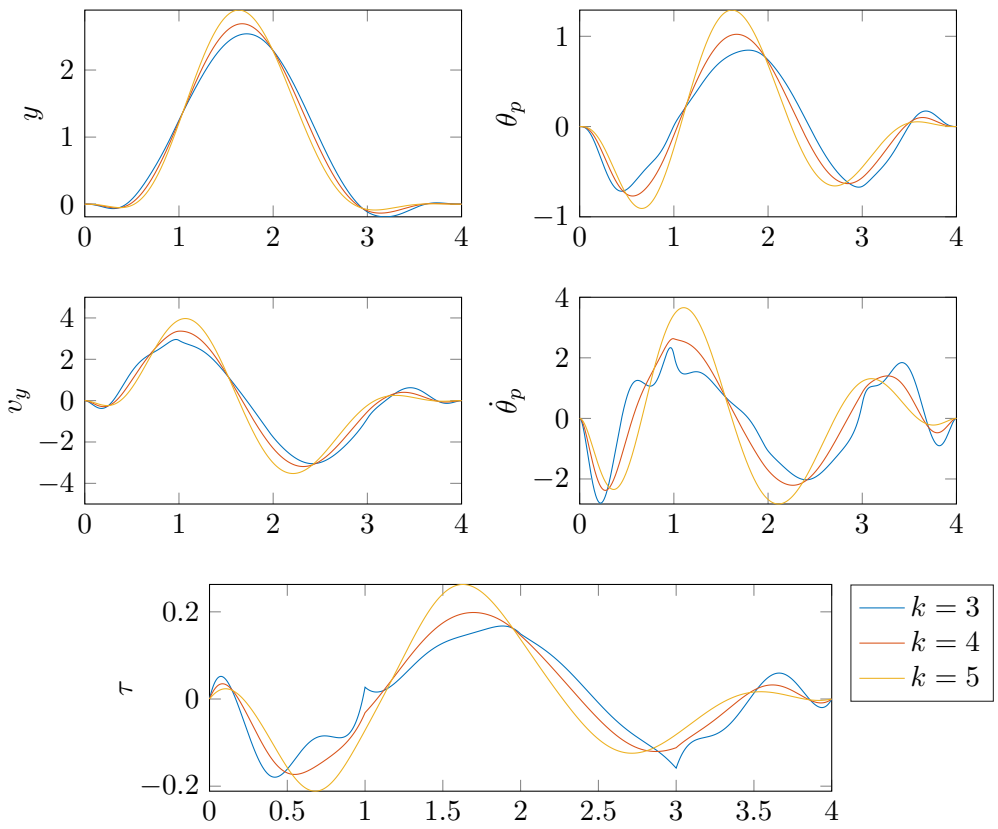


Figure 7.2: A comparison of choices of k in the cost function (7.3.2) for a trajectory through six random waypoints with $k = 3$ (blue), $k = 4$ (red), and $k = 5$ (yellow). The differentially flat model is used to derive state and input trajectories from these polynomials, demonstrating the effect of polynomial smoothness on the actual system trajectories. Clearly $k = 3$ yields non-smooth τ trajectories, and $k = 4$ yields lower peak lean angles than $k = 5$.

that $\ddot{\theta}_p \propto \sin(\theta_p)$, minimising the snap of S_1 and S_2 also acts to reduce the peak lean angle of the trajectory. Both of these schemes are compared in Figure 7.2, along with $k = 3$. As predicted, a choice of $k = 4$ shows reduced peak θ_p and τ trajectories over $k = 5$, whilst maintaining relatively smooth state trajectories. However, a large reduction in smoothness is apparent in reducing k further to $k = 3$, so by this metric $k = 4$ appears to be the most suitable choice.

7.3.2 Polynomial Degree Requirements

Each independent equality constraint increases the required polynomial degrees of freedom by one. For trajectories with fully constrained initial and terminal conditions through $n_w - 2$ intermediary waypoints the minimum average degree of the polynomial segments can be found as

$$\deg(p) \geq \frac{2(n_d+1) + (n_w-2)(n_d+2)}{n_w - 1} - 1 \tag{7.3.3}$$

with limits

$$\deg(p) \geq \begin{cases} 2n_d + 1 & \text{for} \quad n_w = 2 \\ n_d + 1 & \text{for} \quad n_w \to \infty \end{cases} \tag{7.3.4}$$

Similarly, if only the zeroth derivative is constrained at the terminal waypoint, such as in a receding horizon planning scenario, this requirement is reduced to

$$\deg(p) \geq \frac{(n_d+1) + 1 + (n_w-2)(n_d+2)}{n_w - 1} - 1 \tag{7.3.5}$$

$$\geq n_d + 1 \tag{7.3.6}$$

While $\deg(p)$ could be adapted with change in n_w, for simplicity is it decided that $\deg(p) = 2n_d + 1 \; \forall \; n_w \geq 2$ in order to maintain feasibility when $n_w = 2$ with fully constrained initial and terminal states. For $n_w > 2$ the problem is therefore underdetermined, so the extra DoF can be used to better minimise the cost function. Future work could explore reducing polynomial degree as n_w increases, or even using segments of varying degree, which would decrease problem complexity at some increase in trajectory cost.

7.3.3 QP Formulation

The cost function (7.3.2) is found to be quadratic in the polynomial coefficients, so by concatenating these into a vector (7.3.2) can be rewritten in the quadratic form

$$J = x^T H x \tag{7.3.7}$$

where x contains stacked coefficients of polynomials 1 to $n_w - 1$ as

$$x = \begin{bmatrix} p_{1,1} & \dots & p_{1,\deg(p)} & \dots & p_{n_w-1,1} & \dots & p_{n_w-1,\deg(p)} \end{bmatrix}^T \tag{7.3.8}$$

and with $H \succeq 0$. The constraints (7.3.1) can be rewritten in the linear form $Ax = b$ where $A \in \mathbb{R}^{n \times m},\ m < n,\ \text{rank}(A) = m$, defining the problem in the standard quadratic program form

$$\min_x\ x^T H x \qquad s.t.\ Ax = b \tag{7.3.9}$$

This allows a unique solution for the globally optimally smooth piecewise continuous trajectory using iterative quadratic programming techniques such as MATLAB's `quadprog`.

Alternatively, as this is a purely equality constrained quadratic program, it is possible to use QR decomposition to find a faster non-iterative solution. By QR decomposition A can be rewritten as

$$A^T = \begin{bmatrix} \hat{Q} & Q_N \end{bmatrix} \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} \tag{7.3.10}$$

which allows $Ax = b$ to be rewritten as

$$\hat{Q}^T \hat{R}^T x = b \tag{7.3.11}$$

x can then be split into its fixed and free DoF u and v as

$$x = \hat{Q}u + Q_N v \tag{7.3.12}$$

which by substitution into (7.3.11) allows the unique solution of u as

$$u = \hat{R}^{-T} b \tag{7.3.13}$$

and therefore a partial solution of x as

$$x = Q_N v + \hat{Q}\hat{R}^{-T} b \tag{7.3.14}$$

Substituting this into the original quadratic cost function then allows solution of

v as

$$v = -\left(Q_N^T H Q_N\right)^{-1}\left(\hat{Q}\hat{R}^{-T} b H Q_N\right) \tag{7.3.15}$$

which by substitution into (7.3.14) yields a numerical solution for x with a sub-millisecond execution time, even for very large numbers of waypoints. This method is also numerically more robust than iterative solvers, typically achieving a higher degree of optimality.

7.3.4 Experimental Unconstrained Trajectory Tracking

To demonstrate the accuracy of the differentially flat model, and the suitability of describing flat output trajectories using polynomials, the trajectory shown in Figure 7.1 is tracked in simulation, with the resulting state and input trajectories shown in Figure 7.3. These trajectories are generated from a single nonic polynomial in S_2, with constraints enforced up to the fourth derivative, uniquely defining the polynomial. In simulation there is clearly minimal deviation from the planned state trajectories, indicating that they are nearly exactly feasible, and thus the differentially flat model is of sufficient accuracy for this application. The same trajectories are then tracked experimentally, with state and input trajectories given in Figure 7.4. This shows more deviation from the planned trajectories than in simulation, though this is to be expected given the complex unmodelled dynamics acting on the system. Despite these errors, minimal position overshoot is observed, and the experimental trajectories strongly resemble that which was planned.

A more complex trajectory is shown in Figure 7.5, in which a pirouetting path is followed from $(x, y, \phi) = 0$ at $t = 0$ to $(x, y, \phi) = (0, 2, 2\pi)$ at $t = 4$. This shows good tracking of the x, y, v_x, v_y, ϕ, and $\dot{\phi}$ states, though while their shapes are similar the θ_p and $\dot{\theta}_p$ state trajectories are poorly tracked. This is both acceptable and expected; the system's non-minimum phase property means that any deviation of v_y or its integral from the planned dynamically feasible trajectory necessitates a significant deviation of θ_p from its reference if tracking is to be resumed. It is therefore through this θ_p tracking error that good position tracking of the remaining states is achieved. This experiment demonstrates how this planning approach is able to generate complex coupled state trajectories from a simple polynomial basis in the decoupled flat outputs.

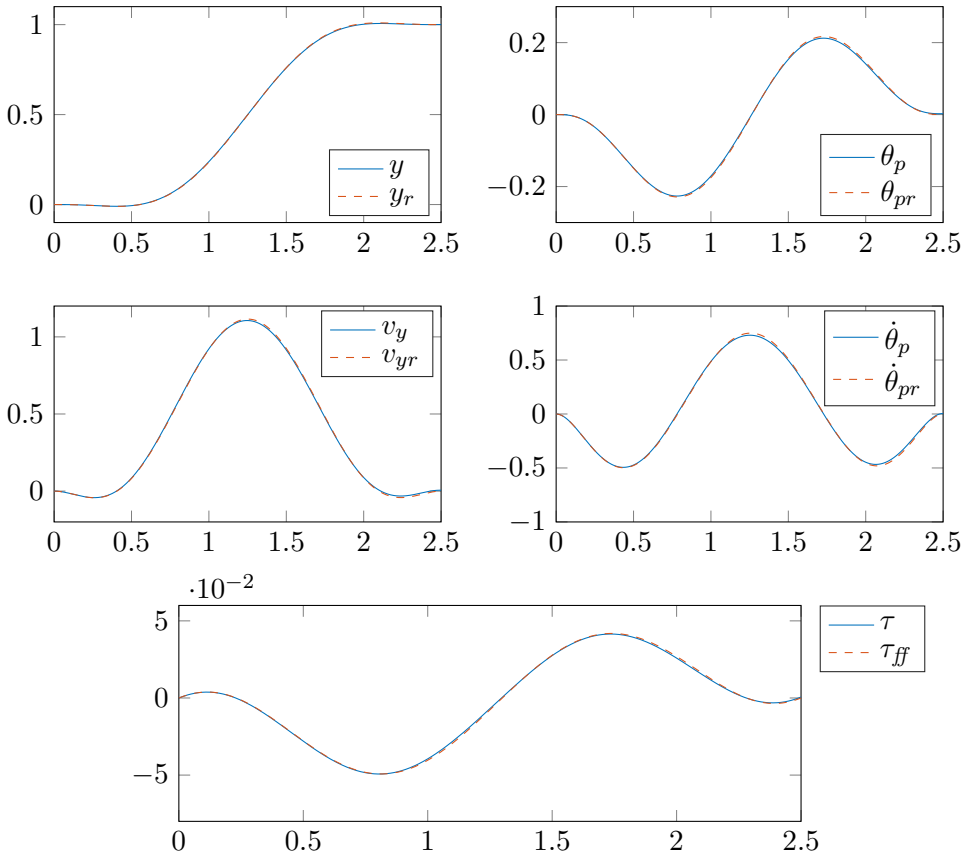


Figure 7.3: Simulated tracking of the trajectory in Figure 7.1. The near-absence of tracking error indicates that the differentially flat model generates close to exactly dynamically feasible state and input trajectories for the isolated θ_p and v_y subsystems, at least for the range of θ_p angles present in this trajectory. All τ and τ_{ff} trajectories are equal, so only one of each is shown.

7.3.5 Selection of Segment Durations

The choice of segment durations T_i has a large effect on the resulting shape of the optimised trajectories. This is demonstrated in Figure 7.6, in which the second segment duration of a trajectory through three waypoints is varied from half of to equal to the duration of the first segment, showing how elaborate paths can be taken as segment durations are varied. This is undesirable, as while smoothness is important, it is still desired that the robot chooses somewhat direct and energy efficient paths. Also, arbitrary choices of T_i may lead to uneven distribution

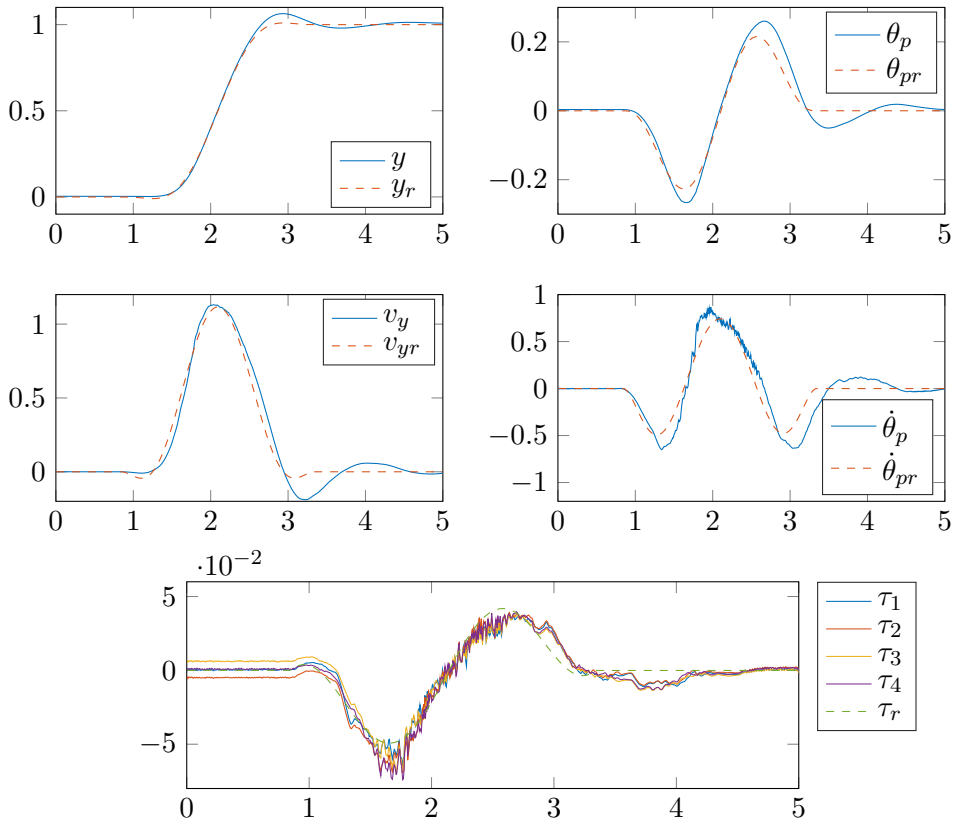


Figure 7.4: Experimental tracking of the trajectory in Figure 7.1. This shows more deviation from the planned trajectories than in simulation, though this is to be expected given the complex unmodelled dynamics acting on the system. Despite these errors, minimal position overshoot is observed, and the tracked trajectories strongly resemble that which was planned.

of cost along the trajectory, making some segments smoother than others and producing step changes in the perceived 'aggressiveness' of the overall trajectory.

A cost function [22] used in the quadrotor trajectory planning literature to optimise for T_i across n flat outputs is defined as

$$\min_{T_j} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n_w-1} \int_0^{T_j} \left(\frac{d^k p_{i,j}(t)}{dp_{i,j}(t)^k} \right)^2 dt + K_t T_j \right\} \tag{7.3.16}$$

where K_t is a scaling factor used to incentivize an appropriate reduction in segment durations. In practise this can be viewed as an analog to the 'aggressiveness'

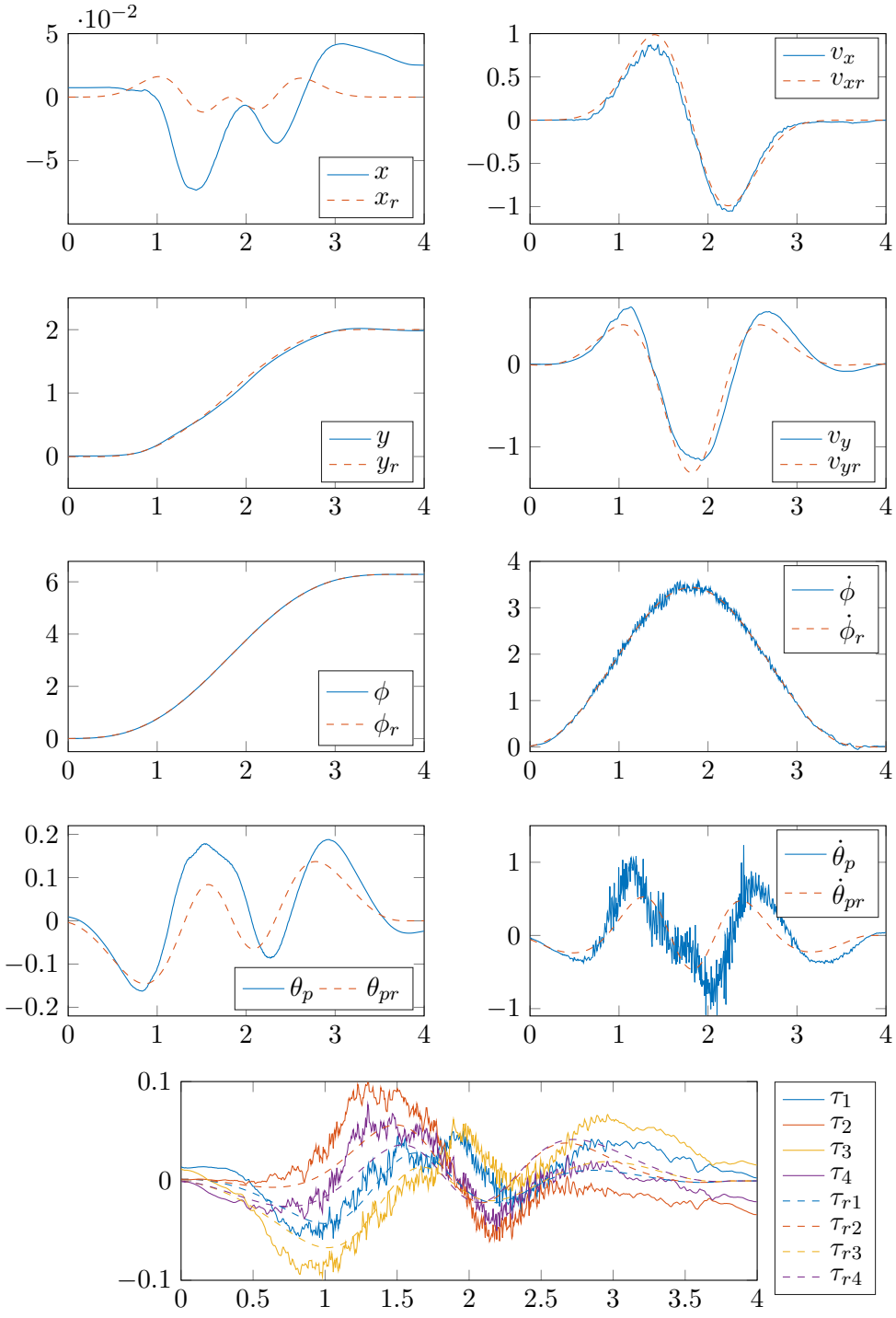


Figure 7.5: Experimental tracking of a pirouetting trajectory from $(x, y, \phi) = 0$ to $(x, y, \phi) = (0, 2, 2\pi)$. Wheel torque data is smoothed to improve readability.

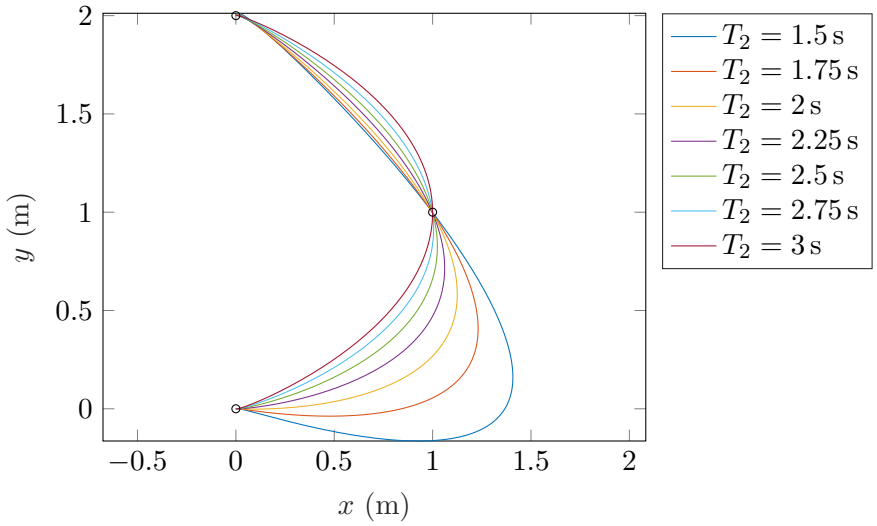
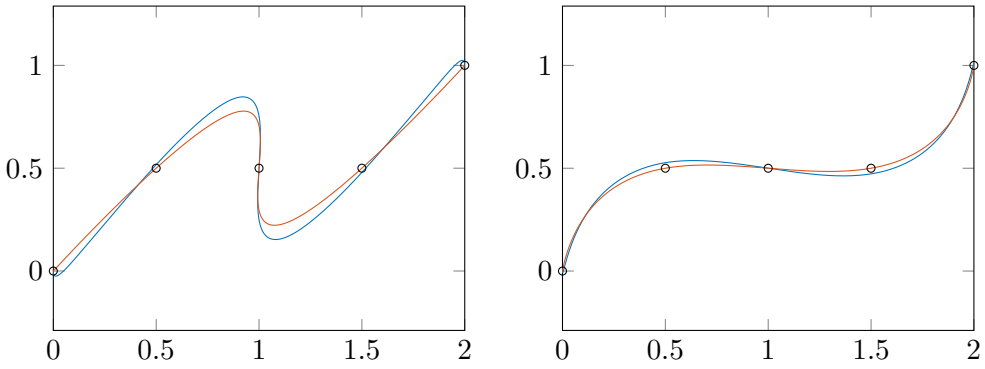


Figure 7.6: Comparison of choice of T_2 for a trajectory through the Cartesian waypoints $(0, 0)$, $(1, 1)$, and $(0, 2)$, with $T_1 = 3\,\mathrm{s}$.

of a trajectory, with a given choice of K_t yielding similarly 'aggressive' trajectories across any set of waypoints.

This cost function can be globally minimised by gradient descent, which when performed using the QR decomposition solution method in Section 7.3.3 can be performed with sub-second execution times for tens of waypoints using an Intel `i7-4720HQ` processor. If the QR decomposition method cannot be used, for example if any inequality constraints are to be included, iterative quadratic programming solvers make the process of numerically calculating gradients excessively expensive. This is compounded by the requirement to solve each QP to a high degree of optimality, as too lax of an optimality tolerance can introduce noise into the numerical gradient estimates. Figure 7.7 shows trajectories through a set of waypoints with both identical segment durations and with optimised durations, in which K_t is chosen to yield an equivalent total duration. This demonstrates how such an optimisation can yield more sensical and direct paths, whilst also providing a ten-fold reduction in total trajectory cost.



(a) $T = \{1, 1, 1, 1\}$ s (b) $T = \{1.58, 0.42, 0.42, 1.58\}$ s

Figure 7.7: Flat output (red) and state trajectories (blue) through five Cartesian waypoints (black markers) with constant segment durations of 1 s (a) and optimised durations (b), in which K_t is chosen to yield equal total duration. The initial QP has cost $J = 6507$, whilst the QP with optimised durations has cost $J = 629$, a ten-fold reduction.

7.4 Velocity Constrained Trajectory Generation

In optimising a simple rest-to-rest trajectory for minimum crackle, the resulting velocity trajectory forms a bell-shaped profile, in which the average velocity is much smaller than the peak, visible in Figure 7.1. In real-world scenarios robotic systems that operate in confined environments or around people must adhere to velocity constraints to bound the system's kinetic energy, meaning the peak of this bell curve must lie within constraint bounds. Using the cost function (7.3.16), K_t must be decreased to lengthen the duration of the trajectory until all peak velocities lie within constraint bounds. This potentially decreases the peak velocity of some segments much below the constraint in order to ensure constraint satisfaction of the segment with greatest violation. Alternatively, these velocity constraints can be represented by smooth barrier penalty functions, allowing the optimal selection of segment durations using the same cost function (7.3.16), so that all segments that were constraint violating have their peak velocities reduced to equal the constraint, at a cost of increased problem complexity. However, both of these methods yield far from time-optimal trajectories, as the resulting velocity profiles between waypoints will only equal the constraint at a single point at most, with the majority of the trajectory being slower than constraints allow.

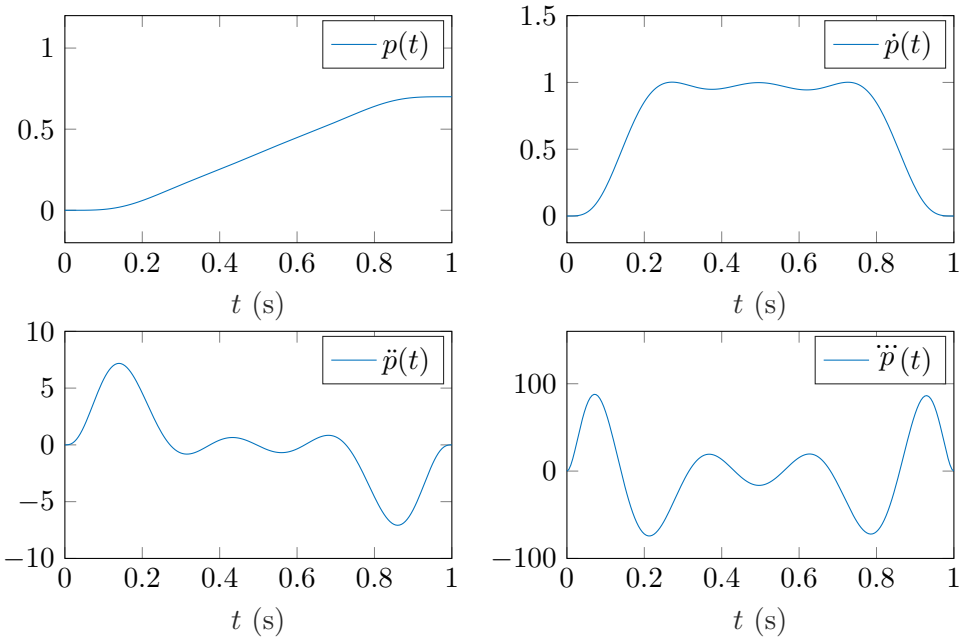


Figure 7.8: A high-degree velocity constrained polynomial exhibiting Runge's phenomena, in which approximating a constant velocity requires large variations in the polynomial's higher order derivatives.

An ideal velocity constrained rest-to-rest trajectory would reach $\overline{v}$ as quickly as the smoothness-to-time cost ratio allows, maintain $\overline{v}$ for an appropriate amount of time, before decelerating as quickly as the smoothness-to-time cost ratio allows. This can be approximated by using polynomials of increased degree, with the constant velocity part of the trajectory flattening with increasing $\deg(p(t))$. This method, however, will always exhibit some deviation from constant velocity during this middle section, and yield undesirable higher trajectory derivatives due to Runge's phenomenon, demonstrated in Figure 7.8. Furthermore, any increase in $\deg(p(t))$ requires larger monomial powers, quickly leading to poor numerical conditioning of the resulting optimisation when $t \not\approx 1$.

7.4.1 Splitting Polynomial Segments

This desired trajectory shape can instead be achieved by splitting all polynomial segments into three separate polynomial subsegments, whilst enforcing the same continuity of derivatives between subsegments, and without any intermediary po-

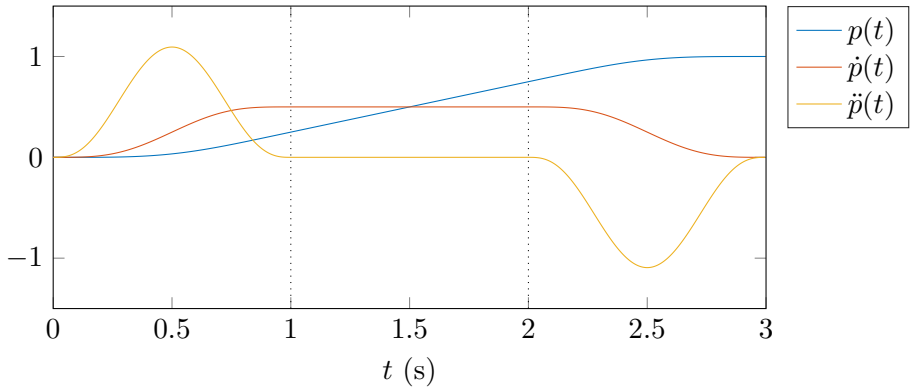


Figure 7.9: A point-to-point trajectory split into three subsegments, with a linear polynomial describing the middle subsegment. Dotted vertical lines represent subsegment boundaries. This demonstrates how this description of point-to-point trajectories in the presence of velocity constraints can be better parametrised by three concatenated polynomial subsegments.

sition constraints. This is exampled in Figure 7.9. Whilst for distant waypoints the middle of these three subsegments should be optimised to represent a constant velocity, and could therefore be represented by a linear polynomial, such a parametrisation will yield less smooth solutions for closely spaced waypoints where the velocity constraint is not reached. The degrees of the three new subsegments are therefore chosen to be equal. This parametrisation is demonstrated in Figure 7.9 for a 1 m point-to-point trajectory with equal segment durations of 1 s.

For state-to-rest trajectories through $n_w - 2$ intermediary position waypoints each subsegment requires an average subsegment polynomial degree of

$$\deg(p) \geq \frac{2(n_d+1) + 2(n_d+1) + (3(n_d+1)+1)(n_w-2)}{3(n_w-1)} - 1 \tag{7.4.1}$$

$$\geq n_d + \frac{n_d}{3(n_w-1)} + \frac{1}{3} \tag{7.4.2}$$

with limits

$$\deg(p) \geq \begin{cases} \frac{4(n_d+1)}{3} - 1 & \text{for} \quad n_w = 2 \\ n_d + 1 & \text{for} \quad n_w \to \infty \end{cases} \tag{7.4.3}$$

or in a receding horizon problem requires

$$\deg(p) \geq \frac{(n_d+1)+1+2(n_d+1)+(3(n_d+1)+1)(n_w-2)}{3(n_w-1)} - 1 \tag{7.4.4}$$

$$\geq n_d + \frac{1}{3} \; \forall \; n_w \geq 2 \tag{7.4.5}$$

7.4.2 Velocity Constraint Approximation

As discussed in Chapter 6, whilst exact constraint satisfaction is achieved by enforcing $\dot{x}^2 + \dot{y}^2 \leq \overline{v}^2$ this forms a quadratic constraint, greatly increasing the complexity of any optimisation. This can be avoided by approximation with a convex polytope [124], defined by the linear inequalities

$$\left(\dot{y} - \overline{v}\sin\left(\frac{2\pi m}{M}\right)\right)\left(\cos\left(\frac{2\pi(m+1)}{M}\right) - \cos\left(\frac{2\pi m}{M}\right)\right) - \left(\dot{x} - \overline{v}\cos\left(\frac{2\pi m}{M}\right)\right)\left(\sin\left(\frac{2\pi(m+1)}{M}\right) - \sin\left(\frac{2\pi m}{M}\right)\right) \geq 0 \tag{7.4.6}$$

where $M \in [3 \; .. \; \infty]$, $m \in [1 \; .. \; M]$. Choices of $M = [4, 8, 16]$ are shown in Figure 7.10. Whilst this linear formulation yields some simplification, any choice of M other than $M = 4$ prevents the decoupling of the S_1 and S_2 trajectories during polynomial optimisation, resulting in around a ten-fold increase in solver execution time over the simpler box constraint. $M = 4$ is therefore chosen to maintain feasible execution times, and $\overline{v}$ is assumed to be scaled to $\sqrt{2}\overline{v}$ as to access the full constrained velocity set. Whilst this allows a constraint violation along the diagonals of the global coordinates, this violation remains bounded to a worst case of 41%. More computationally efficient methods for minimising these violations are introduced later.

7.4.3 Velocity Constraint Enforcement using Quadratic Programming

Velocity constraints can be incorporated into the QP in (7.3.9) as linear inequalities in the form $Ax \leq b$. This requires a finite discrete set of $c = [1 \; .. \; n]$ values of t to be chosen at which the constraint is to be enforced as $\pm\dot{p}(t_c) \leq \overline{v}$. One obvious choice is to uniformly sample $t_c \in [0, T_i]$ for every subsegment, a method known as gridding. A single QP solution is then numerically found, albeit with

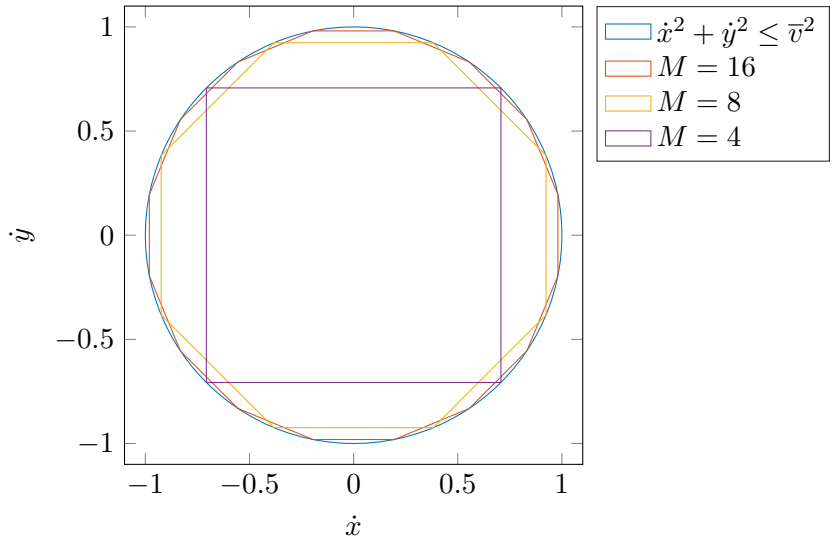


Figure 7.10: Polytope approximations of the quadratic velocity constraint $\dot{x}^2 + \dot{y}^2 \leq \overline{v}^2$ with $\overline{v} = 1$.

a large number of inequalities, and as long as a feasible solution is found it is assumed that the continuous constraints are satisfied, though violation could still occur between the grid points. In practice this method can generate oscillatory trajectories, in which the resulting velocity profile repeatedly satisfies constraints where they are sampled but then violates them between.

Alternatively, velocity constraints can be enforced in a recursive manner, providing a guarantee of constraint satisfaction. This method is described in Algorithm 1. On the first iteration the QP's decision variable invariant H, A_{eq}, and b_{eq} matrices and vectors are calculated, and the A and b matrix and vector in the inequalities $Ax \leq b$ are initialised as empty. The QP is then solved, providing the optimally smooth connecting path between the waypoints in the absence of velocity constraints. Numerical root finding is then used to find the roots t_r of the second derivative of every subsegment within $t_r \in [0, T_i]$, i.e. the values of t_r within $t_r \in [0, T_i]$ where $\ddot{p}(t_r) = 0$. These represent local velocity maxima and minima along the trajectory, and so if $\dot{p}(t_r)$ at these points is velocity constraint violating t_r represents the instant of peak violation. Appropriately signed inequalities can then be inserted at these points by addition of rows to A and b, and the QP is resolved. This process repeats until all local velocity maxima and minima lie within constraint bounds, or fails if an iteration renders the QP

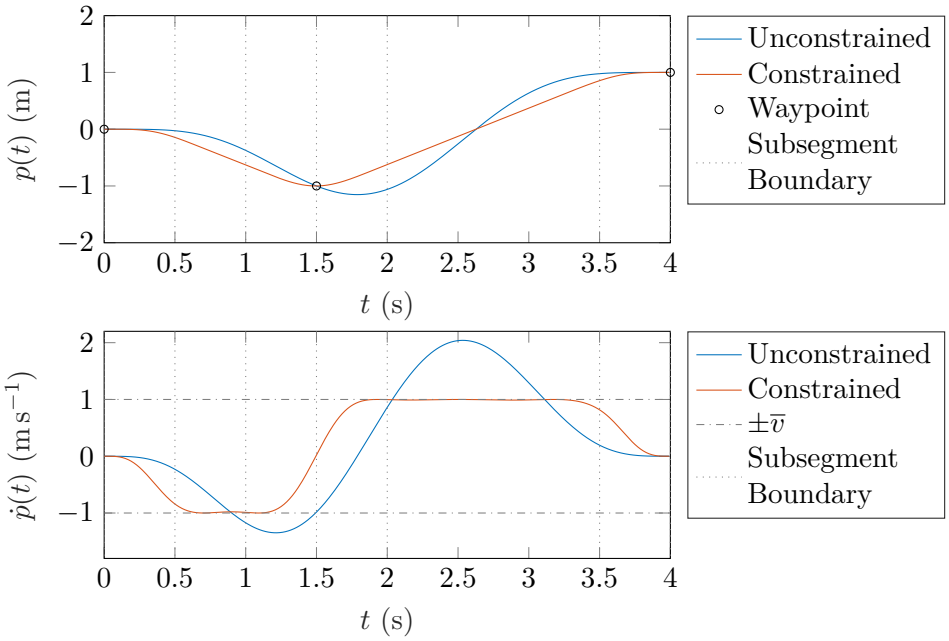


Figure 7.11: A comparison of the original unconstrained polynomial QP solution with the proposed velocity constrained planner. Subsegment boundaries are denoted by vertical dotted lines, the timings of which are manually specified.

infeasible.

An example of this trajectory planner is shown in Figure 7.11, along with the original QP solution without enforcement of velocity constraints. Subsegment durations are selected by a method introduced later in this chapter.

7.4.4 Velocity Constraint Enforcement using Sum-of-Squares Programming

While in the recursively constrained QP method each individual QP can be solved very quickly, often up to around ten iterations can be required to fully constrain the trajectory. When combined with the overhead of performing numerical root finding on every subsegment at each iteration, this can result in a significantly longer overall execution time than the original unconstrained QP. Here a novel non-recursive method is presented, utilizing sum-of-squares (SOS) programming to directly enforce the velocity constraints in continuous time, rather than a discrete enforcement at a recursively grown list of sampling points. This is achieved

Algorithm 1: Recursively velocity constrained QP algorithm

Result: Concatenated polynomial coefficients x
Define H, A_{eq}, b_{eq};
$A_0 = []$, $b_0 = []$;
for *k=1:maxIter* **do**
  Solve $\min_x x^T H x \quad s.t.\ A_{eq}x = b_{eq},\ Ax \leq b$;
  if *QP is infeasible* **then**
    break;
  end
  for *j=1:n_w-1* **do**
    Use numerical root finding to find all t_i, $i \in [1 \,..\, n_r]$ where $\ddot{p}_j(t_i) = 0$;
    for *i=1:n_r* **do**
      if $t_i \in [0, T_j]$ **then**
        if $|\dot{p}_j(t_i)| > \overline{v}$ **then**
          Add row to A_{new} and b_{new} to enforce $|\dot{p}_j(t_i)| \leq \overline{v}$;
        end
      end
    end
  end
  $A_k = \begin{bmatrix} A_{k-1}^T & A_{new}^T \end{bmatrix}^T$, $b_k = \begin{bmatrix} b_{k-1}^T & b_{new}^T \end{bmatrix}^T$;
  if $A_k = A_{k-1}$ **then**
    A_k is invariant, therefore constraints are satisfied;
    return x
  end
end
return *Failed to iteratively constrain trajectory, problem is infeasible.*

whilst maintaining both the same guarantee of optimality and the same solution space as the recursively constrained approach.

A polynomial $p(t)$ of degree $2d$ that is a sum-of-squares polynomial can be written in the Gram matrix form

$$p(t) = z(t)^T H z(t), \quad H \succeq 0 \tag{7.4.7}$$

where $z(t)$ is a column vector containing the monomials of $p(t)$ up to degree d, and H is positive semidefinite. If a univariate polynomial $p(t)$ can be represented in this form, then $p(t) \geq 0 \ \forall\ t \in \mathbb{R}$. All univariate non-negative polynomials are

sum-of-squares polynomials, meaning this basis retains the same set of solutions as that achieved by recursively enforcing non-negativity as in the previous section [132].

Furthermore, the constraint $p(t) \geq 0$ can be enforced on just the interval $t \in [a, b]$ if $p(t)$ can be decomposed in the form

$$p(t) = \begin{cases} s(t) + (t-a)(b-t)q(t), & \textit{if } \deg(p(t)) \textit{ is even} \\ (t-a)s(t) + (b-t)q(t), & \textit{if } \deg(p(t)) \textit{ is odd} \end{cases} \tag{7.4.8}$$

where $s(t)$ and $q(t)$ are SOS. For even $\deg(p)$, $\deg(p) = 2d$, $\deg(s) \leq 2d$, and $\deg(q) \leq 2d-2$. For odd $\deg(p)$, $\deg(p) = 2d+1$, $\deg(s) \leq 2d$, $\deg(q) \leq 2d$ [133]. Similarly, all univariate polynomials that are non-negative on an interval can be decomposed in this way, so again this basis does not yield a reduced polynomial set.

It is therefore possible to enforce the constraint $-\overline{p} \leq p(t) \leq \overline{p} \; \forall \; t \in [a, b]$ by considering the necessary equivalence

$$\overline{p} + p(t) \equiv s_1(t) + (t-a)(b-t)q_1(t) \tag{7.4.9}$$

$$\overline{p} - p(t) \equiv s_2(t) + (t-a)(b-t)q_2(t) \tag{7.4.10}$$

which can be achieved by enforcement of the four SOS constraints

$$\text{SOS}\left(\overline{p} + p(t) - (t-a)(b-t)q_1(t)\right) \tag{7.4.11}$$

$$\text{SOS}\left(\overline{p} - p(t) - (t-a)(b-t)q_2(t)\right) \tag{7.4.12}$$

$$\text{SOS}\left(q_1(t)\right) \tag{7.4.13}$$

$$\text{SOS}\left(q_2(t)\right) \tag{7.4.14}$$

for the even $\deg(p(t))$ case, and

$$\text{SOS}\left(\frac{\overline{p} + p(t) - (b-t)q_1(t)}{(t-a)}\right) \tag{7.4.15}$$

$$\text{SOS}\left(\frac{\overline{p} - p(t) - (b-t)q_2(t)}{(t-a)}\right) \tag{7.4.16}$$

$$\text{SOS}\left(q_1(t)\right) \tag{7.4.17}$$

$$\text{SOS}\left(q_2(t)\right) \tag{7.4.18}$$

for the odd $\deg(p(t))$ case.

Applying these constraints to the first derivative of every subsegment allows the enforcement of the velocity constraint $|\dot{p}_i(t)| \leq \overline{v} \; \forall \; t \in [0, T_i]$.

Writing the convex QP (7.3.9) in the epigraph form

$$\min_{x,t} \; t \quad s.t. \quad x^T H x \leq t, \quad A_{eq} x = b_{eq}, \quad Ax \leq b \tag{7.4.19}$$

and using the Schur complement test for positive semidefiniteness, with positive definite H decomposed as $H = P^T P$ by eigendecomposition, allows (7.4.19) to be written in the linear matrix inequality form

$$\begin{bmatrix} I_\rho & Px & 0_{\rho \times m} \\ x^T P^T & t & 0_{1 \times m} \\ 0_{m \times \rho} & 0_{m \times 1} & \text{diag}(b - Ax) \end{bmatrix} \succeq 0 \tag{7.4.20}$$

where $m = \text{rank}(A)$, $\rho = \text{rank}(P)$. This allows the definition of an equivalent semidefinite program (SDP)

$$\min_{x,t} \; t \quad s.t. \quad \begin{bmatrix} I_\rho & Px & 0_{\rho \times m} \\ x^T P^T & t & 0_{1 \times m} \\ 0_{m \times \rho} & 0_{m \times 1} & \text{diag}(b - Ax) \end{bmatrix} \succeq 0 \tag{7.4.21}$$

The SOS constraints (7.4.11)-(7.4.18) can then be enforced by additionally enforcing the positive semidefinite constraint $H \succeq 0$ in (7.4.7) for each SOS polynomial. The implementation of this is handled by YALMIP's SOS module [134, 135], and the resulting SDP is solved by MOSEK V8.1 [136] using default optimality tolerances.

The distributions of execution time over increasing waypoint number n_w for both the recursively constrained QP and SOS constrained SDP methods are given in Figure 7.12 over $n_w \in [2 \; .. \; 10]$. 1000 random sets of waypoints are evaluated for each value of n_w, generated by selecting $n_w - 1$ waypoints from a standard normal distribution, with the system initialised at the origin. Subsegment timings are chosen as the minimum time solution to a constrained trapezoidal profile, introduced in the next section. Both the QP and SDP solvers are provided by the MOSEK software package for fairness of comparison, and all solver tolerances are left at their defaults. The SOS constrained method shows an average 40%

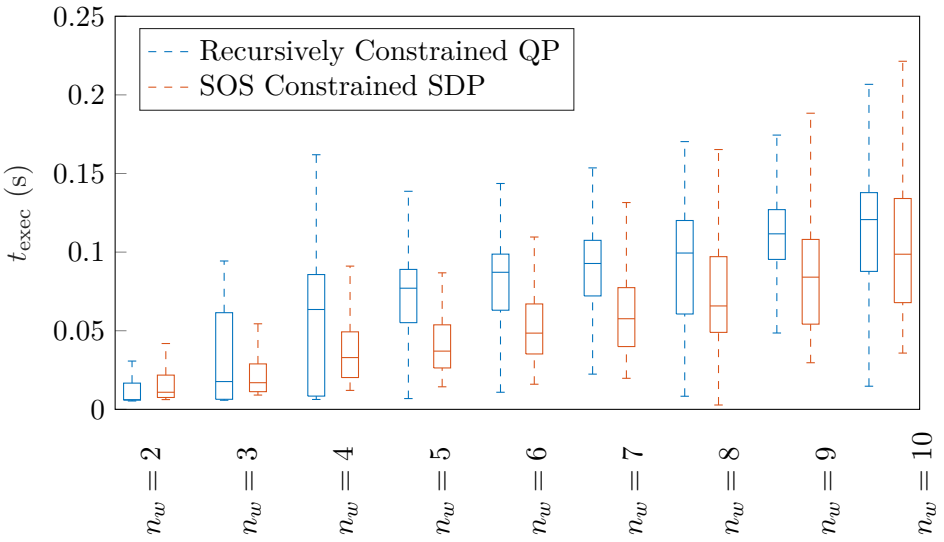


Figure 7.12: Monte Carlo analysis of the distribution of solve time over waypoint quantity for velocity constrained trajectories, solved using both the recursively constrained QP and SOS constrained SDP methods. Each value of n_w is evaluated using a 1000 point Monte Carlo simulation, where each set of waypoints are drawn from a standard normal distribution. Subsegment durations are selected as the minimum time solution to a constrained trapezoidal profile. This shows that the new SOS constrained SDP trajectory optimisation method yields a significant decrease in computation time over the recursively constrained QP method.

reduction in computation time over the recursively constrained QP method, with up to a 60% reduction observed around $n_w = 5$. Both methods prove sufficiently fast for use in online replanning, allowing for quick reaction to dynamic obstacles and disturbance.

7.4.5 Velocity Constrained Subsegment Duration Selection

As in Section 7.3.5 it is necessary to select all segment durations *a priori*, which now also involves the selection of subsegment durations, whilst maintaining total segment duration coherence between flat outputs. This is more challenging than in the unconstrained case, as now selecting too short of a segment duration could require an average segment velocity greater than $\overline{v}$, rendering the problem infeasible.

The globally optimal solution with a similar cost function as (7.3.16) can be found by gradient descent as in Section 7.3.5, with velocity constraint violation included using inequalities with a quadratic cost on violation. Using the recursive QP method in Section 7.4.3 this optimisation takes tens of seconds for trajectories through five random waypoints, and is therefore impossible to perform in real-time.

While it is impossible to optimise segment and subsegment durations for the full problem in real-time, it is possible to optimise a simplified problem sufficiently quickly for real-time implementation. Noting the similarity between the velocity constrained trajectories in Figure 7.11 and a piecewise constant acceleration trapezoidal profile, every middle subsegment of each segment is simplified to a linear polynomial, and all other subsegments are simplified to quadratic polynomials. This reduced basis is then optimised for minimum time whilst enforcing velocity and acceleration constraints. This simplified formulation also allows the exact enforcement of the quadratic velocity constraint $\dot{x}^2 + \dot{y}^2 \leq \overline{v}^2$ without a significant increase in problem complexity, which by lengthening the durations of segments during combined motion along x and y effectively acts to approximately enforce the full quadratic velocity constraint on any subsequent full-degree optimisation. This alleviates some of the potential constraint violation introduced in Section 7.4.2.

Defining subsegment polynomials as $p_{i,j}(t)$, waypoints as $w_{i,j}$, and subsegment durations as $T_{i,j}$, where $i \in [1 \,..\, 3]$ specifies the flat output index, and $j \in [1 \,..\, 3(n_w - 1)]$ specifies the subsegment index, this nonlinear program can be defined as

$$\min_{x,T} \sum_{i=1}^{3} \sum_{j=1}^{3(n_w-1)} T_{i,j} \tag{7.4.22}$$

$$s.t. \quad \sum_{j=k}^{k+2} T_{i,j} = \sum_{j=k}^{k+2} T_{i+1,j} \;\forall\; i \in [1 \,..\, 2],\; k \in [1, 4, .. \, 1 + 3(n_w - 1)] \tag{7.4.23}$$

$$p_{i,j}^{(n)}(T_{i,j}) = p_{i,j+1}^{(n)}(0) \;\forall\; i \in [1 \,..\, 3],\; j \in [1 \,..\, 3(n_w - 1)],\; n \in [0 \,..\, 1] \tag{7.4.24}$$

$$p_{i,1}^{(n)}(0) = w_{i,1}^{(n)} \ \forall \ i \in [1 \,..\, 3], \ n \in [0 \,..\, 1] \tag{7.4.25}$$

$$p_{i,3j}(T_{i,3j}) = w_{i,j+1} \ \forall \ i \in [1 \,..\, 3], \ j \in [1 \,..\, n_w - 1] \tag{7.4.26}$$

$$(\dot{p}_{1,j}(0))^2 + (\dot{p}_{2,j}(0))^2 \leq \overline{v}^2 \ \forall \ i \in [1 \,..\, 3], \ j \in [2, 5, .. \, 3(n_w - 1) - 1] \tag{7.4.27}$$

$$|\dot{p}_{3,j}(0)| \leq \overline{\dot{\phi}} \ \forall \ j \in [2, 5, .. \, 3(n_w - 1) - 1] \tag{7.4.28}$$

$$|\ddot{p}_{i,j}(0)| \leq \overline{\dot{v}} \ \forall \ i \in [1 \,..\, 2], \ j \in [1, 3, 4, 6, 7 \,..\, 3(n_w - 1)] \tag{7.4.29}$$

$$|\ddot{p}_{3,j}(0)| \leq \overline{\ddot{\phi}} \ \forall \ j \in [1, 3, 4, 6, 7 \,..\, 3(n_w - 1)] \tag{7.4.30}$$

This is made up of the cost function (7.4.22), constraints required to ensure segment duration coherence across flat outputs (7.4.23), inter-subsegment continuity constraints (7.4.24), initial state and waypoint position constraints (7.4.25)-(7.4.26), and velocity and acceleration constraints (7.4.27)-(7.4.30). By enforcing exact velocity constraints the violations along the diagonals of the coordinate system discussed in Section 7.4.2 can be significantly reduced.

This minimisation problem forms a nonlinear program (NLP), solved in this thesis using MATLAB's `fmincon`. NLP solve times for increasing numbers of random waypoints are shown in Figure 7.13. This shows real-time execution feasibility, again allowing for fast online replanning for up to $n_w \approx 7$. Furthermore, assuming relatively sparse waypoints it is reasonable to assume that only the first handful of waypoints will have a significant effect on the timings of the first segment. This optimisation could therefore be approached in a receding horizon fashion using a horizon of only three or four waypoints, which would allow for a linear scaling of execution time as $n_w \to \infty$.

A minimum time trapezoidal profile trajectory through nine random waypoints is shown in Figure 7.14, along with the corresponding constrained and optimally smooth full-degree flat output trajectories. The trapezoidal trajectory in S_2 is visibly a bang-bang trajectory for the majority of the path, as would be expected for a minimum-time solution. The only part of S_2 that deviates from bang-bang behaviour is that from the first to second waypoint, where constraints on the trajectory in S_1 prevent any further reduction in segment duration. This demonstrates the coupling between flat outputs that prevents the minimum-time problem from being addressed as three independent subproblems, frustrating the development of a simpler heuristic-based subsegment duration allocation method.

A comparison of trapezoidal and globally optimised subsegment timings for

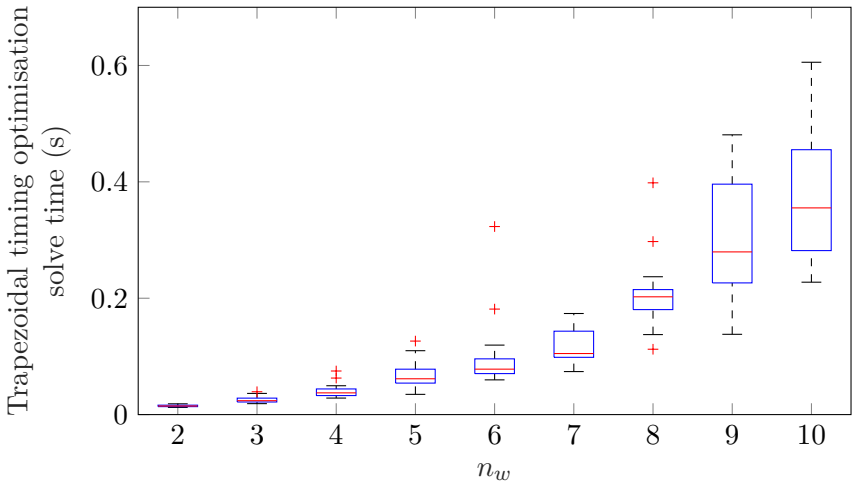


Figure 7.13: Distribution of trapezoidal timing optimisation execution time for increasing numbers of waypoints in three synchronised flat outputs, implemented on an Intel i7-4720HQ processor.

five waypoints is shown in Figure 7.15, showing nearly identical solutions. This two-stage duration selection scheme is found to yield trajectories that are close to the globally optimal solution for smoothness over time, whilst achieving a 100 to 1000 fold reduction in execution time.

7.4.6 Experimental Velocity Constrained Trajectory Planning and Tracking

As the recursively constrained QP and SOS constrained SDP optimization approaches yield identical outputs, and the trapezoidal timing allocation method yields very similar segment timings to the globally optimal approach, there is little to be gained by an experimental comparison of these methods. This final experimental section is therefore intended to serve as a *tour de force*, demonstrating both the performance and manoeuvrability achievable by the CMD using these control and trajectory planning methods.

Figure 7.16 shows a rest-to-rest trajectory passing through a gap narrower than the prototype's width. This serves as a minimal example of the manoeuvrability achievable by the CMD in confined spaces, and how smooth transitions can be achieved between forward and lateral motion. State trajectories for this experi-

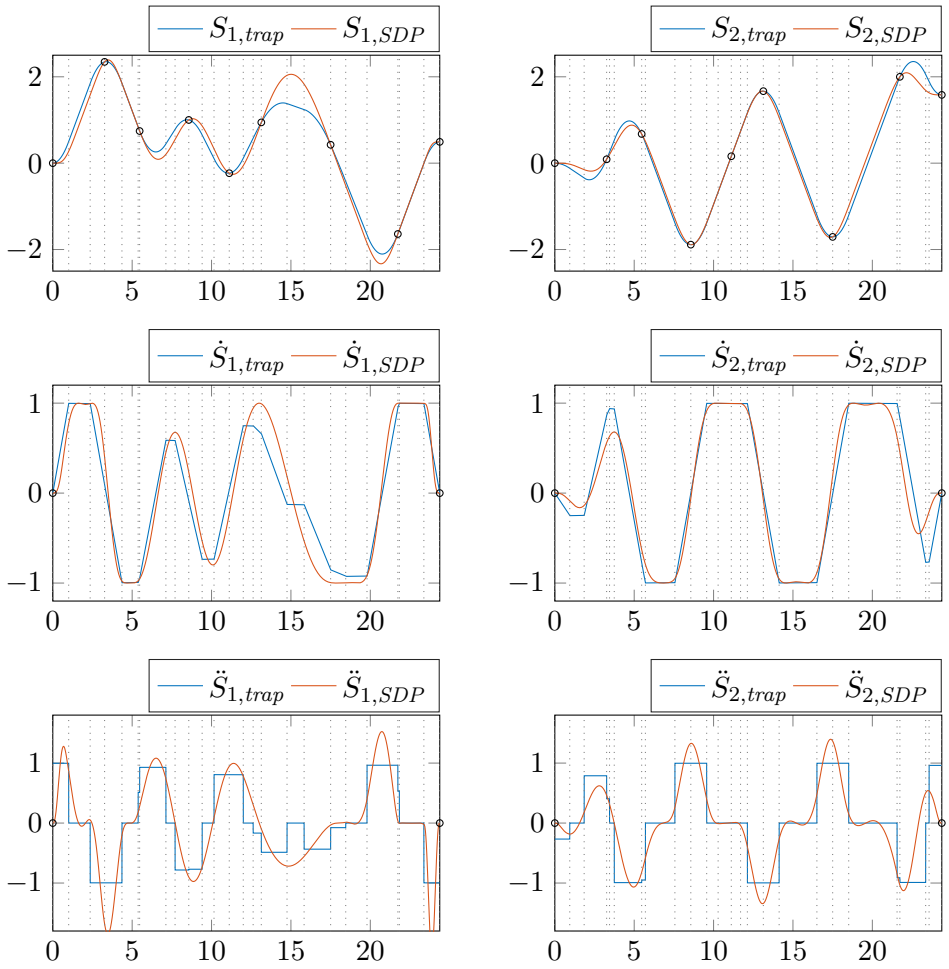


Figure 7.14: Minimum time trapezoidal (blue) and full degree optimally smooth (red) trajectories through nine waypoints (black markers) with constraints $\overline{v} = 1$, $\overline{a} = 1$. The trajectory in S_3 is left at rest and not shown. Vertical dashed lines denote subsegment boundaries.

ment are shown in Figure 7.18, demonstrating good tracking, further validating the accuracy of the differentially flat model.

Figure 7.17 shows a rest-to-rest pirouetting trajectory from the origin to $(x, y, \phi) = (0, 2, 4\pi)$, in which two blue LEDs are used to capture the tracked trajectory. This again demonstrates smooth transitions between different body relative directions of motion, and shows zero overshoot. Corresponding state trajectories are shown in Figure 7.19. While a small tracking error is observed, this error

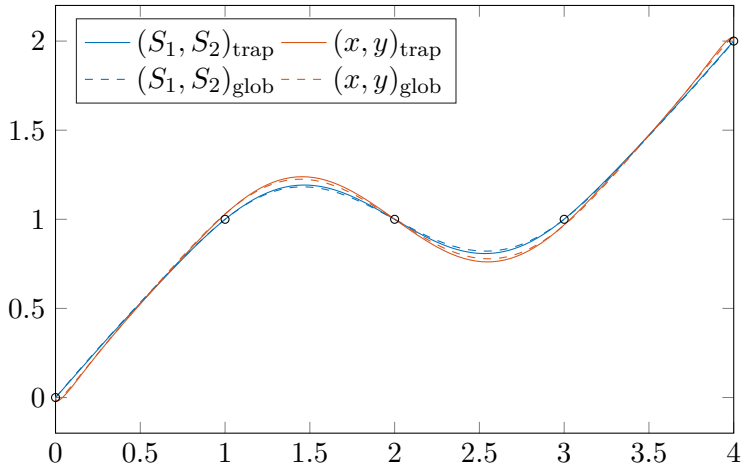


Figure 7.15: Optimally smooth flat output (blue) and state (red) trajectories for trapezoidal optimised polynomial durations (solid) and gradient descent optimised polynomial durations (dashed) through five waypoints (black markers), with equal total duration between the two schemes.

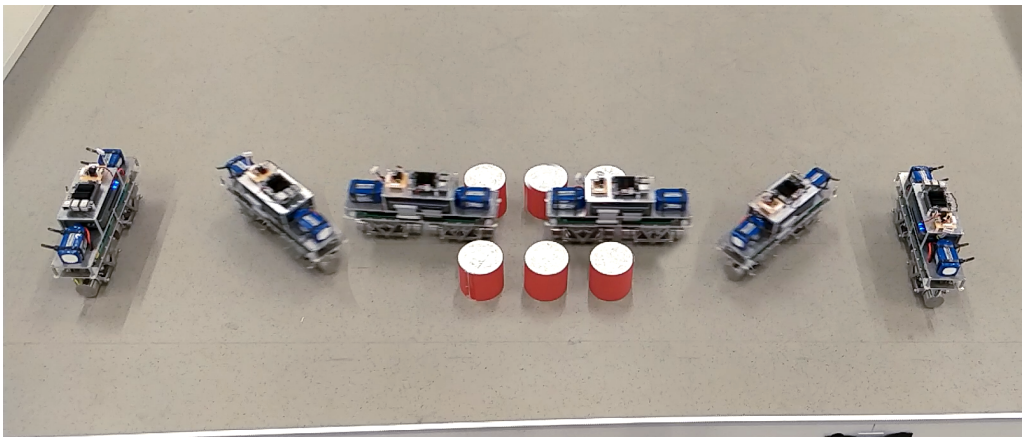


Figure 7.16: Navigation of a gap narrower than the robot's width, demonstrating one of the advantages of the Collinear Mecanum Drive. This trajectory passes through four waypoints, spaced over 2 m and progressing right across the figure in a total of 3 s, in which a 90° ϕ angle is enforced at the middle two waypoints.

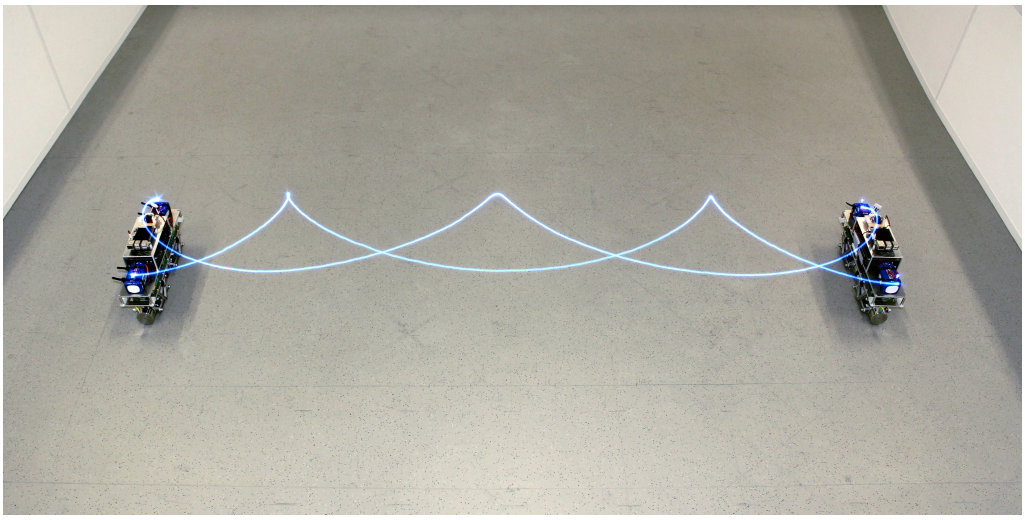


Figure 7.17: A long exposure image of a trajectory performing a 2 m translation and a 720° rotation in ϕ, with two blue LEDs used to capture the path of the robot, moving left to right across the figure. This is intended to demonstrate the smoothness of the resulting trajectories achievable with this approach, and the accuracy with which they can be tracked.

remains bounded over the duration of the trajectory, and is indistinguishable in the long exposure image in Figure 7.17.

Finally, as a demonstration of the complete freedom of movement attainable by the CMD, Figure 7.20 shows a long exposure image capturing the prototype tracking a circular trajectory through the four corners of a 1.5 m square, whilst also performing a rotation in ϕ of 6π rad. This also serves as a good demonstration of the navigation accuracy achievable using only dead reckoning, showing a drift of only around a 4 cm during this experiment. Reference and tracked state trajectories for a trajectory similar to this are shown in Figure 7.21, in which a reduced rotation of 4π rad is performed to improve the clarity of the figure.

7.5 Conclusion

This chapter has shown that through the selection of suitable fictitious flat outputs and model simplification the dynamic model of a CMD can be differentially flattened, allowing the derivation of dynamically feasible state trajectories from any sufficiently smooth trajectories in the flat outputs. The accuracy of this

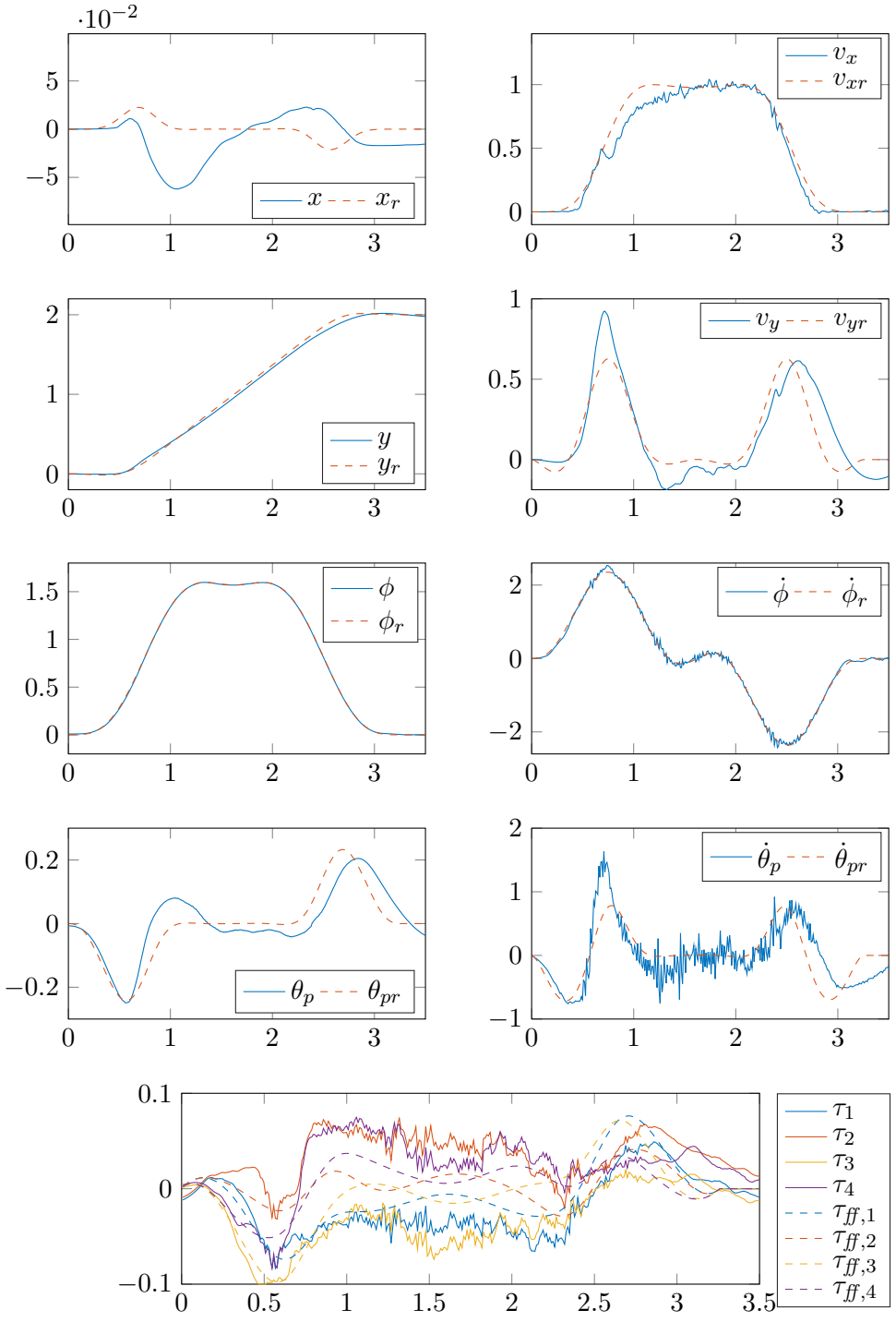


Figure 7.18: State trajectories for the gap navigation experiment shown in Figure 7.16, in which a 2 m translation is performed through a gap narrower than the CMD's width, achieved by rotating the CMD to translate along its wheel axle.

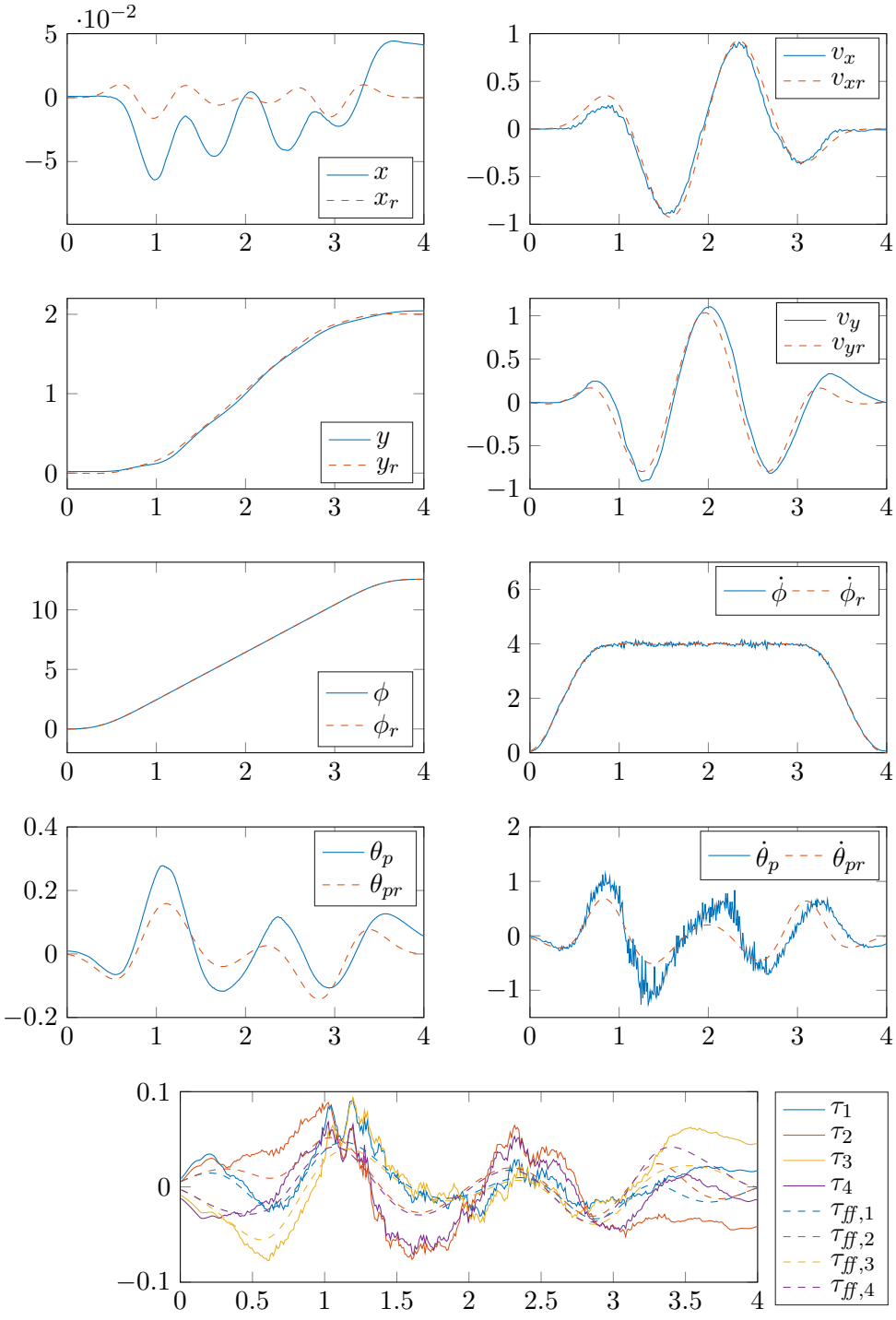


Figure 7.19: State trajectories for the pirouetting trajectory experiment shown in Figure 7.17, in which the CMD translates from the origin to $(x, y, \phi) = (0, 2, 4\pi)$ in 4 s.

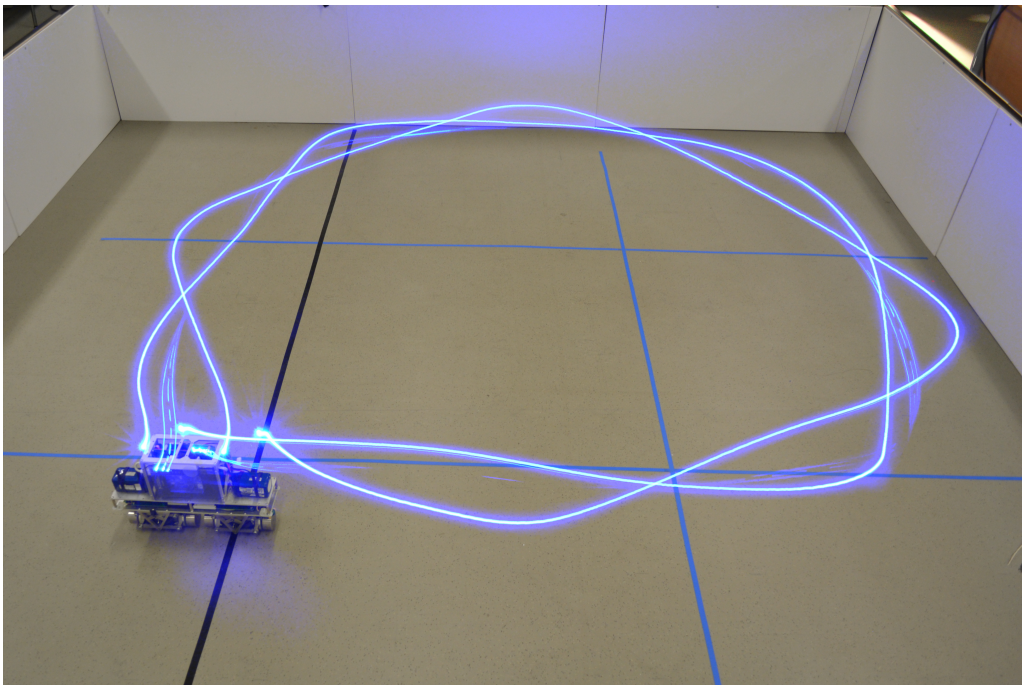


Figure 7.20: A long exposure image of a circular trajectory passing through the four corners of a 1.5 m square whilst performing a rotation in ϕ of 6π rad. This is intended to demonstrate the complete freedom of motion achievable using the CMD.

model has then been demonstrated both in simulation and on the experimental prototype, showing it generates nearly exactly dynamically feasible trajectories that are closely tracked in real-world testing.

Two novel velocity constrained trajectory planners have then been demonstrated, allowing for the online derivation of optimally smooth and constrained trajectories through large numbers of waypoints. Finally, a method for the selection of the timing properties of these waypoints as been demonstrated, which is again sufficiently computationally efficient for real-time implementation. Together, these methods allow for a CMD to smoothly navigate complex environments in a close to time-optimal fashion.

Future work would explore the autonomous selection of minimal sets of waypoints required to describe paths between locations in an occupancy map, allowing the navigation of environments with obstacles. This problem has been well addressed in the quadrotor trajectory planning literature [22, 137, 138]. In this

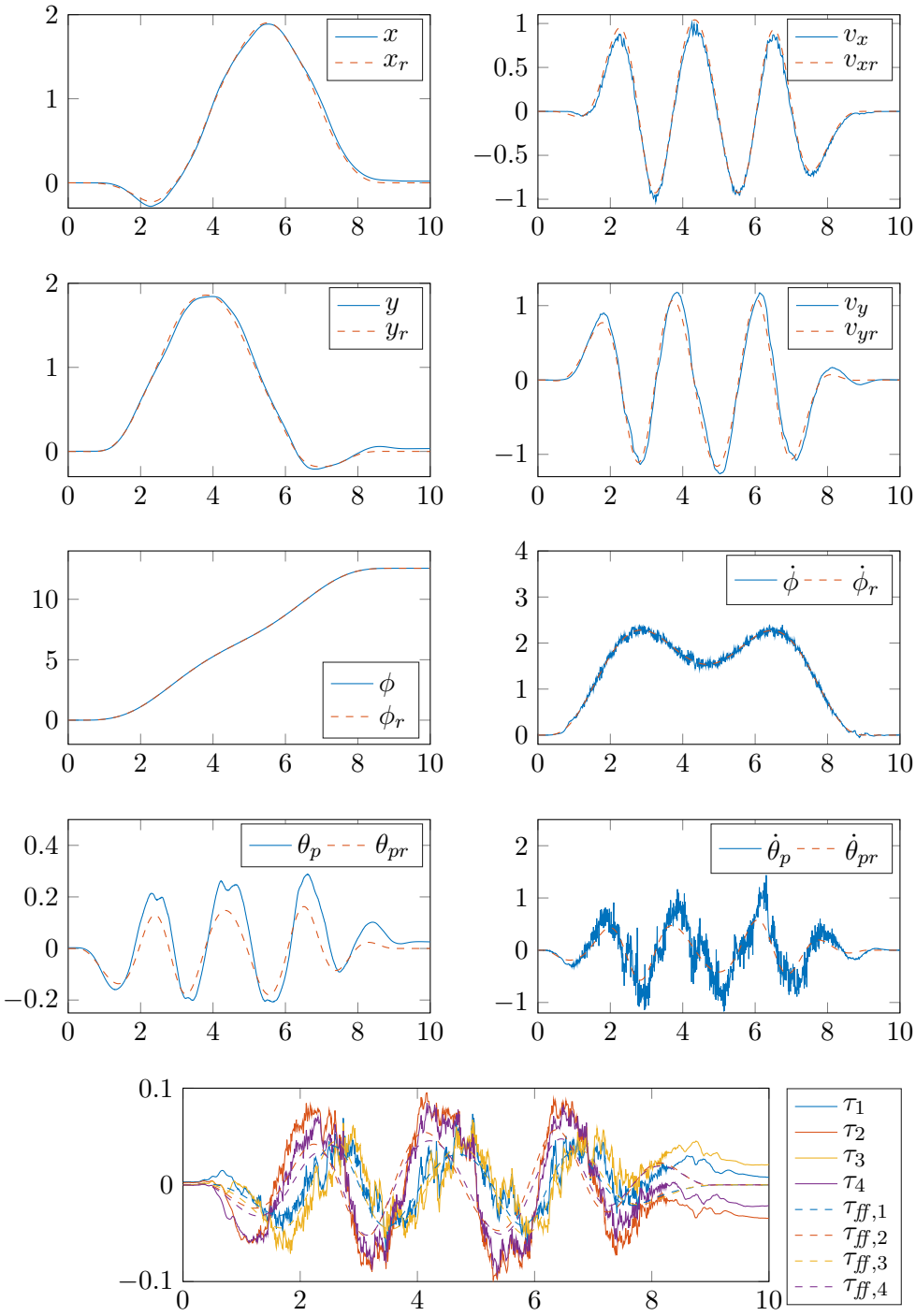


Figure 7.21: State trajectories for the same experiment as shown in Figure 7.20, but whilst only performing a rotation of 4π rad in ϕ in order to improve clarity.

application these methods could be extended to incorporate knowledge of the non-spherical shape of this robot. This would allow for the appropriate choice of heading trajectories in order to navigate smaller gaps than could be achieved when using the standard spherical robot model, similar to that which has been achieved for statically stable omnidirectional robots [40].

Chapter 8

Conclusion

This concluding chapter serves to summarise and present the work of this thesis as a whole, performs a qualitative comparison of the CMD control and planning techniques developed within, and sets out proposals for future work building on that detailed in this thesis.

This thesis has presented a systematic and comprehensive analysis of the Collinear Mecanum Drive, which should come to represent the *de facto* standard reference for the modelling and analysis of this type of system.

This analysis includes the kinematic and dynamics modelling of the CMD with first order friction models, a proof of the CMD's controllability, determination of size of the maximum feedback linearisable subsystem, and an analysis of the system's underactuation and nonholonomy properties. Following this, three distinct approaches to the control of this system have been developed, implemented in both simulation and on an experimental prototype, and analysed.

First, partial feedback linearisation and backstepping techniques are used to develop local body frame velocity, inertial frame velocity, and inertial frame position controllers, all with stability guarantees for the full nonlinear model. These enforce asymptotic body acceleration constraints, yielding smooth motion and an approximate wheel torque constraint. The two velocity controllers also enforce an asymptotic lean angle constraint, and the position controller also enforces asymptotic velocity constraints.

Second, an input and output constrained linear model predictive controller is developed, allowing for the systematic satisfaction of constraints whilst maintaining a stability guarantee for the linearised model. Despite the difficulties sur-

rounding the implementation of online optimisation based controllers on plants with fast unstable dynamics such as the CMD, the controller developed here showed good experimental performance. This controller is most useful in scenarios where aggressive control is required while suffering from poor wheel traction, necessitating the rationing of available wheel grip, and tolerating prolonged wheel actuator saturation without detrimentally affecting system stability.

Finally, a method for the tracking of smooth polynomials has been developed, using the concept of differential flatness. A novel velocity constrained polynomial trajectory planner is then demonstrated, allowing the planning and tracking of near time-optimal trajectories through multiple position waypoints. Provided a method of selecting suitable waypoints in a map, this method allows the optimal navigation of an environment with obstacles.

8.1 A Comparison of Controllers

In this section a qualitative comparison of the properties of the backstepping, MPC, and differential flatness based controllers developed in this thesis is performed. While a quantitative comparison would usually be preferred, the three controllers in this thesis are all designed for different end uses, and so a quantitative would not allow for a meaningful comparison. Instead, a qualitative approach is used to compare the overall properties of the controllers, and their suitability for different applications is considered.

8.1.1 Aggressiveness of Control

Defining the 'aggressiveness' of a controller as how near it is able to come to generating minimum-time trajectories for given input and state constraints, and to what degree the controller is able to reject disturbances, an approximate comparison of the maximum aggressiveness achievable by each controller can be made.

The backstepping controllers developed in Chapter 5 are inherently conservative due to the need to enforce the assumption of convergence of the inner linear $\theta_p \rightarrow \theta_{pr}$ controller. This acts to limit $\dot{\theta}_p$, preventing fast trajectory tracking and disturbance rejection in the v_y state and its integral. However, high bandwidth disturbance rejection is maintained for the feedback linearised subsystems,

though gains must be chosen as to avoid excessive actuator saturation and subsequent instability.

Chapter 6's model predictive controller is the most aggressive developed in this thesis, capable of optimally saturating the wheel torque setpoints for prolonged durations without any negative effect on system stability, along with exactly enforcing velocity and lean angle constraints. Decreasing R in the underlying cost function effectively tends this controller towards the generation of minimum-time bang-bang trajectories for the linearised plant.

The differential flatness based polynomial planner of Chapter 7 does not incorporate input constraint enforcement, and so segment durations must be conservative as to avoid actuator saturation. The tracking TVLQR controller is also unable to enforce wheel torque constraints, so again gains must be conservatively chosen as to maintain stability for a given degree of trajectory deviation and input constraint.

8.1.2 Computational Requirements

The computational complexity of the developed controllers varies considerably between methods, with a large impact on onboard processing requirements and subsequent power consumption.

Chapter 5's backstepping controllers possess the least demanding computational requirements of this thesis, allowing for implementation on low-cost microcontrollers. Conversely, Chapter 6's model predictive controller brings the greatest computational requirements, as every control iteration requires the numerical solution of a constrained quadratic program whilst introducing as little execution delay as possible, necessitating the use of a high-performance processor. However, the CMD studied in this thesis represents a difficult CMD to which to apply this control approach, as this particular system's low inertia and center of mass yield very fast dynamics and subsequent sensitivity to control computation delay. A taller, human-sized system would possess significantly slower dynamics, and would therefore be much less sensitive to this delay, potentially allowing for implementation on low-power hardware.

The differential flatness based trajectory planning approach of Chapter 7 imposes different timing requirements for the inner trajectory tracking TVLQR controller and for the outer trajectory planner. As the TVLQR controller is re-

sponsible for closed-loop stability and high bandwidth disturbance rejection, real time operation is required, which for this simple control formulation can be trivially performed by a low-power microcontroller. The outer trajectory planner has much greater computational requirements, but only needs to plan new trajectories during large disturbances or a change in objective, potentially allowing this planning to be performed in parallel with other high-level tasks such as SLAM and communication.

8.1.3 Actuation Power Requirements

To allow a comparison of the energy requirements of each method total actuation power consumption can be approximated as purely that due to resistive losses in the motor windings as $E = \int I(t)^2 \, dt$, which is in turn proportional to motor torque, so $E \propto \int \|\tau\|_2^2 \, dt$. The most power efficient controller would therefore be that which uses the least energy to perform some point-to-point manoeuvre, provided equivalent total duration and aggressiveness of disturbance rejection. This latter property is difficult to quantify, and so therefore an experimental comparison is difficult to fairly perform.

The backstepping derived controllers all asymptotically constrain $\dot{v}_x$, $\ddot{\phi}$, and $\ddot{\theta}_p$ accelerations, which will in turn act to indirectly minimise $\|\tau\|_2^2$. This controller does not incorporate any mechanism for the planning of the most efficient trajectories through multiple waypoints. The MPC approach, on the other hand, is able to both directly optimally minimise $\|\tau\|_2^2$, whilst using advance reference knowledge to optimise for minimal control action over a receding horizon. The differential flatness based trajectory planning approach is able to perform this minimisation in a more global sense, finding the smoothest possible path achievable using the given polynomial basis between any number of waypoints. This optimal smoothness is, however, only an analogue to minimising $\|\tau\|_2^2$.

8.1.4 Safety

The safety of a controller is considered here in terms of the ability to maintain closed-loop stability in the presence of input and output constraints. The nonlinear backstepping control approach provides stability and convergence guarantees for the full nonlinear plant, though conservative acceleration constraint selection is required in order to avoid wheel slip. The MPC approach maintains a stabil-

ity proof whilst directly observing wheel torque constraints, though this proof only holds for a linearisation of the CMD dynamics. The TVLQR controller in Chapter 7 provides no such constrained stability guarantee, and can therefore only operate for small deviations from the planned trajectory, meaning large deviations much be addressed by sufficiently fast trajectory replanning.

8.1.5 Smoothness and Grace of Motion

In user-facing applications smoothness and decisiveness of motion may be an important criteria in control performance. Here, the 'gracefulness' of a trajectory is considered as the ability for a controller to achieve a desired motion with minimal perceived control 'action', such as perceived changes in lean angle direction.

While in simulation the backstepping derived controllers yield smooth and decisive trajectories, in practice model mismatch causes the inertial velocity and position controllers to exhibit a small consistent variation in lean angle whilst maintaining $\|v\|\dot{\phi} \neq 0$. A lack of future reference knowledge utilisation prevents these controllers from taking smoother paths through multiple waypoints.

The enforcement of torque and velocity constraints tends Chapter 6's model predictive controller toward aggressive transitions between constant and varying torque and velocity trajectories, and no cost is associated with changes in input, resulting in input trajectories with unbounded first derivatives. It's receding reference previewing horizon does, however, allow some trajectory optimisation based on knowledge of future waypoints.

The smoothest and most graceful motion achieved in this thesis is that using Chapter 7's differential flatness based approach, as this planner directly optimises for trajectory smoothness, and does not suffer from the same performance degradation due to model mismatch as the backstepping derived approaches. This planner finds the globally optimally smooth trajectory through any number of waypoints, making maximal use of advanced reference knowledge to yield sensical and direct paths.

8.1.6 Map Navigation

It is expected that in future the CMD will be required to navigate a 2D occupancy map via some provided waypoints, whilst avoiding collisions with obstacles. This

will therefore require the additional enforcement of positional constraints on the system.

The backstepping derived nonlinear controllers of Chapter 5 are poorly suited to this task, as no straightforward method is provided for the incorporation of position constraints. The MPC approach is much more amenable to this task, with the possibility of enforcing positional constraints in the same way as velocity constraints are already enforced, provided the chosen set of path constraints remains convex. The differential flatness and polynomial based planning techniques of Chapter 7 are even better suited, allowing the planning of smooth trajectories around obstacles, whilst enforcing positional constraints using the same techniques as used in velocity constraint enforcement. Again, this constraint set must remain convex for each polynomial, so to navigate through a complex map with multiple route options a sampling-based planner must be used to select the desired convex obstacle corridor.

8.1.7 A Ranking of Controllers & Application Suitability

Table 8.1 aims to roughly assign a ranking to each of the presented controllers against the above qualitative metrics. Conclusion as to controller suitability for likely applications are then drawn. For use as a personal vehicle or teleoperated platform, safety and smoothness of motion are of greatest importance, making the backstepping controller of Chapter 5 the most suitable choice. For a fast reactive or local planning application, such as an autonomous robot that is to track a moving reference, the MPC controller of Chapter 6 is most suitable, achieving fast setpoint tracking through optimal constraint satisfaction and toward bang-bang control. Finally, for fully autonomous applications requiring the navigation of an obstacle filled map, the differential flatness and polynomial based planning techniques of Chapter 7 are most suitable, as these allow for straightforward incorporation of obstacle avoidance whilst producing maximally smooth trajectories.

8.2 Future Work

While this thesis has attempted to comprehensively analyse and control the CMD, a large number of areas of future research still exist, with four select

Table 8.1: A qualitative ranking of the three presented controllers against a number of metrics, where a value of 1 denotes the most suitable controller for a given metric.

Metric	Feedback Linearised Nonlinear Control	MPC	Differential Flatness Based Planning
Aggressiveness	3	1	2
Computational Complexity	1	2	3
Power requirements	3	1	2
Safety	1	2	3
Smoothness of Motion	2	3	1
Map Navigation	3	2	1

topics listed as follows:

8.2.1 Improved Mecanum Wheel Modelling and Identification, End-to-End CMD Design Optimisation

Both the optimisation of CMD parameters in Chapter 3, and to varying degrees all three of the model-based controllers derived in this thesis, are sensitive to the accuracy of the CMD dynamics model. As Mecanum wheels have complex and difficult to model friction properties, there will be a gap between the behaviours captured by the purely linear friction models used in this thesis and the true dynamics. It would therefore be useful to improve this modelling, ideally with validation using a single Mecanum wheel under controlled test conditions. This would likely be achieved by mounting a Mecanum wheel above a moving treadmill, with system identification performed over varying heading angle, normal force, applied torque, and wheel geometry. This data could feed back into the design of a more optimal Mecanum wheel, which could again be validated using the test rig. This analysis could also be extending to varying ground materials, potentially allowing for predictive adaptation to varying terrains.

With the CMD friction models improved and verified, an end-to-end multi-disciplinary optimisation of all CMD parameters could be performed to improve

performance for a given specification. The value in performing such an optimisation is set out in Chapter 3, potentially allowing for significant efficiency and performance improvements, as well as a reduction in system cost.

8.2.2 Functional Safety

Being able to prove that a dynamically stable mobile robot is sufficiently safe for use in public spaces is a key prerequisite to their significant commercial adoption, as clearly a dynamically stable system will naturally topple upon removal of power. This will include both proving safe failure modes, and proving either single or even multiple fault tolerance. Future work would explore possible mechanical systems to achieve static stability under complete power failure, along with exploring control schemes for safely handling failure of potentially all but one wheel actuator, despite the subsequent loss of system controllability.

Safety must also be considered in the event of external disturbance - after all, any mobile robot will fall over if pushed hard enough. There could be interesting research in devising a testing methodology and set of requirements that a dynamically stable system must meet in order to present an equivalent level of disturbance resistance to an equivalent statically stable mobile robot. If one could prove that a dynamically stable system required the same force to push over as an existing commercial robot, then the argument against the safety of such a system is diminished.

8.2.3 Shape-aware Trajectory Planning for Autonomous Navigation of Cluttered Environments

While Chapter 7 demonstrates optimally smooth trajectory planning between waypoints, no method has been presented for the selection of these waypoints. In reality this will have to be performed in an autonomous manner given a map of an environment, allowing autonomous navigation between distant locations in a cluttered environment. In the literature this is often achieved using 2D kinematic path planners such as RRT* and A* to find the shortest connecting path, which is then subsampled into waypoints for use in the existing planner. However, all of these methods simply model the mobile system as a sphere, as this is a good approximation for commonly studied systems such as quadrotors. This is not suitable for this application, as a bounding sphere will prevent the CMD from

travelling through gaps smaller than its length along $\hat{b}_x$, failing to exploit its ability to small navigate gaps between obstacles using direct lateral translation.

A CMD specific path planner will therefore need to be shape-aware, presenting a problem more similar to that commonly explored in the manipulator trajectory planning literature.

8.2.4 The Holonomic Collinear Mecanum Drive

The CMD has a number of disadvantages compared to existing statically stable omnidirectional robots. It is underactuated and non-minimum phase, meaning it must first move away from a desired direction of travel to generate a non-zero lean angle and subsequent sustained acceleration. This, combined with the instability of the upright equilibrium, means the system must constantly manipulate its wheels and thus move about a target position in order to maintain balance. As a shape-accelerated system the CMD cannot generate forward acceleration without leaning, meaning any attached cameras or interfaces cannot maintain a constant elevation, and this type of motion makes it clear to a bystander that this system is only dynamically stable, and could therefore fall on software error or loss of power. Also, as fast deceleration can only be achieved by leaning, the system may fall if an obstacle blocks a movement of the wheels required to maintain balance. These effects result in reduced control bandwidth, increased energy consumption, and poor user perception due to this clear instability about the upright pose.

A solution to this problem was invented by the author toward the end of this research, referred to as the Holonomic Collinear Mecanum Drive (HCMD), protected by UK patent application number 1901297.0. This invention combines concepts from the CMD and a traditional reaction wheel inverted pendulum to achieve full actuation of the overall system. A reaction wheel inverted pendulum balances by applying a torque to a high inertia mass that is mounted to the inverted pendulum body, generating an opposing counter torque that can be controlled to modulate the pendulum's lean angle. An example of a HCMD using a reaction wheel is shown in Figure 8.1. A similar system can also be created in which a time invariant torque on the body can be generated by replacing the reaction wheel with a fixed mass offset from the center of rotation. By manipulating the position of this mass the overall center of mass of the system can be altered, enabling non-zero lean angles to be maintained in steady state.

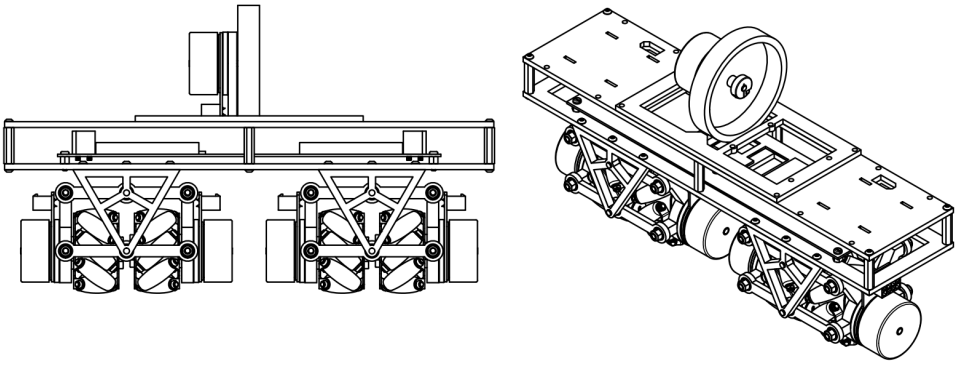


Figure 8.1: An example of a Holonomic Collinear Mecanum Drive using a reaction wheel. While here the reaction wheel is located above the wheel axle, it could actually be located coaxially with the Mecanum wheels, allowing for a compact modular design.

Through appropriate control of the HCMD's reaction wheel or actuated mass the requirement for the robot to lean to accelerate can be completely removed, which combined with the CMD's existing omnidirectionality renders the HCMD holonomic. This means that the system is able to instantaneously generate sustained acceleration in any direction whilst still balancing. Holonomy is a property only shared with statically stable omnidirectional robots, which require a much larger ground footprint.

Such a holonomic dynamically balancing system has never been demonstrated, either theoretically or experimentally, and will represent a step-change in the design and performance of a broad range of mobile robots.

Bibliography

[1] D. A. Winter, *Biomechanics and motor control of human movement*. Wiley, 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470549148

[2] R.W.Brockett, "Asymptotic stability and feedback stabilization," in *Differential Geometric Control Theory*, 1983, pp. 181–191. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.324.9912

[3] B. E. Ilon, "Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base," Patent US3 876 255, 1975. [Online]. Available: https://patents.google.com/patent/US3876255

[4] A. Gfrerrer, "Geometry and kinematics of the mecanum wheel," *Computer Aided Geometric Design*, vol. 25, no. 9, pp. 784–791, Dec. 2008. [Online]. Available: http://www.geometrie.tugraz.at/gfrerrer/publications/MecanumWheel.pdf

[5] L. Ferriere, B. Raucent, and G. Campion, "Design of omnimobile robot wheels," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4. IEEE, 2002, pp. 3664–3670. [Online]. Available: http://ieeexplore.ieee.org/document/509271/

[6] S. Dickerson and B. Lapin, "Control of an omni-directional robotic vehicle with mecanum wheels," in *NTC '91 - National Telesystems Conference Proceedings*. IEEE, 1991, pp. 323–328. [Online]. Available: http://ieeexplore.ieee.org/document/148039/

[7] U. Nagarajan, G. Kantor, and R. L. Hollis, "Human-robot physical interaction with dynamically stable mobile robots," in *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction -*

HRI '09. New York, New York, USA: ACM Press, 2009, p. 205. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1514095.1514176

[8] M. S. Ju and J. M. Mansour, "Comparison of methods for developing the dynamics of rigid-body systems," *The International Journal of Robotics Research*, vol. 8, no. 6, pp. 19–27, 1989. [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1177/027836498900800602

[9] L. A. Sandino, M. Bejar, and A. Ollero, "A survey on methods for elaborated modeling of the mechanics of a small-size helicopter. analysis and comparison," vol. 72, no. 2, pp. 219–238, 2013. [Online]. Available: https://link.springer.com/article/10.1007/s10846-013-9821-y

[10] B. Siciliano, L. Sciavicco, L. Villani, and O. Giuseppe, *Robotics : Modelling, Planning and Control*. Springer, 2009. [Online]. Available: https://www.springer.com/gp/book/9781846286414

[11] K. Thanjavur and R. Rajagopalan, "Ease of dynamic modelling of wheeled mobile robots (WMRS) using Kane's approach," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. IEEE, 2002, pp. 2926–2931. [Online]. Available: http://ieeexplore.ieee.org/document/606731/

[12] M. W. Spong, "Underactuated mechanical systems," in *Control Problems in Robotics and Automation*, vol. 230, 1998, pp. 135–150. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=90248BB5065A6863DAD1B87872AB8166?doi=10.1.1.51.3969{&}rep=rep1{&}type=pdf

[13] ——, "Energy based control of a class of underactuated mechanical systems," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 2828–2832, Jun. 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.467.5126&rep=rep1&type=pdf

[14] A. Isidori, *Nonlinear control systems : an introduction*. Springer-Verlag, 1989. [Online]. Available: https://www.springer.com/gp/book/9783540199168

[15] I. Fantoni and R. Lozano, *Non-linear Control for Underactuated Mechanical Systems*. Springer-Verlag London, 2002. [Online]. Available: https://www.springer.com/gp/book/9781447110866

[16] R. Olfati-Saber, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2001. [Online]. Available: https://dspace.mit.edu/handle/1721.1/8979

[17] S. Devasia and B. Paden, "Exact output tracking for nonlinear time-varying systems," in *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, vol. 3. IEEE, 1994, pp. 2346–2355. [Online]. Available: http://ieeexplore.ieee.org/document/411465/

[18] L. Benvenuti, M. D. di Benedetto, and J. W. Grizzle, "Approximate output tracking for nonlinear non-minimum phase systems with an application to flight control," *International Journal of Robust and Nonlinear Control*, vol. 4, no. 3, pp. 397–414, Jan. 1994. [Online]. Available: http://doi.wiley.com/10.1002/rnc.4590040307

[19] D. Chen and B. Paden, "Stable inversion of nonlinear non-minimum phase systems," *International Journal of Control*, vol. 64, no. 1, pp. 81–97, May 1996. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207179608921618

[20] M. Fliess, J. Levine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, Jun. 1995. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207179508921959

[21] M. van Nieuwstadt and R. M. Murray, "Real time trajectory generation for differentially flat systems," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 2301–2306, Jun. 1996. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-1239%28199809%298%3A11%3C995%3A%3AAID-RNC373%3E3.0.CO%3B2-W

[22] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Springer Tracts in Advanced Robotics*, vol. 114, 2016, pp. 649–666. [Online]. Available: https://link.springer.com/chapter/10.1007{%}2F978-3-319-28872-7{_}37

[23] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 2520–2525. [Online]. Available: http://ieeexplore.ieee.org/document/5980409/

[24] S. Ramasamy, G. Wu, and K. Sreenath, "Dynamically feasible motion planning through partial differential flatness," *Robotics: Science and Systems*, 2014. [Online]. Available: http://www.roboticsproceedings.org/rss10/p53.pdf

[25] C. Sferrazza, D. Pardo, and J. Buchli, "Numerical search for local (partial) differential flatness," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, 2016, pp. 3640–3646. [Online]. Available: https://ieeexplore.ieee.org/document/7759536

[26] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *2000 IEEE Int'l Conf. on Robotics and Automation (ICRA 2000)*, 2002, pp. 995–1001. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/844730

[27] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: http://ieeexplore.ieee.org/document/4082128/

[28] L. E. Kavraki, P. Svestka, J. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug 1996. [Online]. Available: https://ieeexplore.ieee.org/document/508439

[29] C. Warren, "Global path planning using artificial potential fields," in *1989*

International Conference on Robotics and Automation, 1989. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/100007/

[30] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013, pp. 5054–5061. [Online]. Available: http://ieeexplore.ieee.org/document/6631299/

[31] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *Society for Industrial and Applied Mathematics*, vol. 59, no. 4, pp. 849–904, 2017. [Online]. Available: http://www.siam.org/journals/ojsa.php

[32] B. Triggs, "Motion planning for nonholonomic vehicles: An introduction," Oxford University Robotics Group, Tech. Rep., 1993. [Online]. Available: https://hal.inria.fr/inria-00548415/document

[33] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. MIT Press, 2004, vol. 23. [Online]. Available: https://mitpress.mit.edu/books/introduction-autonomous-mobile-robots-second-edition

[34] Z. Hendzel and Rykała, "Modelling of dynamics of a wheeled mobile robot with Mecanum wheels with the use of Lagrange equations of the second kind," *International Journal of Applied Mechanics and Engineering*, vol. 22, no. 1, pp. 81–99, 2017. [Online]. Available: https://www.degruyter.com/dg/viewarticle/j$002fijame.2017.22.issue-1$002fijame-2017-0005$002fijame-2017-0005.xml

[35] N. Tlale and M. D. Villiers, "Kinematics and dynamics modelling of a Mecanum wheeled mobile platform," in *15th International Conference on Mechatronics and Machine Vision in Practice, M2VIP'08*. IEEE, Dec. 2008, pp. 657–662. [Online]. Available: http://ieeexplore.ieee.org/document/4749608/

[36] M. D. Correia, A. Gustavo, and S. Conceição, "Modeling of a three wheeled omnidirectional robot including friction models," *IFAC*

Proceedings Volumes, vol. 45, no. 22, pp. 7–12, 2012. [Online]. Available: https://doi.org/10.3182/20120905-3-HR-2030.00002

[37] K. L. Han, H. Kim, and J. S. Lee, "The sources of position errors of omni-directional mobile robot with Mecanum wheel," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*. IEEE, Oct. 2010, pp. 581–586. [Online]. Available: http://ieeexplore.ieee.org/document/5642009/

[38] O. Purwin and R. D'Andrea, "Trajectory generation and control for four wheeled omnidirectional vehicles," *Robotics and Autonomous Systems*, vol. 54, no. 1, pp. 13–22, 2006. [Online]. Available: https://ieeexplore.ieee.org/document/1470795

[39] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 2009, pp. 2427–2433. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5354805

[40] C. Sprunk, "Highly accurate mobile robot navigation," Ph.D. dissertation, Albert-Ludwigs-University of Freiburg, 2015. [Online]. Available: http://ais.informatik.uni-freiburg.de/publications/papers/sprunk15phd.pdf

[41] M. Dakulovic, C. Sprunk, L. Spinello, I. Petrovic, and W. Burgard, "Efficient navigation for anyshape holonomic mobile robots in dynamic environments," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 2644–2649. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6696729

[42] SoftBank Robotics. Pepper the humanoid robot. Online. Accessed: 2019-08-19. [Online]. Available: https://www.softbankrobotics.com/emea/en/pepper

[43] R. P. M. Chan, K. A. Stol, and C. R. Halkyard, "Review of modelling and control of two-wheeled robots," *Annual Reviews in Control*, vol. 37, no. 1, pp. 89–103, Apr. 2013. [Online]. Available: https://doi.org/10.1016/j.arcontrol.2013.03.004

[44] O. Boubaker, "The inverted pendulum benchmark in nonlinear control theory: A survey," *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, p. 233, May 2013. [Online]. Available: http://journals.sagepub.com/doi/10.5772/55058

[45] Z. Li, C. Yang, and L. Fan, *Advanced control of wheeled inverted pendulum systems*. Springer, 2013. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4471-2963-9

[46] F. Grasser, A. D'Arrigo, S. Colombi, and A. C. Rufer, "JOE: A mobile, inverted pendulum," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 1, pp. 107–114, 2002. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/982254

[47] L. Jingtao, G. Xueshan, H. Qiang, D. Qinjun, and D. Xingguang, "Mechanical design and dynamic modeling of a two-wheeled inverted pendulum mobile robot," in *Proceedings of the IEEE International Conference on Automation and Logistics, ICAL 2007*. IEEE, aug 2007, pp. 1614–1619. [Online]. Available: http://ieeexplore.ieee.org/document/4338830/

[48] S. Y. Seo, S. H. Kim, S. H. Lee, S. H. Han, and H. S. Kim, "Simulation of attitude control of a wheeled inverted pendulum," in *ICCAS 2007 - International Conference on Control, Automation and Systems*. IEEE, 2007, pp. 2264–2269. [Online]. Available: http://ieeexplore.ieee.org/document/4406742/

[49] K. M. Goher, M. O. Tokhi, and N. H. Siddique, "Dynamic modeling and control of a two wheeled robotic vehicle with a virtual payload," *ARPN Journal of Engineering and Applied Sciences*, vol. 6, no. 3, pp. 7–41, 2011.

[50] S. Kim and S. J. Kwon, "Dynamic modeling of a two-wheeled inverted pendulum balancing mobile robot," *International Journal of Control, Automation and Systems*, vol. 13, no. 4, pp. 926–933, aug 2015. [Online]. Available: http://link.springer.com/10.1007/s12555-014-0564-8

[51] K. Pathak, J. Franch, and S. K. S. Agrawal, "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization," *IEEE*

Transactions on Robotics, vol. 21, no. 3, pp. 505–513, Jun. 2005. [Online]. Available: http://ieeexplore.ieee.org/document/1435497/

[52] M. Muhammad, S. Buyamin, M. N. Ahmad, and S. W. Nawawi, "Dynamic modeling and analysis of a two-wheeled inverted pendulum robot," in *Proceedings - CIMSim 2011: 3rd International Conference on Computational Intelligence, Modelling and Simulation*. IEEE, Sep. 2011, pp. 159–164. [Online]. Available: http://ieeexplore.ieee.org/document/6076349/

[53] Y. Kim, S. H. Kim, and Y. K. Kwak, "Dynamic analysis of a nonholonomic two-wheeled inverted pendulum robot," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 44, no. 1, pp. 25–46, Sep. 2005. [Online]. Available: http://link.springer.com/10.1007/s10846-005-9022-4

[54] J. Jahaya, S. W. Nawawi, and Z. Ibrahim, "Multi input single output closed loop identification of two wheel inverted pendulum mobile robot," in *Proceedings - 2011 IEEE Student Conference on Research and Development, SCOReD 2011*. IEEE, Dec. 2011, pp. 138–143. [Online]. Available: http://ieeexplore.ieee.org/document/6148723/

[55] A. Shimada and N. Hatakeyama, "High-speed motion control of wheeled inverted pendulum robots," in *Mechatronics, 2007 IEEE International Conference on*. IEEE, May 2007, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/4280028/

[56] R. M. Brisilla and V. Sankaranarayanan, "Nonlinear control of mobile inverted pendulum," *Robotics and Autonomous Systems*, vol. 70, pp. 145–155, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2015.02.012

[57] Y.-S. Ha and S. Yuta, "Trajectory tracking control for navigation of the inverse pendulum type self-contained mobile robot," *Robotics and Autonomous Systems*, vol. 17, no. 1-2, pp. 65–80, Apr. 1996. [Online]. Available: https://doi.org/10.1016/0921-8890(95)00062-3

[58] J. Huang, Z. H. Guan, T. Matsuno, T. Fukuda, and K. Sekiyama, "Sliding-mode velocity control of mobile-wheeled inverted-pendulum

systems," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 750–758, aug 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5512655/

[59] M. Yue, C. An, and J.-Z. Sun, "An efficient model predictive control for trajectory tracking of wheeled inverted pendulum vehicles with various physical constraints," *International Journal of Control, Automation and Systems*, vol. 18, no. 1, pp. 265–274, 2018. [Online]. Available: http://dx.doi.org/10.1007/s12555-016-0393-zhttp://www.springer.com/12555

[60] N. Dini and V. J. Majd, "Model predictive control of a wheeled inverted pendulum robot," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*. IEEE, Oct. 2015, pp. 152–157. [Online]. Available: http://ieeexplore.ieee.org/document/7367776/

[61] N. Hirose, R. Tajima, N. Koyama, K. Sukigara, and M. Tanaka, "Following control approach based on model predictive control for wheeled inverted pendulum robot," *ADVANCED ROBOTICS*, vol. 30, no. 6, pp. 374–385, 2016. [Online]. Available: http://dx.doi.org/10.1080/01691864.2016.1141115

[62] Z. Li and C. Xu, "Adaptive fuzzy logic control of dynamic balance and motion for wheeled inverted pendulums," *Fuzzy Sets and Systems*, vol. 160, no. 12, pp. 1787–1803, Jun. 2009. [Online]. Available: https://doi.org/10.1016/j.fss.2008.09.013

[63] C.-H. Huang, W.-J. Wang, and C.-H. Chiu, "Design and implementation of fuzzy control on a two-wheel inverted pendulum," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 2988–3001, Jul. 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5556004/

[64] S. Jung and S. S. Kim, "Control experiment of a wheel-driven mobile inverted pendulum using neural network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 297–303, Mar. 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4431881/

[65] C. Yang, Z. Li, R. Cui, and B. Xu, "Neural network-based motion control of an underactuated wheeled inverted pendulum model," *IEEE Transactions on Neural Networks and Learning Systems*,

vol. 25, no. 11, pp. 2004–2016, Nov. 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6762995/

[66] R. P. M. Chan, "Traction control of two-wheeled robots with model predictive control," Ph.D. dissertation, 2015. [Online]. Available: http://hdl.handle.net/2292/27667

[67] Boston Dynamics. Handle. Online. Accessed: 2019-08-19. [Online]. Available: https://www.bostondynamics.com/handle

[68] Ralph L. Hollis, "Dynamic balancing mobile robot," Patent US7 847 504B2, Oct., 2007. [Online]. Available: https://patents.google.com/patent/US7847504B2/en

[69] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot," B.S. thesis, Eidgenössische Technische Hochschule Zürich, 2010. [Online]. Available: https://doi.org/10.3929/ethz-a-010056685

[70] U. Nagarajan, G. Kantor, and R. Hollis, "The ballbot: An omnidirectional balancing mobile robot," *The International Journal of Robotics Research*, vol. 33, no. 6, pp. 917–930, May 2014. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913509126

[71] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," in *2008 International Conference on Control, Automation and Systems, ICCAS 2008*. IEEE, Oct. 2008, pp. 433–438. [Online]. Available: http://ieeexplore.ieee.org/document/4694680/

[72] G. Seyfarth, A. Bhatia, O. Sassnick, M. Shomin, M. Kumagai, and R. Hollis, "Initial results for a ballbot driven with a spherical induction motor," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June. IEEE, May 2016, pp. 3771–3776. [Online]. Available: http://ieeexplore.ieee.org/document/7487565/

[73] M. Shomin and R. Hollis, "Fast, dynamic trajectory planning for a dynamically stable mobile robot," in *IEEE International Conference on Intelligent Robots and Systems*, 2014, pp. 3636–3641. [Online]. Available: https://ieeexplore.ieee.org/document/6943072

[74] U. Nagarajan, G. Kantor, and R. L. Hollis, "Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009, pp. 3743–3748. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5152624

[75] A. N. Inal, O. Morgul, and U. Saranli, "A 3D dynamic model of a spherical wheeled self-balancing robot," in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 5381–5386. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6385689

[76] A. Bonci, "New dynamic model for a ballbot system," in *MESA 2016 - 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications - Conference Proceedings*. IEEE, aug 2016, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7587176/

[77] U. Nagarajan, "Fast and graceful balancing mobile robots," Ph.D. dissertation, Carnegie Mellon University, 2012. [Online]. Available: https://www.ri.cmu.edu/publications/fast-and-graceful-balancing-mobile-robots/

[78] L. Hertig, D. Schindler, M. Bloesch, C. D. Remy, and R. Siegwart, "Unified state estimation for a ballbot," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, May 2013, pp. 2471–2476. [Online]. Available: http://ieeexplore.ieee.org/document/6630913/

[79] L. A. University of California. RoMeLa. Online. Accessed: 2019-08-19. [Online]. Available: http://www.romela.org/robots/

[80] Honda. Honda Global - UNI-CUB. Online. Accessed: 2019-08-19. [Online]. Available: https://global.honda/innovation/robotics/UNI-CUB.html

[81] M. Shomin and R. Hollis, "Differentially flat trajectory generation for a dynamically stable mobile robot," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, May 2013, pp. 4467–4472. [Online]. Available: http://ieeexplore.ieee.org/document/6631211/

[82] U. Nagarajan, "Dynamic constraint-based optimal shape trajectory planner for shape-accelerated underactuated balancing systems," *Robotics: Science*

and Systems, 2010. [Online]. Available: http://www.roboticsproceedings.org/rss06/p31.pdf

[83] D. Pardo, L. Moller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating direct transcription and nonlinear optimization methods for robot motion planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 946–953, Jul. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7400956/

[84] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June. IEEE, May 2016, pp. 1398–1404. [Online]. Available: http://ieeexplore.ieee.org/document/7487274/

[85] S. Reynolds-Haertle and M. Stilman, "Design and development of a dynamically-balancing holonomic robot," Georgia Institute of Technology. Center for Robotics and Intelligent Machines, Tech. Rep., 2011. [Online]. Available: https://smartech.gatech.edu/handle/1853/41706

[86] K. Zimmermann, I. Zeidis, and M. Abdelrahman, "Dynamics of mechanical systems with Mecanum wheels," in *Springer Proceedings in Mathematics and Statistics*, vol. 93, 2014, pp. 269–279. [Online]. Available: https://doi.org/10.1007/978-3-319-08266-0_19

[87] J. Dormand and P. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19 – 26, 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0771050X80900133

[88] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017. [Online]. Available: https://books.google.co.uk/books?id=8uS3AQAACAAJ

[89] H. H. Nijmeijer and A. J. van der. Schaft, *Nonlinear dynamical control systems*. Springer-Verlag, 1990. [Online]. Available: https://www.springer.com/gp/book/9780387972343

[90] A. Salerno and J. Angeles, "A new family of two-wheeled mobile robots: Modeling and controllability," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 169–173, Feb. 2007. [Online]. Available: http://ieeexplore.ieee.org/document/4084575/

[91] A. Salerno, "Design, dynamics and control of a fast two-wheeled quasiholonomic robot," Ph.D. dissertation, McGill University, 2006. [Online]. Available: https://core.ac.uk/download/pdf/41887250.pdf

[92] R. Marino, "On the largest feedback linearizable subsystem," *Systems and Control Letters*, vol. 6, no. 5, pp. 345–351, Jan. 1986. [Online]. Available: https://doi.org/10.1016/0167-6911(86)90130-1

[93] A. J. Fossard and D. Normand-Cyrot, *Nonlinear Systems : Control 3*. Springer US, 1997. [Online]. Available: https://doi.org/10.1007/978-1-4615-6395-2

[94] ArduCopter. (2019) ArduCopter complete parameter documentation. [Online]. Available: http://ardupilot.org/copter/docs/parameters.html

[95] *MPU9250 Product Specification Revision 1.1*, Online, InvenSense, 2016. [Online]. Available: http://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

[96] *NI myRIO-1900 User Guide and Specifications*, Online, National Instruments. [Online]. Available: http://www.ni.com/pdf/manuals/376047d.pdf

[97] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1271397

[98] A. Isidori, *Nonlinear Control Systems*, ser. Communications and Control Engineering. London: Springer London, 1995. [Online]. Available: http://link.springer.com/10.1007/978-1-84628-615-5

[99] B. Srinivasan, P. Huguenin, and D. Bonvin, "Global stabilization of an inverted pendulum–control strategy and experimental verification,"

Automatica, vol. 45, no. 1, pp. 265–269, Jan. 2009. [Online]. Available: https://doi.org/10.1016/j.automatica.2008.07.004

[100] Wei Zhong and H. Rock, "Energy and passivity based control of the double inverted pendulum on a cart," in *Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01) (Cat. No.01CH37204)*. IEEE, 2002, pp. 896–901. [Online]. Available: http://ieeexplore.ieee.org/document/973983/

[101] M. W. Spong, P. Corke, and R. Lozano, "Nonlinear control of the reaction wheel pendulum," *Automatica*, vol. 37, no. 11, pp. 1845–1851, Nov. 2001. [Online]. Available: https://doi.org/10.1016/S0005-1098(01)00145-5

[102] M. W. Spong, "The swing up control problem for the acrobot," *IEEE Control Systems*, vol. 15, no. 1, pp. 49–55, Feb. 1995. [Online]. Available: https://ieeexplore.ieee.org/document/341864/

[103] J. Hauser and R. M. Murray, "Nonlinear controllers for non-integrable systems: the acrobot example," in *1990 American Control Conference*. IEEE, May 1990, pp. 669–671. [Online]. Available: https://ieeexplore.ieee.org/document/4790817/

[104] M. A. Karkoub and M. Parent, "Modelling and non-linear feedback stabilization of a two-wheel vehicle," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 218, no. 8, pp. 675–686, Dec. 2004. [Online]. Available: http://journals.sagepub.com/doi/10.1177/095965180421800807

[105] A. Mokhtari and A. Benallegue, "Dynamic feedback controller of Euler angles and wind parameters estimation for a quadrotor unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004.* IEEE, 2004, pp. 2359–2366 Vol.3. [Online]. Available: http://ieeexplore.ieee.org/document/1307414/

[106] A. Benallegue, A. Mokhtari, and L. Fridman, "Feedback linearization and high order sliding mode observer for a quadrotor UAV," in *Proceedings of the 2006 International Workshop on Variable Structure*

Systems, VSS'06, vol. 2006. IEEE, 2006, pp. 365–372. [Online]. Available: http://ieeexplore.ieee.org/document/1644545/

[107] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 1996, vol. 3.

[108] D. J. Leith and W. E. Leithead, "Survey of gain-scheduling analysis and design," *International Journal of Control*, vol. 73, no. 11, pp. 1001–1025, Jan. 2000. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/002071700411304

[109] R. Xu and U. Ozguner, "Sliding mode control of a quadrotor helicopter," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 4957–4962. [Online]. Available: http://ieeexplore.ieee.org/document/4177181/

[110] M. A. Khanesar, M. Teshnehlab, and M. A. Shoorehdeli, "Sliding mode control of rotary inverted pendulm," in *2007 Mediterranean Conference on Control and Automation, MED*. IEEE, Jun. 2007, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/4433653/

[111] K. P. Tee, S. S. Ge, and E. H. Tay, "Barrier Lyapunov functions for the control of output-constrained nonlinear systems," *Automatica*, vol. 45, no. 4, pp. 918–927, 2009. [Online]. Available: https://doi.org/10.1016/j.automatica.2008.11.017

[112] A. Mills, A. Wills, and B. Ninness, "Nonlinear model predictive control of an inverted pendulum," in *Proceedings of the American Control Conference*. IEEE, 2009, pp. 2335–2340. [Online]. Available: http://ieeexplore.ieee.org/document/5160391/

[113] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000. [Online]. Available: https://doi.org/10.1016/S0005-1098(99)00214-9

[114] J. A. Rossiter, *Model-Based Predictive Control: A Practical Approach*. Sheffield, UK: CRC Press, 2003. [Online]. Available: https://doi.org/10.1201/9781315272610

[115] J. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002. [Online]. Available: http://www-control.eng.cam.ac.uk/jmm/mpcbook/mpcbook.html

[116] L. Magni, R. Scattolini, and K. Åström, "Global stabilization of the inverted pendulum using model predictive control," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 141–146, 2002. [Online]. Available: https://doi.org/10.3182/20020721-6-ES-1901.00592

[117] P. J. Gawthrop and L. Wang, "Intermittent predictive control of an inverted pendulum," *Control Engineering Practice*, vol. 14, no. 11, pp. 1347–1356, 2006. [Online]. Available: https://doi.org/10.1016/j.conengprac.2005.09.002

[118] T. Ohhira and A. Shimada, "Model predictive control for an inverted-pendulum robot with time-varying constraints," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 776–781, Jul. 2017. [Online]. Available: https://doi.org/10.1016/j.ifacol.2017.08.252

[119] S. S. Dughman and J. A. Rossiter, "A survey of guaranteeing feasibility and stability in MPC during target changes," in *IFAC-PapersOnLine*, vol. 28, no. 8, 2015, pp. 813–818. [Online]. Available: https://doi.org/10.1016/j.ifacol.2015.09.069

[120] D. Simon, J. Lofberg, and T. Glad, "Reference tracking MPC using terminal set scaling," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, Dec. 2012, pp. 4543–4548. [Online]. Available: http://ieeexplore.ieee.org/document/6426550/

[121] S. S. Dughman and J. A. Rossiter, "Systematic and effective embedding of feedforward of target information into MPC," *International Journal of Control*, pp. 1–15, Jan. 2017. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207179.2017.1281439

[122] B. Pluymers, J. Rossiter, J. Suykens, and B. De Moor, "The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty," in *Proceedings of the 2005, American*

Control Conference, 2005. IEEE, 2005, pp. 804–809. [Online]. Available: http://ieeexplore.ieee.org/document/1470058/

[123] E. G. Gilbert and K. T. Tan, "Linear systems with state and control constraints: The theory and application of maximal output admissible sets," *IEEE Transactions on Automatic Control*, vol. 36, no. 9, pp. 1008–1020, 1991. [Online]. Available: http://ieeexplore.ieee.org/document/83532/

[124] Yongxing Hao, A. Davari, and A. Manesh, "Differential flatness-based trajectory planning for multiple unmanned aerial vehicles using mixed-integer linear programming," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 104–109. [Online]. Available: http://ieeexplore.ieee.org/document/1469916/

[125] G. Valencia-Palomo, M. Pelegrinis, J. A. Rossiter, and R. Gondhalekar, "A move-blocking strategy to improve tracking in predictive control," pp. 6293–6298, Jun. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5531512/

[126] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014. [Online]. Available: https://doi.org/10.1007/s12532-014-0071-1

[127] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, Mar. 1998. [Online]. Available: http://arc.aiaa.org/doi/10.2514/2.4231

[128] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proceedings of the IEEE Conference on Decision and Control.* IEEE, Dec. 2010, pp. 7681–7687. [Online]. Available: http://ieeexplore.ieee.org/document/5717430/

[129] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proceedings - IEEE International Conference on Robotics and Automation.* IEEE,

May 2013, pp. 5054–5061. [Online]. Available: http://ieeexplore.ieee.org/document/6631299/

[130] U. Nagarajan, B. Kim, and R. Hollis, "Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012, pp. 130–135. [Online]. Available: https://ieeexplore.ieee.org/document/6225065

[131] P. J. Davis, *Interpolation and approximation*. Blaisdell Pub. Co, 1963.

[132] S. Prajna, A. Papachristodoulou, P. Seiler, and P. Parrilo, "New developments in sum of squares optimization and SOSTOOLS," in *Proceedings of the 2004 American Control Conference*. IEEE, 2004, pp. 5606–5611 vol.6. [Online]. Available: https://ieeexplore.ieee.org/document/1384747/

[133] G. Polya and G. Szego, *Problems and theorems in analysis*. Springer, 1998. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4757-1640-5

[134] J. Lofberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. IEEE, 2004, pp. 284–289. [Online]. Available: http://ieeexplore.ieee.org/document/1393890/

[135] J. Löfberg, "Pre- and post-processing sum-of-squares programs in practice," *IEEE Transactions on Automatic Control*, vol. 54, no. 5, pp. 1007–1011, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/4908937

[136] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017. [Online]. Available: http://docs.mosek.com/8.1/toolbox/index.html

[137] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June. IEEE, May 2016, pp. 1476–1483. [Online]. Available: http://ieeexplore.ieee.org/document/7487283/

[138] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7839930/

Appendix A

CMD Model Matrix Definitions

A.1 Matrices for CMD Dynamics Model in $(q, \dot{q})$

$$M(q) = \begin{bmatrix} m_{1,1} & 0 & m_{1,3} & m_{1,4} & \\ 0 & m_{2,2} & m_{2,3} & m_{2,4} & 0_{4\times 8} \\ m_{3,1} & m_{3,2} & m_{3,3} & 0 & \\ m_{4,1} & m_{4,2} & 0 & m_{4,4} & \\ & & 0_{8\times 4} & & 0_{8\times 8} \end{bmatrix} \tag{A.1.1}$$

where

$$m_{1,1} = m_p + 4m_w$$

$$m_{1,3} = h_p m_p \cos(\phi)\sin(\theta_p) - l_2 m_w \sin(\phi) - l_3 m_w \sin(\phi) - l_4 m_w \sin(\phi) - l_1 m_w \sin(\phi)$$

$$m_{1,4} = h_p m_p \cos(\theta_p)\sin(\phi)$$

$$m_{2,2} = m_p + 4m_w$$

$$m_{2,3} = l_1 m_w \cos(\phi) + l_2 m_w \cos(\phi) + l_3 m_w \cos(\phi) + l_4 m_w \cos(\phi) + h_p m_p \sin(\phi)\sin(\theta_p)$$

$$m_{2,4} = -h_p m_p \cos(\phi)\cos(\theta_p)$$

$$m_{3,1} = h_p m_p \cos(\phi)\sin(\theta_p) - l_2 m_w \sin(\phi) - l_3 m_w \sin(\phi) - l_4 m_w \sin(\phi) - l_1 m_w \sin(\phi)$$

$$m_{3,2} = l_1 m_w \cos(\phi) + l_2 m_w \cos(\phi) + l_3 m_w \cos(\phi) + l_4 m_w \cos(\phi) + h_p m_p \sin(\phi)\sin(\theta_p)$$

$$m_{3,3} = 4I_{wyz} + I_{pby}\sin(\theta_p)^2 + l_1^2 m_w + l_2^2 m_w + l_3^2 m_w + l_4^2 m_w - I_{pbz}(\sin(\theta_p)^2 - 1) + h_p^2 m_p \sin(\theta_p)^2$$

$$m_{4,1} = h_p m_p \cos(\theta_p) \sin(\phi)$$
$$m_{4,2} = -h_p m_p \cos(\phi) \cos(\theta_p)$$
$$m_{4,4} = m_p h_p^2 + I_{pbx}$$

$$C(q, \dot{q}) = \begin{bmatrix} 0 & 0 & c_{1,3} & c_{1,4} & \\ 0 & 0 & c_{2,3} & c_{2,4} & 0_{4\times 8} \\ 0 & 0 & c_{3,3} & c3,4 & \\ 0 & 0 & c_{4,3} & 0 & \\ & & 0_{8\times 4} & & 0_{8\times 8} \end{bmatrix} \tag{A.1.2}$$

where

$$c_{1,3} = h_p m_p \cos(\phi) \cos(\theta_p)\dot{\theta}_p - l_2 m_w \cos(\phi)\dot{\phi} - l_3 m_w \cos(\phi)\dot{\phi} - l_4 m_w \cos(\phi)\dot{\phi} - l_1 m_w \cos(\phi)\dot{\phi} - h_p m_p \sin(\phi) \sin(\theta_p)\dot{\phi}$$
$$c_{1,4} = h_p m_p \cos(\phi) \cos(\theta_p)\dot{\phi} - h_p m_p \sin(\phi) \sin(\theta_p)\dot{\theta}_p$$
$$c_{2,3} = h_p m_p \cos(\phi) \sin(\theta_p)\dot{\phi} - l_2 m_w \sin(\phi)\dot{\phi} - l_3 m_w \sin(\phi)\dot{\phi} - l_4 m_w \sin(\phi)\dot{\phi} - l_1 m_w \sin(\phi)\dot{\phi} + h_p m_p \cos(\theta_p) \sin(\phi)\dot{\theta}_p$$
$$c_{2,4} = h_p m_p \cos(\theta_p) \sin(\phi)\dot{\phi} + h_p m_p \cos(\phi) \sin(\theta_p)\dot{\theta}_p$$
$$c_{3,3} = 1/2(\sin(2\theta_p)\dot{\theta}_p(m_p h_p^2 + I_{pby} - I_{pbz}))$$
$$c_{3,4} = 1/2(\sin(2\theta_p)\dot{\phi}(m_p h_p^2 + I_{pby} - I_{pbz}))$$
$$c_{4,3} = -1/2(\sin(2\theta_p)\dot{\phi}(m_p h_p^2 + I_{pby} - I_{pbz}))$$

$$G(q) = \begin{bmatrix} 0_{3\times 1} \\ -g h_p m_p \sin(\theta_p) \\ 0_{8\times 1} \end{bmatrix} \tag{A.1.3}$$

$$F = \begin{bmatrix} & 0 & 0 & 0 & 0 & 0 & f_{1,9} & f_{1,10} & f_{1,11} & f_{1,12} \\ & 0 & 0 & 0 & 0 & 0 & f_{2,9} & f_{2,10} & f_{2,11} & f_{2,12} \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & f_{4,4} & k_v w & k_v w & k_v w & k_v w & 0 & 0 & 0 & 0 \\ 0_{8\times 3} & k_{vw} & f_{5,5} & 0 & 0 & 0 & f_{5,9} & 0 & 0 & 0 \\ & k_{vw} & 0 & f_{6,6} & 0 & 0 & 0 & f_{6,10} & 0 & 0 \\ & k_{vw} & 0 & 0 & f_{7,7} & 0 & 0 & 0 & f_{7,11} & 0 \\ & k_{vw} & 0 & 0 & 0 & f_{8,8} & 0 & 0 & 0 & f_{8,12} \\ 0_{4\times 3} & & & & & 0_{4\times 9} & & & & \end{bmatrix} \tag{A.1.4}$$

where

$$f_{1,9} = \frac{k_{vr}\cos(\phi)\sin(\alpha_1)}{r_r - r_w} \qquad f_{1,10} = \frac{k_{vr}\cos(\phi)\sin(\alpha_2)}{r_r - r_w}$$
$$f_{1,11} = \frac{k_{vr}\cos(\phi)\sin(\alpha_3)}{r_r - r_w} \qquad f_{1,12} = \frac{k_{vr}\cos(\phi)\sin(\alpha_4)}{r_r - r_w}$$
$$f_{2,9} = \frac{k_{vr}\sin(\phi)\sin(\alpha_1)}{r_r - r_w} \qquad f_{2,10} = \frac{k_{vr}\sin(\phi)\sin(\alpha_2)}{r_r - r_w}$$
$$f_{2,11} = \frac{k_{vr}\sin(\phi)\sin(\alpha_3)}{r_r - r_w} \qquad f_{2,12} = \frac{k_{vr}\sin(\phi)\sin(\alpha_4)}{r_r - r_w}$$
$$f_{4,4} = -4k_{vw} \qquad f_{5,5} = -k_{rw} - k_{vw}$$
$$f_{5,9} = k_{vr}\cos(\alpha_1) \qquad f_{6,6} = -k_{rw} - k_{vw}$$
$$f_{6,9} = k_{vr}\cos(\alpha_2) \qquad f_{7,7} = -k_{rw} - k_{vw}$$
$$f_{7,9} = k_{vr}\cos(\alpha_3) \qquad f_{8,8} = -k_{rw} - k_{vw}$$
$$f_{8,9} = k_{vr}\cos(\alpha_4)$$

$$B = \begin{bmatrix} 0_{3\times 4} \\ -1_{1\times 4} \\ I_{4\times 4} \\ 0_{4\times 4} \end{bmatrix} \tag{A.1.5}$$

A.2 Matrices for CMD Dynamics Model in $(p, \dot{p})$

$$M_p(p) = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & 0 \\ m_{4,1} & m_{4,2} & 0 & m_{4,4} \end{bmatrix} \tag{A.2.1}$$

where

$$\begin{aligned}
m_{1,1} = {} & m_p + 4m_w - (I_{wx}\cos(\alpha_1+\phi)^2)/(r_w^2(\cos(\alpha_1)^2-1)) \\
& - (I_{wx}\cos(\alpha_2+\phi)^2)/(r_w^2(\cos(\alpha_2)^2-1)) \\
& - (I_{wx}\cos(\alpha_3+\phi)^2)/(r_w^2(\cos(\alpha_3)^2-1)) \\
& - (I_{wx}\cos(\alpha_4+\phi)^2)/(r_w^2(\cos(\alpha_4)^2-1)) \\
m_{1,2} = {} & (I_{wx}\sin(2\alpha_4+2\phi))/(2r_w^2\sin(\alpha_4)^2) + (I_{wx}(\sin(\alpha_1)^2\sin(\alpha_2)^2\sin(2\alpha_3+2\phi) \\
& + \sin(\alpha_1)^2\sin(\alpha_3)^2\sin(2\alpha_2+2\phi) \\
& + \sin(\alpha_2)^2\sin(\alpha_3)^2\sin(2\alpha_1+2\phi)))/(2r_w^2\sin(\alpha_1)^2\sin(\alpha_2)^2\sin(\alpha_3)^2) \\
m_{1,3} = {} & h_p m_p\cos(\phi)\sin(\theta_p) - l_2 m_w\sin(\phi) - l_3 m_w\sin(\phi) - l_4 m_w\sin(\phi) \\
& - l_1 m_w\sin(\phi) + (I_{wx}l_1\cos(\alpha_1+\phi))/(r_w^2\sin(\alpha_1)) \\
& + (I_{wx}l_2\cos(\alpha_2+\phi))/(r_w^2\sin(\alpha_2)) + (I_{wx}l_3\cos(\alpha_3+\phi))/(r_w^2\sin(\alpha_3)) \\
& + (I_{wx}l_4\cos(\alpha_4+\phi))/(r_w^2\sin(\alpha_4)) \\
m_{1,4} = {} & h_p m_p\cos(\theta_p)\sin(\phi) \\
m_{2,1} = {} & (I_{wx}\sin(2\alpha_4+2\phi))/(2r_w^2\sin(\alpha_4)^2) \\
& + (I_{wx}(\sin(\alpha_1)^2\sin(\alpha_2)^2\sin(2\alpha_3+2\phi) + \sin(\alpha_1)^2\sin(\alpha_3)^2\sin(2\alpha_2+2\phi) \\
& + \sin(\alpha_2)^2\sin(\alpha_3)^2\sin(2\alpha_1+2\phi)))/(2r_w^2\sin(\alpha_1)^2\sin(\alpha_2)^2\sin(\alpha_3)^2) \\
m_{2,2} = {} & m_p + 4m_w + (I_{wx}\sin(\alpha_1+\phi)^2)/(r_w^2\sin(\alpha_1)^2) \\
& + (I_{wx}\sin(\alpha_2+\phi)^2)/(r_w^2\sin(\alpha_2)^2) + (I_{wx}\sin(\alpha_3+\phi)^2)/(r_w^2\sin(\alpha_3)^2) \\
& + (I_{wx}\sin(\alpha_4+\phi)^2)/(r_w^2\sin(\alpha_4)^2) \\
m_{2,3} = {} & l_1 m_w\cos(\phi) + l_2 m_w\cos(\phi) + l_3 m_w\cos(\phi) + l_4 m_w\cos(\phi) \\
& + h_p m_p\sin(\phi)\sin(\theta_p) + (I_{wx}l_1\sin(\alpha_1+\phi))/(r_w^2\sin(\alpha_1)) \\
& + (I_{wx}l_2\sin(\alpha_2+\phi))/(r_w^2\sin(\alpha_2)) + (I_{wx}l_3\sin(\alpha_3+\phi))/(r_w^2\sin(\alpha_3)) \\
& + (I_{wx}l_4\sin(\alpha_4+\phi))/(r_w^2\sin(\alpha_4)) \\
m_{2,4} = {} & -h_p m_p\cos(\phi)\cos(\theta_p)
\end{aligned}$$

$$
\begin{aligned}
m_{3,1} = {} & h_p m_p \cos(\phi)\sin(\theta_p) - l_2 m_w \sin(\phi) - l_3 m_w \sin(\phi) - l_4 m_w \sin(\phi) \\
& - l_1 m_w \sin(\phi) + (I_{wx} l_1 \cos(\alpha_1 + \phi))/(r_w^2 \sin(\alpha_1)) \\
& + (I_{wx} l_2 \cos(\alpha_2 + \phi))/(r_w^2 \sin(\alpha_2)) + (I_{wx} l_3 \cos(\alpha_3 + \phi))/(r_w^2 \sin(\alpha_3)) \\
& + (I_{wx} l_4 \cos(\alpha_4 + \phi))/(r_w^2 \sin(\alpha_4)) \\
m_{3,2} = {} & l_1 m_w \cos(\phi) + l_2 m_w \cos(\phi) + l_3 m_w \cos(\phi) + l_4 m_w \cos(\phi) \\
& + h_p m_p \sin(\phi)\sin(\theta_p) + (I_{wx} l_1 \sin(\alpha_1 + \phi))/(r_w^2 \sin(\alpha_1)) \\
& + (I_{wx} l_2 \sin(\alpha_2 + \phi))/(r_w^2 \sin(\alpha_2)) + (I_{wx} l_3 \sin(\alpha_3 + \phi))/(r_w^2 \sin(\alpha_3)) \\
& + (I_{wx} l_4 \sin(\alpha_4 + \phi))/(r_w^2 \sin(\alpha_4)) \\
m_{3,3} = {} & 4 I_{wyz} + (I_{wx} l_1^2 + I_{wx} l_2^2 + I_{wx} l_3^2 + I_{wx} l_4^2)/r_w^2 + I_{pby} \sin(\theta_p)^2 + l_1^2 m_w + l_2^2 m_w \\
& + l_3^2 m_w + l_4^2 m_w - I_{pbz}(\sin(\theta_p)^2 - 1) + h_p^2 m_p \sin(\theta_p)^2 \\
m_{4,1} = {} & h_p m_p \cos(\theta_p)\sin(\phi) \\
m_{4,2} = {} & -h_p m_p \cos(\phi)\cos(\theta_p) \\
m_{4,4} = {} & m_p h_p^2 + I_{pbx}
\end{aligned}
$$

$$
C_p(p, \dot{p}) = \begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} \\ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} \\ 0 & 0 & c_{4,3} & 0 \end{bmatrix} \tag{A.2.2}
$$

$$
\begin{aligned}
c_{1,1} = {} & -(I_{wx} \sin(2\alpha_4 + 2\phi)\dot{\phi})/(2 r_w^2 \sin(\alpha_4)^2) \\
& - (I_{wx}\dot{\phi}(\sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(2\alpha_3 + 2\phi) + \sin(\alpha_1)^2 \sin(\alpha_3)^2 \sin(2\alpha_2 + 2\phi) \\
& + \sin(\alpha_2)^2 \sin(\alpha_3)^2 \sin(2\alpha_1 + 2\phi)))/(2 r_w^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2) \\
c_{1,2} = {} & -(I_{wx} \cos(\alpha_1 + \phi)^2 \dot{\phi})/(r_w^2(\cos(\alpha_1)^2 - 1)) \\
& - (I_{wx} \cos(\alpha_2 + \phi)^2 \dot{\phi})/(r_w^2(\cos(\alpha_2)^2 - 1)) \\
& - (I_{wx} \cos(\alpha_3 + \phi)^2 \dot{\phi})/(r_w^2(\cos(\alpha_3)^2 - 1)) \\
& - (I_{wx} \cos(\alpha_4 + \phi)^2 \dot{\phi})/(r_w^2(\cos(\alpha_4)^2 - 1)) \\
c_{1,3} = {} & h_p m_p \cos(\phi)\cos(\theta_p)\dot{\theta}_p - l_2 m_w \cos(\phi)\dot{\phi} - l_3 m_w \cos(\phi)\dot{\phi} - l_4 m_w \cos(\phi)\dot{\phi} \\
& - l_1 m_w \cos(\phi)\dot{\phi} - h_p m_p \sin(\phi)\sin(\theta_p)\dot{\phi}
\end{aligned}
$$

$$
\begin{aligned}
c_{1,4} &= h_p m_p \cos(phi)\cos(\theta_p)\dot{\phi} - h_p m_p \sin(\phi)\sin(\theta_p)\dot{\theta}_p \\
c_{2,1} &= -(I_{wx}\sin(\alpha_4+\phi)^2\dot{\phi})/(r_w^2\sin(\alpha_4)^2) - (I_{wx}\dot{\phi}(\sin(\alpha_1+\phi)^2\sin(\alpha_2)^2\sin(\alpha_3)^2 \\
&\quad + \sin(\alpha_2+\phi)^2\sin(\alpha_1)^2\sin(\alpha_3)^2 + \sin(\alpha_3+\phi)^2\sin(\alpha_1)^2\sin(\alpha_2)^2)) \\
&\quad /(r_w^2\sin(\alpha_1)^2\sin(\alpha_2)^2\sin(\alpha_3)^2) \\
c_{2,2} &= (I_{wx}\sin(2\alpha_4+2\phi)\dot{\phi})/(2r_w^2\sin(\alpha_4)^2) + (I_{wx}\dot{\phi}(\sin(\alpha_1)^2\sin(\alpha_2)^2\sin(2\alpha_3+2\phi) \\
&\quad + \sin(\alpha_1)^2\sin(\alpha_3)^2\sin(2\alpha_2+2\phi) + \sin(\alpha_2)^2\sin(\alpha_3)^2\sin(2\alpha_1+2\phi))) \\
&\quad /(2r_w^2\sin(\alpha_1)^2\sin(\alpha_2)^2\sin(\alpha_3)^2) \\
c_{2,3} &= h_p m_p \cos(\phi)\sin(\theta_p)\dot{\phi} - l_2 m_w \sin(\phi)\dot{\phi} - l_3 m_w \sin(\phi)\dot{\phi} - l_4 m_w \sin(\phi)\dot{\phi} \\
&\quad - l_1 m_w \sin(\phi)\dot{\phi} + h_p m_p \cos(\theta_p)\sin(\phi)\dot{\theta}_p \\
c_{2,4} &= h_p m_p \cos(\theta_p)\sin(\phi)\dot{\phi} + h_p m_p \cos(\phi)\sin(\theta_p)\dot{\theta}_p \\
c_{3,1} &= -(I_{wx}\dot{\phi}(l_1\cos(\phi) \\
&\quad + l_2\cos(\phi) + l_3\cos(\phi) + l_4\cos(\phi) + l_1\sin(\phi)\cot(\alpha_1) + l_2\sin(\phi)\cot(\alpha_2) \\
&\quad + l_3\sin(\phi)\cot(\alpha_3)))/r_w^2 - (I_{wx}l_4\sin(\phi)\cot(\alpha_4)\dot{\phi})/r_w^2 \\
c_{3,2} &= (I_{wx}l_4\cos(\phi)\cot(\alpha_4)\dot{\phi})/r_w^2 - (I_{wx}\dot{\phi}(l_1\sin(\phi) + l_2\sin(\phi) + l_3\sin(\phi) + l_4\sin(\phi) \\
&\quad - l_1\cos(\phi)\cot(\alpha_1) - l_2\cos(\phi)\cot(\alpha_2) - l_3\cos(\phi)\cot(\alpha_3)))/r_w^2 \\
c_{3,3} &= (\sin(2\theta_p)\dot{\theta}_p(m_p h_p^2 + I_{pby} - I_{pbz}))/2 \\
c_{3,4} &= (\sin(2\theta_p)\dot{\phi}(m_p h_p^2 + I_{pby} - I_{pbz}))/2 \\
c_{4,3} &= -(\sin(2\theta_p)\dot{\phi}(m_p h_p^2 + I_{pby} - I_{pbz}))/2
\end{aligned}
$$

$$
G_p(p) = \begin{bmatrix} 0_{3\times 1} \\ -g h_p m_p \sin(\theta_p) \end{bmatrix} \tag{A.2.3}
$$

$$
F_p(p) = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & f_{1,4} \\ f_{2,1} & f_{2,2} & f_{2,3} & f_{2,4} \\ f_{3,1} & f_{3,2} & f_{3,3} & f_{3,4} \\ f_{4,1} & f_{4,2} & f_{4,3} & f_{4,4} \end{bmatrix} \tag{A.2.4}
$$

where

$$
\begin{aligned}
f_{1,1} = {} & (k_{vr}\cos(\phi)\sin(\phi)\cot(\alpha_1))/(r_r r_w) \\
& - ((k_{rw}+k_{vw})(\cos(2\alpha_1+2\phi)+1))/(2r_w^2\sin(\alpha_1)^2) \\
& - ((k_{rw}+k_{vw})(\cos(2\alpha_2+2\phi)+1))/(2r_w^2\sin(\alpha_2)^2) \\
& - ((k_{rw}+k_{vw})(\cos(2\alpha_3+2\phi)+1))/(2r_w^2\sin(\alpha_3)^2) \\
& - ((k_{rw}+k_{vw})(\cos(2\alpha_4+2\phi)+1))/(2r_w^2\sin(\alpha_4)^2) \\
& - (k_{vr}\cos(\phi)^2\cot(\alpha_1)^2)/(r_r r_w) - (k_{vr}\cos(\phi)^2\cot(\alpha_2)^2)/(r_r r_w) \\
& - (k_{vr}\cos(\phi)^2\cot(\alpha_3)^2)/(r_r r_w) - (k_{vr}\cos(\phi)^2\cot(\alpha_4)^2)/(r_r r_w) \\
& - (4k_{vr}\cos(\phi)^2)/(-r_r^2+r_w r_r) + (k_{vr}\cos(\phi)\sin(\phi)\cot(\alpha_2))/(r_r r_w) \\
& + (k_{vr}\cos(\phi)\sin(\phi)\cot(\alpha_3))/(r_r r_w) + (k_{vr}\cos(\phi)\sin(\phi)\cot(\alpha_4))/(r_r r_w) \\
f_{1,2} = {} & (k_{vr}\cot(\alpha_1))/(2r_r r_w) - (2k_{vr}\sin(2\phi))/(-r_r^2+r_w r_r) + (k_{vr}\cot(\alpha_2))/(2r_r r_w) \\
& + (k_{vr}\cot(\alpha_3))/(2r_r r_w) + (k_{vr}\cot(\alpha_4))/(2r_r r_w) \\
& - (\sin(2\alpha_1+2\phi)(k_{rw}+k_{vw}))/(2r_w^2\sin(\alpha_1)^2) \\
& - (\sin(2\alpha_2+2\phi)(k_{rw}+k_{vw}))/(2r_w^2\sin(\alpha_2)^2) \\
& - (\sin(2\alpha_3+2\phi)(k_{rw}+k_{vw}))/(2r_w^2\sin(\alpha_3)^2) \\
& - (\sin(2\alpha_4+2\phi)(k_{rw}+k_{vw}))/(2r_w^2\sin(\alpha_4)^2) \\
& - (k_{vr}\sin(2\phi)\cot(\alpha_1)^2)/(2r_r r_w) - (k_{vr}\sin(2\phi)\cot(\alpha_2)^2)/(2r_r r_w) \\
& - (k_{vr}\sin(2\phi)\cot(\alpha_3)^2)/(2r_r r_w) - (k_{vr}\sin(2\phi)\cot(\alpha_4)^2)/(2r_r r_w) \\
& - (k_{vr}\cos(2\phi)\cot(\alpha_1))/(2r_r r_w) - (k_{vr}\cos(2\phi)\cot(\alpha_2))/(2r_r r_w) \\
& - (k_{vr}\cos(2\phi)\cot(\alpha_3))/(2r_r r_w) - (k_{vr}\cos(2\phi)\cot(\alpha_4))/(2r_r r_w) \\
f_{1,3} = {} & -(l_4\cos(\alpha_4+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_4)) \\
& - ((k_{rw}+k_{vw})(l_1\cos(\alpha_1+\phi)\sin(\alpha_2)\sin(\alpha_3) \\
& + l_2\cos(\alpha_2+\phi)\sin(\alpha_1)\sin(\alpha_3) \\
& + l_3\cos(\alpha_3+\phi)\sin(\alpha_1)\sin(\alpha_2)))/(r_w^2\sin(\alpha_1)\sin(\alpha_2)\sin(\alpha_3)) \\
f_{1,4} = {} & (4k_{vw}\sin(\phi))/r_w - (k_{vw}\cos(\phi)\cot(\alpha_1))/r_w - (k_{vw}\cos(\phi)\cot(\alpha_2))/r_w \\
& - (k_{vw}\cos(\phi)\cot(\alpha_3))/r_w - (k_{vw}\cos(\phi)\cot(\alpha_4))/r_w
\end{aligned}
$$

$$
\begin{aligned}
f_{2,1} = {} & (\sin(2\alpha_1 + 2\phi)(k_{rw} + k_{vw}))/(r_w^2(\cos(2\alpha_1) - 1)) \\
& + (\sin(2\alpha_2 + 2\phi)(k_{rw} + k_{vw}))/(r_w^2(\cos(2\alpha_2) - 1)) \\
& + (\sin(2\alpha_3 + 2\phi)(k_{rw} + k_{vw}))/(r_w^2(\cos(2\alpha_3) - 1)) \\
& + (\sin(2\alpha_4 + 2\phi)(k_{rw} + k_{vw}))/(r_w^2(\cos(2\alpha_4) - 1)) \\
& \quad + (\cos(\phi)((k_{vr}\sin(\phi)\sin(\alpha_1))/(r_r - r_w) \\
& - (k_{vr}\sin(\alpha_1 + \phi)\cos(\alpha_1))/(r_w\sin(\alpha_1))))/(r_r\sin(\alpha_1)) \\
& \quad + (\cos(\phi)((k_{vr}\sin(\phi)\sin(\alpha_2))/(r_r - r_w) \\
& - (k_{vr}\sin(\alpha_2 + \phi)\cos(\alpha_2))/(r_w\sin(\alpha_2))))/(r_r\sin(\alpha_2)) \\
& \quad + (\cos(\phi)((k_{vr}\sin(\phi)\sin(\alpha_3))/(r_r - r_w) \\
& - (k_{vr}\sin(\alpha_3 + \phi)\cos(\alpha_3))/(r_w\sin(\alpha_3))))/(r_r\sin(\alpha_3)) \\
& \quad + (\cos(\phi)((k_{vr}\sin(\phi)\sin(\alpha_4))/(r_r - r_w) \\
& \quad - (k_{vr}\sin(\alpha_4 + \phi)\cos(\alpha_4))/(r_w\sin(\alpha_4))))/(r_r\sin(\alpha_4))
\end{aligned}
$$

$$
\begin{aligned}
f_{2,2} = {} & (\sin(\phi)((k_{vr}\sin(\phi)\sin(\alpha_1))/(r_r - r_w) \\
& + (k_{vr}\sin(\alpha_1 + \phi)(2\sin(\alpha_1/2)^2 - 1))/(r_w\sin(\alpha_1))))/(r_r\sin(\alpha_1)) \\
& \quad + (\sin(\phi)((k_{vr}\sin(\phi)\sin(\alpha_2))/(r_r - r_w) \\
& + (k_{vr}\sin(\alpha_2 + \phi)(2\sin(\alpha_2/2)^2 - 1))/(r_w\sin(\alpha_2))))/(r_r\sin(\alpha_2)) \\
& \quad + (\sin(\phi)((k_{vr}\sin(\phi)\sin(\alpha_3))/(r_r - r_w) \\
& + (k_{vr}\sin(\alpha_3 + \phi)(2\sin(\alpha_3/2)^2 - 1))/(r_w\sin(\alpha_3))))/(r_r\sin(\alpha_3)) \\
& \quad + (\sin(\phi)((k_{vr}\sin(\phi)\sin(\alpha_4))/(r_r - r_w) \\
& + (k_{vr}\sin(\alpha_4 + \phi)(2\sin(\alpha_4/2)^2 - 1))/(r_w\sin(\alpha_4))))/(r_r\sin(\alpha_4)) \\
& \quad - (\sin(\alpha_1 + \phi)^2(k_{rw} + k_{vw}))/(r_w^2\sin(\alpha_1)^2) \\
& \quad - (\sin(\alpha_2 + \phi)^2(k_{rw} + k_{vw}))/(r_w^2\sin(\alpha_2)^2) \\
& \quad - (\sin(\alpha_3 + \phi)^2(k_{rw} + k_{vw}))/(r_w^2\sin(\alpha_3)^2) \\
& \quad - (\sin(\alpha_4 + \phi)^2(k_{rw} + k_{vw}))/(r_w^2\sin(\alpha_4)^2)
\end{aligned}
$$

$$
\begin{aligned}
f_{2,3} = {} & -(l_4\sin(\alpha_4 + \phi)(k_{rw} + k_{vw}))/(r_w^2\sin(\alpha_4)) \\
& \quad - ((k_{rw} + k_{vw})(l_1\sin(\alpha_1 + \phi)\sin(\alpha_2)\sin(\alpha_3) \\
& \quad + l_2\sin(\alpha_2 + \phi)\sin(\alpha_1)\sin(\alpha_3) \\
& + l_3\sin(\alpha_3 + \phi)\sin(\alpha_1)\sin(\alpha_2)))/(r_w^2\sin(\alpha_1)\sin(\alpha_2)\sin(\alpha_3))
\end{aligned}
$$

$$
\begin{aligned}
f_{2,4} = & -(4k_{vw}\cos(\phi))/r_w - (k_{vw}\sin(\phi)\cot(\alpha_1))/r_w - (k_{vw}\sin(\phi)\cot(\alpha_2))/r_w \\
& - (k_{vw}\sin(\phi)\cot(\alpha_3))/r_w - (k_{vw}\sin(\phi)\cot(\alpha_4))/r_w \\
f_{3,1} = & -(l_1\cos(\alpha_1+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_1)) \\
& - (l_2\cos(\alpha_2+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_2)) \\
& - (l_3\cos(\alpha_3+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_3)) \\
& - (l_4\cos(\alpha_4+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_4)) \\
& - (k_{vr}l_1\cos(\phi)\cot(\alpha_1))/(r_r r_w) - (k_{vr}l_2\cos(\phi)\cot(\alpha_2))/(r_r r_w) \\
& - (k_{vr}l_3\cos(\phi)\cot(\alpha_3))/(r_r r_w) - (k_{vr}l_4\cos(\phi)\cot(\alpha_4))/(r_r r_w) \\
f_{3,2} = & -(l_1\sin(\alpha_1+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_1)) \\
& - (l_2\sin(\alpha_2+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_2)) \\
& - (l_3\sin(\alpha_3+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_3)) \\
& - (l_4\sin(\alpha_4+\phi)(k_{rw}+k_{vw}))/(r_w^2\sin(\alpha_4)) \\
& - (k_{vr}l_1\sin(\phi)\cot(\alpha_1))/(r_r r_w) - (k_{vr}l_2\sin(\phi)\cot(\alpha_2))/(r_r r_w) \\
& - (k_{vr}l_3\sin(\phi)\cot(\alpha_3))/(r_r r_w) - (k_{vr}l_4\sin(\phi)\cot(\alpha_4))/(r_r r_w) \\
f_{3,3} = & -((k_{rw}+k_{vw})(l_1^2+l_2^2+l_3^2+l_4^2))/r_w^2 \\
f_{3,4} = & -(k_{vw}(l_1+l_2+l_3+l_4))/r_w \\
f_{4,1} = & (4k_{vw}\sin(\phi))/r_w - (k_{vw}\cos(\phi)\cot(\alpha_1))/r_w - (k_{vw}\cos(\phi)\cot(\alpha_2))/r_w \\
& - (k_{vw}\cos(\phi)\cot(\alpha_3))/r_w - (k_{vw}\cos(\phi)\cot(\alpha_4))/r_w \\
f_{4,2} = & -(4k_{vw}\cos(\phi))/r_w - (k_{vw}\sin(\phi)\cot(\alpha_1))/r_w - (k_{vw}\sin(\phi)\cot(\alpha_2))/r_w \\
& - (k_{vw}\sin(\phi)\cot(\alpha_3))/r_w - (k_{vw}\sin(\phi)\cot(\alpha_4))/r_w \\
f_{4,3} = & -(k_{vw}(l_1+l_2+l_3+l_4))/r_w \\
f_{4,4} = & -4k_{vw}
\end{aligned}
$$

$$
B_p(p) = \begin{bmatrix}
\frac{-\cos(\alpha_1+\phi}{r_w\sin(\alpha_1)} & \frac{-\cos(\alpha_2+\phi}{r_w\sin(\alpha_2)} & \frac{-\cos(\alpha_3+\phi}{r_w\sin(\alpha_3)} & \frac{-\cos(\alpha_4+\phi}{r_w\sin(\alpha_4)} \\
\frac{-\sin(\alpha_1+\phi}{r_w\sin(\alpha_1)} & \frac{-\sin(\alpha_2+\phi}{r_w\sin(\alpha_2)} & \frac{-\sin(\alpha_3+\phi}{r_w\sin(\alpha_3)} & \frac{-\sin(\alpha_4+\phi}{r_w\sin(\alpha_4)} \\
\frac{-l_1}{r_w} & \frac{-l_2}{r_w} & \frac{-l_3}{r_w} & \frac{-l_4}{r_w} \\
-1 & -1 & -1 & -1
\end{bmatrix} \tag{A.2.5}
$$

A.3 Matrices for CMD Dynamics Model in (p, v)

$$M_v(p) = \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & 0 \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & 0 \\ 0 & m_{4,2} & 0 & m_{4,4} \end{bmatrix} \tag{A.3.1}$$

$$\begin{aligned}
m_{1,1} &= m_p + 4m_w - (4I_{wx})/r_w^2 + I_{wx}/(r_w^2 \sin(\alpha_1)^2) + I_{wx}/(r_w^2 \sin(\alpha_2)^2) \\
&\quad + I_{wx}/(r_w^2 \sin(\alpha_3)^2) + I_{wx}/(r_w^2 \sin(\alpha_4)^2) \\
m_{1,2} &= (I_{wx}(\cot(\alpha_1) + \cot(\alpha_2) + \cot(\alpha_3) + \cot(\alpha_4)))/r_w^2 \\
m_{1,3} &= (I_{wx}(l_1 \cot(\alpha_1) + l_2 \cot(\alpha_2) + l_3 \cot(\alpha_3) + l_4 \cot(\alpha_4)))/r_w^2 + h_p m_p \sin(\theta_p) \\
m_{2,1} &= (I_{wx}(\cot(\alpha_1) + \cot(\alpha_2) + \cot(\alpha_3) + \cot(\alpha_4)))/r_w^2 \\
m_{2,2} &= m_p + 4m_w + (4I_{wx})/r_w^2 \\
m_{2,3} &= ((m_w r_w^2 + I_{wx})(l_1 + l_2 + l_3 + l_4))/r_w^2 \\
m_{2,4} &= -h_p m_p \cos(\theta_p) \\
m_{3,1} &= (I_{wx}(l_1 \cot(\alpha_1) + l_2 \cot(\alpha_2) + l_3 \cot(\alpha_3) + l_4 \cot(\alpha_4)))/r_w^2 + h_p m_p \sin(\theta_p) \\
m_{3,2} &= ((m_w r_w^2 + I_{wx})(l_1 + l_2 + l_3 + l_4))/r_w^2 \\
m_{3,3} &= 4I_{wyz} + (I_{wx} l_1^2 + I_{wx} l_2^2 + I_{wx} l_3^2 + I_{wx} l_4^2)/r_w^2 + I_{pby} \sin(\theta_p)^2 + l_1^2 m_w + l_2^2 m_w \\
&\quad + l_3^2 m_w + l_4^2 m_w - I_{pbz}(\sin(\theta_p)^2 - 1) + h_p^2 m_p \sin(\theta_p)^2 \\
m_{4,2} &= -h_p m_p \cos(\theta_p) \\
m_{4,4} &= m_p h_p^2 + I_{pbx}
\end{aligned}$$

$$C_v(p, v) = \begin{bmatrix} 0 & c_{1,2} & c_{1,3} & c_{1,4} \\ c_{2,1} & 0 & c_{2,3} & c_{2,4} \\ c_{3,1} & c_{3,2} & c_{3,3} & c_{3,4} \\ c_{4,1} & 0 & c_{4,3} & 0 \end{bmatrix} \tag{A.3.2}$$

$$\begin{aligned}
c_{1,2} &= -\dot{\phi}(m_p + 4m_w) \\
c_{1,3} &= h_p m_p \cos(\theta_p)\dot{\theta}_p - l_2 m_w \dot{\phi} - l_3 m_w \dot{\phi} - l_4 m_w \dot{\phi} - l_1 m_w \dot{\phi}
\end{aligned}$$

$$
\begin{aligned}
c_{1,4} &= h_p m_p \cos(\theta_p)\dot{\phi} \\
c_{2,1} &= \dot{\phi}(m_p + 4m_w) \\
c_{2,3} &= h_p m_p \sin(\theta_p)\dot{\phi} \\
c_{2,4} &= h_p m_p \sin(\theta_p)\dot{\theta}_p \\
c_{3,1} &= m_w \dot{\phi}(l_1 + l_2 + l_3 + l_4) \\
c_{3,2} &= -h_p m_p \sin(\theta_p)\dot{\phi} \\
c_{3,3} &= (\sin(2\theta_p)\dot{\theta}_p(m_p h_p^2 + I_{pby} - I_{pbz}))/2 \\
c_{3,4} &= (\sin(2\theta_p)\dot{\phi}(m_p h_p^2 + I_{pby} - I_{pbz}))/2 \\
c_{4,1} &= -h_p m_p \cos(\theta_p)\dot{\phi} \\
c_{4,3} &= -(\sin(2\theta_p)\dot{\phi}(m_p h_p^2 + I_{pby} - I_{pbz}))/2
\end{aligned}
$$

$$
G_v(p) = \begin{bmatrix} 0_{3\times 1} \\ -g h_p m_p \sin(\theta_p) \end{bmatrix} \tag{A.3.3}
$$

$$
F_v = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & f_{1,4} \\ f_{2,1} & f_{2,2} & f_{2,3} & f_{2,4} \\ f_{3,1} & f_{3,2} & f_{3,3} & f_{3,4} \\ f_{4,1} & f_{4,2} & f_{4,3} & f_{4,4} \end{bmatrix} \tag{A.3.4}
$$

$$
\begin{aligned}
f_{1,1} = (&k_{vr} r_w^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 - k_{vw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_3)^2 - k_{vw} r_r^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&- k_{vw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 + k_{vr} r_w^2 \sin(\alpha_1)^2 \sin(\alpha_3)^2 \\
&+ k_{vr} r_w^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 - k_{rw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \\
&- k_{rw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_3)^2 - k_{rw} r_r^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&- k_{vr} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 - k_{vr} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_3)^2 \\
&- k_{vr} r_r r_w \sin(\alpha_2)^2 \sin(\alpha_3)^2 + k_{vw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 \\
&+ k_{vw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_3)^2 + k_{vw} r_r r_w \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&+ 4k_{rw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 + 4k_{vw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&+ k_{rw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 + k_{rw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_3)^2 \\
&+ k_{rw} r_r r_w \sin(\alpha_2)^2 \sin(\alpha_3)^2 - 4k_{rw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&+ 4k_{vr} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&- 4k_{vw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2) \\
&/(r_r r_w^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 (r_r - r_w)) \\
&- (k_{rw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 + k_{vw} r_r^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&- k_{vr} r_w^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 - k_{rw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&+ k_{vr} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \\
&- k_{vw} r_r r_w \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2) \\
&/(r_r r_w^2 \sin(\alpha_1)^2 \sin(\alpha_2)^2 \sin(\alpha_3)^2 \sin(\alpha_4)^2 (r_r - r_w))
\end{aligned}
$$

$$
\begin{aligned}
f_{1,2} &= -((k_{rw} + k_{vw})(\cot(\alpha_1) + \cot(\alpha_2) + \cot(\alpha_3) + \cot(\alpha_4)))/r_w^2 \\
f_{1,3} &= -((k_{rw} + k_{vw})(l_1 \cot(\alpha_1) + l_2 \cot(\alpha_2) + l_3 \cot(\alpha_3) + l_4 \cot(\alpha_4)))/r_w^2 \\
f_{1,4} &= -(k_{vw}(\cot(\alpha_1) + \cot(\alpha_2) + \cot(\alpha_3) + \cot(\alpha_4)))/r_w \\
f_{2,1} &= -((k_{rw} r_r + k_{vw} r_r + k_{vr} r_w)(\cot(\alpha_1) + \cot(\alpha_2) + \cot(\alpha_3) + \cot(\alpha_4)))/(r_r r_w^2) \\
f_{2,2} &= -(4(k_{rw} + k_{vw}))/r_w^2 \\
f_{2,3} &= -((k_{rw} + k_{vw})(l_1 + l_2 + l_3 + l_4))/r_w^2 \\
f_{2,4} &= -(4k_{vw})/r_w \\
f_{3,1} &= -((k_{rw} r_r + k_{vw} r_r + k_{vr} r_w)(l_1 \cot(\alpha_1) + l_2 \cot(\alpha_2) + l_3 \cot(\alpha_3) \\
&\qquad + l_4 \cot(\alpha_4)))/(r_r r_w^2) \\
f_{3,2} &= -((k_{rw} + k_{vw})(l_1 + l_2 + l_3 + l_4))/r_w^2
\end{aligned}
$$

$$
\begin{aligned}
f_{3,3} &= -((k_{rw}+k_{vw})(l_1^2+l_2^2+l_3^2+l_4^2))/r_w^2 \\
f_{3,4} &= -(k_{vw}(l_1+l_2+l_3+l_4))/r_w \\
f_{4,1} &= -(k_{vw}(\cot(\alpha_1)+\cot(\alpha_2)+\cot(\alpha_3)+\cot(\alpha_4)))/r_w \\
f_{4,2} &= -(4k_{vw})/r_w \\
f_{4,3} &= -(k_{vw}(l_1+l_2+l_3+l_4))/r_w \\
f_{4,4} &= -4k_{vw}
\end{aligned}
$$

$$
B_v = \begin{bmatrix}
-\frac{\cot(\alpha_1)}{r_w} & -\frac{\cot(\alpha_2)}{r_w} & -\frac{\cot(\alpha_3)}{r_w} & -\frac{\cot(\alpha_4)}{r_w} \\
-\frac{1}{r_w} & -\frac{1}{r_w} & -\frac{1}{r_w} & -\frac{1}{r_w} \\
-\frac{l_1}{r_w} & -\frac{l_2}{r_w} & -\frac{l_3}{r_w} & -\frac{l_4}{r_w} \\
-1 & -1 & -1 & -1
\end{bmatrix} \tag{A.3.5}
$$

Appendix B

MATLAB Model Derivation and Analysis Code

```
function [ out ] = model_derivation( generateBlocks )

syms t real;
syms m_p m_w I_pbx I_pby I_pbz I_wx I_wyz h_p l_1 l_2 ...
    l_3 l_4 r_w r_r g alpha_1 alpha_2 alpha_3 alpha_4 ...
    k_vr k_vw K_rw real;
syms x(t) y(t) phi(t) theta_p(t) theta_1(t) theta_2(t) ...
    theta_3(t) theta_4(t) omega_1(t) omega_2(t) omega_3(...
    t) omega_4(t);
syms dx(t) dy(t) dtheta_p(t) dphi(t) dtheta_1(t) ...
    dtheta_2(t) dtheta_3(t) dtheta_4(t) domega_1(t) ...
    domega_2(t) domega_3(t) domega_4(t);
syms vx(t) vy(t);
syms ddx ddy ddphi ddtheta_p ddtheta_1 ddtheta_2 ...
    ddtheta_3 ddtheta_4 ddomega_1 ddomega_2 ddomega_3 ...
    ddomega_4 real;
syms dvx dvy real;
syms tau_1 tau_2 tau_3 tau_4 real;

assumeAlso(x(t), 'real');
assumeAlso(y(t), 'real');
assumeAlso(phi(t), 'real');
```

```
15 assumeAlso(theta_p(t), 'real');
16 assumeAlso(theta_1(t), 'real');
17 assumeAlso(theta_2(t), 'real');
18 assumeAlso(theta_3(t), 'real');
19 assumeAlso(theta_4(t), 'real');
20 assumeAlso(omega_1(t), 'real');
21 assumeAlso(omega_2(t), 'real');
22 assumeAlso(omega_3(t), 'real');
23 assumeAlso(omega_4(t), 'real');
24 assumeAlso(dx(t), 'real');
25 assumeAlso(dy(t), 'real');
26 assumeAlso(dphi(t), 'real');
27 assumeAlso(dtheta_p(t), 'real');
28 assumeAlso(dtheta_1(t), 'real');
29 assumeAlso(dtheta_2(t), 'real');
30 assumeAlso(dtheta_3(t), 'real');
31 assumeAlso(dtheta_4(t), 'real');
32 assumeAlso(domega_1(t), 'real');
33 assumeAlso(domega_2(t), 'real');
34 assumeAlso(domega_3(t), 'real');
35 assumeAlso(domega_4(t), 'real');
36 assumeAlso(vx(t), 'real');
37 assumeAlso(vy(t), 'real');
38 assumeAlso(m_p > 0);
39 assumeAlso(m_w > 0);
40 assumeAlso(I_pbx > 0);
41 assumeAlso(I_pby > 0);
42 assumeAlso(I_pbz > 0);
43 assumeAlso(I_wx > 0);
44 assumeAlso(I_wyz > 0);
45 assumeAlso(r_w > 0);
46 assumeAlso((alpha_1 > -pi/2 & alpha_1 < 0) | (alpha_1 <...
       pi/2 & alpha_1 > 0));
47 assumeAlso((alpha_2 > -pi/2 & alpha_2 < 0) | (alpha_2 <...
```

```
    pi/2 & alpha_2 > 0));
assumeAlso((alpha_3 > -pi/2 & alpha_3 < 0) | (alpha_3 <...
    pi/2 & alpha_3 > 0));
assumeAlso((alpha_4 > -pi/2 & alpha_4 < 0) | (alpha_4 <...
    pi/2 & alpha_4 > 0));

% Define number of wheels
nw = 4;

if nw == 3
theta_i = {theta_1 theta_2 theta_3};
dtheta_i = {dtheta_1 dtheta_2 dtheta_3};
ddtheta_i = {ddtheta_1 ddtheta_2 ddtheta_3};
omega_i = {omega_1 omega_2 omega_3};
domega_i = {domega_1 domega_2 domega_3};
ddomega_i = {ddomega_1 ddomega_2 ddomega_3};
l_i = [l_1 l_2 l_3];
alpha_i = [alpha_1 alpha_2 alpha_3];
tau_i = [tau_1 tau_2 tau_3]';
elseif nw == 4
theta_i = {theta_1 theta_2 theta_3 theta_4};
dtheta_i = {dtheta_1 dtheta_2 dtheta_3 dtheta_4};
ddtheta_i = {ddtheta_1 ddtheta_2 ddtheta_3 ddtheta_4};
omega_i = {omega_1 omega_2 omega_3 omega_4};
domega_i = {domega_1 domega_2 domega_3 domega_4};
domega_i = {ddomega_1 ddomega_2 ddomega_3 ddomega_4};
l_i = [l_1 l_2 l_3 l_4];
alpha_i = [alpha_1 alpha_2 alpha_3 alpha_4];
tau_i = [tau_1 tau_2 tau_3 tau_4]';
end
I_w = diag([I_wx I_wyz I_wyz]);
I_p = diag([I_pbx I_pby I_pbz]);


```

```
% Full generalised coordinates
if nw == 3
q = {x; y; phi; theta_p; theta_1; theta_2; theta_3; ...
    omega_1; omega_2; omega_3;};
dq = {dx; dy; dphi; dtheta_p; dtheta_1; dtheta_2; ...
    dtheta_3; domega_1; domega_2; domega_3};
ddq = {ddx; ddy; ddphi; ddtheta_p; ddtheta_1; ddtheta_2...
    ; ddtheta_3; ddomega_1; ddomega_2; ddomega_3};
elseif nw == 4
q = {x; y; phi; theta_p; theta_1; theta_2; theta_3; ...
    theta_4; omega_1; omega_2; omega_3; omega_4};
dq = {dx; dy; dphi; dtheta_p; dtheta_1; dtheta_2; ...
    dtheta_3; dtheta_4; domega_1; domega_2; domega_3; ...
    domega_4};
ddq = {ddx; ddy; ddphi; ddtheta_p; ddtheta_1; ddtheta_2...
    ; ddtheta_3; ddtheta_4; ddomega_1; ddomega_2; ...
    ddomega_3; ddomega_4};
end

% Reduced generalised coordinates
p = {x; y; phi; theta_p};
dp = {dx; dy; dphi; dtheta_p};
ddp = [ddx; ddy; ddphi; ddtheta_p];

% pv generalised coordinates
v = {vx; vy; dphi; dtheta_p};
dv = [dvx; dvy; ddphi; ddtheta_p];

% input
u = [tau_i];


% It seems either these rotation matrices or their ...
    transpose can be
```

```
104 % correct, provided the correct sign is applied to l_i ...
      in the constraint
105 % matrix. Kim and others use this convention, ...
      autonomous mobile robots uses
106 % another.
107
108 % Now using convention of x is right, y is forward, z ...
      is up, all rotations
109 % are by right hand rule.
110
111 %% Define rotation matrices
112 % Rotation from intertial frame E to body attached ...
      frame B
113 R_EB = [cos(phi) -sin(phi) 0;
114 sin(phi) cos(phi) 0;
115 0 0 1];
116 sel = [eye(2); 0 0];
117 R_EB_2d = sel'*R_EB*sel;
118
119 % Rotation from body attached frame B to pendulum rigid...
      body frame P
120 R_BP = [1 0 0
121 0 cos(theta_p) -sin(theta_p)
122 0 sin(theta_p) cos(theta_p)];
123
124 % Rotation from body to wheel frame
125 for i=1:nw
126 R_BW{i} = [1 0 0
127 0 cos(theta_i{i}) -sin(theta_i{i})
128 0 sin(theta_i{i}) cos(theta_i{i})];
129 end
130
131 % Rotation from body to non-rotating roller frame (...
      acceptable simplification if roller is massless)
```

```
132 for i=1:nw
133 R_BR{i} = [cos(alpha_i(i)) -sin(alpha_i(i)) 0;
134 sin(alpha_i(i)) cos(alpha_i(i)) 0;
135 0 0 1];
136 end
137
138 %% Velocities derivation
139 % Define body angular velocity
140 omega_b = [0; 0; dphi];
141
142 % Define pendulum angular velocity
143 omega_p = R_BP.'*omega_b + [dtheta_p; 0; 0];
144
145 % Define wheel angular velocity
146 for i=1:nw
147 % Both of these expressions give identical results
148 % omega_w{i} = R_BW{i}'*omega_b + [dtheta_i{i}; 0; 0];
149 omega_w{i} = omega_b + [dtheta_i{i}; 0; 0];
150 end
151
152 % Define body frame velocities in terms of q
153 v_b = R_EB.'*[dx; dy; 0];
154
155 % Define pendulum translational velocities
156 v_p = R_BP.'*v_b + cross(omega_p, [0; 0; h_p]);
157
158 % Define wheel translational velocities
159 for i=1:nw
160 % Both of these expressions give identical results
161 % v_w{i} = R_BW{i}'*v_b + cross(omega_w{i}, [l_i(i); 0;...
      0]);
162 v_w{i} = v_b + cross(omega_w{i}, [l_i(i); 0; 0]);
163 end
164
```

```
165 %% Derivation of constraints, inverse kinematics, and M...
      (q)
166 for i=1:nw
167 % Body frame velocity at center of wheel (W)
168 v_Wb{i} = v_b + [0; dphi*l_i(i); 0];
169
170 % Body frame velocity of roller contact point (C)
171 v_Cb{i} = v_Wb{i} + [0; -r_w*-dtheta_i{i}; 0];
172
173 % Body frame roller rotation axis u_p and transverse ...
      axis u_t
174 u_p{i} = R_BR{i}*[1;0;0];
175 u_t{i} = R_BR{i}*[0;1;0];
176
177 % Contraints
178 constraintNoSlip(i,1) = dot(formula(v_Cb{i}),u_p{i});
179 constraintRolling(i,1) = dot(formula(v_Cb{i}),u_t{i}) -...
       -r_r*domega_i{i};
180 end
181
182 % Solve no slip constraint for theta_i
183 for i=1:nw
184 theta_iSolved(i,1) = solve(stripTimeDependence(...
      constraintNoSlip(i)), stripTimeDependence(formula(...
      dtheta_i{i})));
185 end
186
187 % Pfaffian constraint matrix
188 A = subs(equationsToMatrix(stripTimeDependence([...
      constraintNoSlip; constraintRolling]), ...
      stripTimeDependence(formula([dx; dy; dphi; dtheta_p;...
       dtheta_i'; domega_i'])))),stripTimeDependence(...
      formula(phi)),formula(phi));
189
```

```
190
191 %% Energies
192 % Translational kinetic energy
193 K_trans = 0.5*m_p*(v_p.'*v_p);
194 for i=1:nw
195 K_trans = K_trans + 0.5*m_w*(v_w{i}.'*v_w{i});
196 end
197
198 % Rotational kinetic energy
199 K_rot = 0.5*omega_p.'*I_p*omega_p;
200 for i=1:nw
201 K_rot = K_rot + 0.5*omega_w{i}.'*I_w*omega_w{i};
202 end
203
204 K_trans = simplify(K_trans);
205 K_rot = simplify(K_rot, 5); % Doesnt seem to simplify ...
      cos(x)^2+sin(x)^2=1 with less than 5 simplification ...
      steps
206
207 % Potential Energy
208 U = m_p*g*h_p*cos(theta_p);
209
210 % Total Energy
211 L = K_trans + K_rot - U;
212
213
214 %% Euler-Lagrange Derivation
215 % Generalised forces
216 Q_tau = [
217 zeros(3,1);
218 -sum(tau_i);
219 tau_i;
220 zeros(nw,1);
221 ];
```

```
222
223 if nw == 3
224 Q_fric = [
225 [eye(2) zeros(2,1)]*R_EB*k_vr*[1/(-r_w+r_r);0;0]*dot...
       ([0;1;0],formula(R_BR{1}*[domega_1;0;0] + R_BR{2}*[...
       domega_2;0;0] + R_BR{3}*[domega_3;0;0]));
226 0;
227 sum(k_vw*(dtheta_i - dtheta_p));
228 -K_rw*dtheta_1 + k_vw*(dtheta_p - dtheta_1) + k_vr*dot...
       ([1;0;0],formula(R_BR{1}*[domega_1; 0; 0]));
229 -K_rw*dtheta_2 + k_vw*(dtheta_p - dtheta_2) + k_vr*dot...
       ([1;0;0],formula(R_BR{2}*[domega_2; 0; 0]));
230 -K_rw*dtheta_3 + k_vw*(dtheta_p - dtheta_3) + k_vr*dot...
       ([1;0;0],formula(R_BR{3}*[domega_3; 0; 0]));
231 -k_vr*domega_i';
232 ];
233 elseif nw == 4
234 Q_fric = [
235 [eye(2) zeros(2,1)]*R_EB*k_vr*[1/(-r_w+r_r);0;0]*dot...
       ([0;1;0],formula(R_BR{1}*[domega_1;0;0] + R_BR{2}*[...
       domega_2;0;0] + R_BR{3}*[domega_3;0;0] + R_BR{4}*[...
       domega_4;0;0]));
236 0;
237 sum(k_vw*(dtheta_i - dtheta_p));
238 -K_rw*dtheta_1 + k_vw*(dtheta_p - dtheta_1) + k_vr*dot...
       ([1;0;0],formula(R_BR{1}*[domega_1; 0; 0]));
239 -K_rw*dtheta_2 + k_vw*(dtheta_p - dtheta_2) + k_vr*dot...
       ([1;0;0],formula(R_BR{2}*[domega_2; 0; 0]));
240 -K_rw*dtheta_3 + k_vw*(dtheta_p - dtheta_3) + k_vr*dot...
       ([1;0;0],formula(R_BR{3}*[domega_3; 0; 0]));
241 -K_rw*dtheta_4 + k_vw*(dtheta_p - dtheta_4) + k_vr*dot...
       ([1;0;0],formula(R_BR{4}*[domega_4; 0; 0]));
242 -k_vr*domega_i';
243 ];
```

```
244 end
245
246 % Compute Lagrangian
247 lambda = sym('lambda', [size(A,1) 1], 'real');
248 model = diff(functionalDerivative(L,dq),t) - ...
      functionalDerivative(L,q) == A.'*lambda + Q_tau + ...
      Q_fric;
249
250 % Substitute diff(q,t) for dq and diff(dq,t) for ddq
251 model = subs(model, diff(q,t), dq);
252 model = subs(model, diff(dq,t), ddq);
253
254 %% Fitting model to standard form M(q)ddq + C(q,dq)dq +...
       G(q) == A'lambda + Bu + Fdq
255 % Rearrange LHS into terms of inertia and coriolis ...
      matrix
256 out.qdq.Mq = equationsToMatrix(lhs(model), ddq);
257
258 % Test for symmetry
259 if ¬isequal(simplify(out.qdq.Mq-out.qdq.Mq.'), sym(...
      zeros(numel(q))))
260 warning('Mq is asymmetric')
261 end
262
263 % Test for positive SEMIdefiniteness of Mq - expected ...
      as rollers are massless & inertialess
264 if all(isAlways(eig(simplify(substituteParameters(...
      out.qdq.Mq)))>-eps)) == false
265 warning('Mq is not positive semidefinite');
266 end
267
268 % Derive C(q,dq) matrix using Christoffel symbols of ...
      Mq. This operation is
269 % very slow, but seems to just be due to the number of ...
```

```
    calls to
% functionalDerivative, i.e. no way no improve. Running...
     this as a single
% thread profiled at 554s, running with the parfor at ...
    212s.
for i = 1:size(out.qdq.Mq,1)
parfor j = 1:size(out.qdq.Mq,1)
Cqdq(i,j) = sym(0); % Required to set type of cell
for k = 1:size(out.qdq.Mq,1)
Cqdq(i,j) = Cqdq(i,j) + 0.5*( ...
functionalDerivative(out.qdq.Mq(i,j),q{k}) ...
+ functionalDerivative(out.qdq.Mq(i,k),q{j}) ...
- functionalDerivative(out.qdq.Mq(j,k),q{i})...
) * dq{k};
end
end
end
out.qdq.Cqdq = Cqdq;
clear Cqdq;

Test for skew symmetry of dMq - 2Cqdq
dMq_minus_2Cqdq = subs(diff(out.qdq.Mq,t) - 2*...
    out.qdq.Cqdq, diff(q,t), dq);
if ¬isequal(simplify(dMq_minus_2Cqdq + dMq_minus_2Cqdq....
    '), sym(zeros(numel(q))))
warning('dMq-2Cqdq is not skew symmetric');
end

% Derive input matrix B
out.qdq.B = equationsToMatrix(rhs(model), u);

if any(Q_fric ≠ 0)
% Derive friction matrix Fq
temp = sym('temp',[numel(dq) 1]);
```

```
299 out.qdq.F = equationsToMatrix(subs(rhs(model) - A.'*...
        lambda, dq, temp), temp);
300 else
301 out.qdq.F = sym(zeros(numel(q)));
302 end
303
304 % Derive gravity matrix Gq
305 out.qdq.Gq = formula(simplify(lhs(model) - out.qdq.Mq*...
        ddq - out.qdq.Cqdq*dq));
306
307 % Gq should contain a single term
308 if nnz(formula(out.qdq.Gq)) > 1
309 warning('Mq and Cqdq do not fully describe passive ...
        unconstrained Lagrangian');
310 end
311
312 % Package matrices and model
313 out.qdq.model = out.qdq.Mq*ddq + out.qdq.Cqdq*dq + ...
        out.qdq.Gq == out.qdq.F*dq + out.qdq.B*tau_i;
314
315
316 %% Convert model from generalised coordinates q to p
317 % Calculate null matrix such that AH=0
318 % Calculating the null matrix for a reordering of q ...
        into q =
319 % [theta_i x y phi theta_p] then swapping back to the ...
        original
320 % ordering yields an identity mapping for x y phi ...
        theta_p
321 out.pdp.H = null([A(:,(numel(p)+1):end) A(:,1:numel(p))...
        ]);
322 out.pdp.H = [out.pdp.H((end-(numel(p)-1)):end,:); ...
        out.pdp.H(1:numel(q)-numel(p),:)];
323
```

```
324 if ¬isequal(simplify(A*out.pdp.H), sym(zeros(numel(q)-...
       numel(p), numel(p))))
325 error('H is not a basis for the nullspace of A');
326 end
327
328 % Redefine Mp
329 out.pdp.Mp = simplify(out.pdp.H.'*out.qdq.Mq*out.pdp.H)...
       ;
330
331 Retest for symmetry
332 if ¬isequal(simplify(out.pdp.Mp-out.pdp.Mp.'), sym(...
       zeros(numel(p))))
333 warning('Mp is asymmetric')
334 end
335
336 % Retest for positive definiteness
337 if all(isAlways(eig(simplify(substituteParameters(...
       out.pdp.Mp)))>eps)) == false
338 warning('Mp is not positive definite');
339 end
340
341 % Redefine Cqdq
342 out.pdp.Cpdp = simplify(out.pdp.H.'*out.qdq.Cqdq*...
       out.pdp.H + out.pdp.H.'*out.qdq.Mq*subs(diff(...
       out.pdp.H,t), diff(p,t), dp));
343
344 % Retest for skew symmetry of dMq - 2Cqdq
345 dMp_minus_2Cpdp = subs(diff(out.pdp.Mp,t) - 2*...
       out.pdp.Cpdp, diff(p,t), dp);
346 if ¬isequal(simplify(dMp_minus_2Cpdp + dMp_minus_2Cpdp....
       '), sym(zeros(numel(p))))
347 warning('dMp-2Cpdp is not skew symmetric');
348 end
349
```

```
350 % Redefine gravity matrix
351 out.pdp.Gp = out.pdp.H.' * out.qdq.Gq;
352
353 % Redefine input matrix
354 out.pdp.Bp = simplify(out.pdp.H.' * out.qdq.B);
355
356 % Redefine friction matrix
357 out.pdp.Fp = simplify(out.pdp.H.' * out.qdq.F * ...
       out.pdp.H);
358
359 % % Build model
360 % out.pdp.model = out.pdp.Mp*ddp + out.pdp.Cpdp*dp + ...
       out.pdp.Gp == out.pdp.Fp*dp + out.pdp.Bp*tau_i;
361 % % Currently runs out of memory when inverting Mp ...
       without substituting parameters
362 % out.pdp.dynamics = inv(substituteParameters(...
       out.pdp.Mp))*(out.pdp.Fp*dp + out.pdp.Bp*tau_i - ...
       out.pdp.Cpdp*dp - out.pdp.Gp);
363 % out.pdp.dpddp = [dp; out.pdp.dynamics];
364
365 % Solve no slip constraint for theta_i, and rolling ...
       constraint for omega_i
366 temp = sym('temp', [4 1]);
367 for i=1:nw
368 theta_iSolved(i,1) = solve(subs(constraintNoSlip(i), ...
       dtheta_i(i), temp(i)), temp(i));
369 omega_iSolved(i,1) = solve(subs(constraintRolling(i), ...
       domega_i(i), temp(i)), temp(i));
370 end
371 out.pdp.inverseKinematicsM = equationsToMatrix(subs(...
       theta_iSolved, dp, temp), temp);
372 out.pdp.inverseKinematicsIncRollersM = ...
       equationsToMatrix(subs([theta_iSolved; omega_iSolved...
       ], dp, temp), temp);
```

```
373
374
375 % Calculate generalised momentums conjugate to external...
       and shape variables
376 out.pdp.p_x = out.pdp.Mp(1:2,1:2)*dp(1:2) + out.pdp.Mp...
      (1:2,3:4)*dp(3:4);
377 out.pdp.p_s = out.pdp.Mp(3:4,1:2)*dp(1:2) + out.pdp.Mp...
      (3:4,3:4)*dp(3:4);
378
379 % Calculate normalised momentums
380 out.pdp.pi_x = inv(out.pdp.Mp(1:2,1:2)) * out.pdp.p_x;
381 out.pdp.pi_s = inv(out.pdp.Mp(3:4,1:2)) * out.pdp.p_s;
382
383 % Derivation method 2 for comparison
384 out.pdp.pi_x_2 = dp(1:2) + inv(out.pdp.Mp(1:2,1:2))*...
      out.pdp.Mp(1:2,3:4)*dp(3:4);
385 out.pdp.pi_s_2 = dp(1:2) + inv(out.pdp.Mp(3:4,1:2))*...
      out.pdp.Mp(3:4,3:4)*dp(3:4);
386
387 %% Convert model from generalised coordinates (q,dq,ddq...
      ) to (p,v,dv)
388 % Calculate null matrix such that AH=0
389 % Calculating the null matrix for a reordering of q ...
      into q =
390 % [theta_i x y phi theta_p] then swapping back to the ...
      original
391 % ordering yields an identity mapping for x y phi ...
      theta_p
392 out.pv.H = null([A(:,5:end) A(:,1:4)]);
393 out.pv.H = [out.pv.H(end-3:end,:); out.pv.H(1:numel(q)-...
      numel(p),:)];
394
395 % Rotate dx dy to vx vy
396 out.pv.Rv = blkdiag(formula(R_EB_2d),sym(eye(2)));
```

```
397 out.pv.H = out.pv.H*out.pv.Rv;
398
399 if ¬isequal(simplify(A*out.pv.H), sym(zeros(numel(q)-...
       numel(p), numel(p))))
400 error('H is not a basis for the nullspace of A');
401 end
402
403 % Redefine Mp
404 out.pv.Mp = simplify(out.pv.H.'*out.qdq.Mq*out.pv.H);
405
406 % Retest for symmetry
407 if ¬isequal(simplify(out.pv.Mp-out.pv.Mp.'), sym(zeros(...
       numel(p))))
408 warning('Mp is asymmetric')
409 end
410
411 % Retest for positive definiteness
412 if all(isAlways(eig(simplify(substituteParameters(...
       out.pv.Mp)))>eps)) == false
413 warning('Mp is not positive definite');
414 end
415
416 % Redefine Cqv
417 out.pv.Cpv = simplify(out.pv.H.'*out.qdq.Cqdq*out.pv.H ...
       + out.pv.H.'*out.qdq.Mq*subs(diff(out.pv.H,t), diff(...
       p,t), out.pv.Rv*v));
418
419 % Retest for skew symmetry of dMp - 2Cpv
420 dMp_minus_2Cpv = subs(diff(out.pv.Mp,t) - 2*out.pv.Cpv,...
        diff(p,t), out.pv.Rv*v);
421 if ¬isequal(simplify(dMp_minus_2Cpv + dMp_minus_2Cpv.')...
       , sym(zeros(numel(p))))
422 warning('dMp-2Cpv is not skew symmetric');
423 end
```

```
424
425 % Redefine gravity matrix
426 out.pv.Gp = out.pv.H.'*out.qdq.Gq;
427
428 % Redefine input matrix
429 out.pv.B = simplify(out.pv.H.'*out.qdq.B);
430
431 % Calculate simplified input u
432 out.pv.Bu = colspace(out.pv.B);
433 out.pv.Htau = pinv(out.pv.Bu)*out.pv.B;
434 out.pv.Htausubs = double(substituteParameters(...
      out.pv.Htau));
435
436 % Redefine friction matrix
437 out.pv.F = simplify(out.pv.H.'*out.qdq.F*out.pv.H);
438
439 % Redefine model
440 out.pv.model = out.pv.Mp*dv + out.pv.Cpv*v + out.pv.Gp ...
      == out.pv.F*v + out.pv.B*tau_i;
441 out.pv.dynamics = inv(out.pv.Mp)*(out.pv.F*v + out.pv.B...
      *tau_i - out.pv.Cpv*v - out.pv.Gp);
442 out.pv.inverseDynamics = pinv(out.pv.B)*(out.pv.Mp*dv +...
       out.pv.Cpv*v + out.pv.Gp - out.pv.F*v);
443 out.pv.dpdv = [out.pv.Rv*v; out.pv.dynamics];
444
445 % Redefine inverse kinematics in terms of v
446 out.pv.inverseKinematicsM = simplify(...
      out.pdp.inverseKinematicsM * out.pv.Rv);
447 out.pv.inverseKinematicsIncRollersM = simplify(...
      out.pdp.inverseKinematicsIncRollersM * out.pv.Rv);
448 out.pv.inverseKinematics = out.pv.inverseKinematicsM*v;
449 out.pv.inverseKinematicsIncRollers = ...
      out.pv.inverseKinematicsIncRollersM*v;
450
```

```
451 % Calculate generalised momentums conjugate to external...
        and shape variables
452 out.pv.p_x = out.pv.Mp(1:3,1:3)*v(1:3) + out.pv.Mp...
      (1:3,4)*v(4);
453 out.pv.p_s = out.pv.Mp(4,1:3)*v(1:3) + out.pv.Mp(4,4)*v...
      (4);
454
455 % Calculate normalised momentums
456 out.pv.pi_x = inv(out.pv.Mp(1:3,1:3)) * out.pv.p_x;
457 out.pv.pi_s = pinv(out.pv.Mp(4,1:3)) * out.pv.p_s;
458
459 % Derivation method 2 for comparison
460 out.pv.pi_x_2 = v(1:3) + inv(out.pv.Mp(1:3,1:3))*...
      out.pv.Mp(1:3,4)*v(4);
461 out.pv.pi_s_2 = v(1:3) + pinv(out.pv.Mp(4,1:3))*...
      out.pv.Mp(4,4)*v(4);
462
463
464 % Define expression and function for calculation of dvy
465 out.pv.dvy_tau = substituteParameters(out.pv.dynamics...
      (2));
466 state = sym('state',[8 1]);
467 out.pv.dvy_tau_fun = matlabFunction(subs(out.pv.dvy_tau...
      , [p;v], state), 'Vars', {state, tau_i}, 'File', '...
      Feedback_Linearisation/generatedFunctions/dvy_tau');
468
469
470 % Arrange model in form d(p,v) = f(p,v) + g u
471 out.pv.f = [out.pv.Rv*v; inv(out.pv.Mp)*(out.pv.F*v - ...
      out.pv.Cpv*v - out.pv.Gp)];
472 out.pv.gtau = [zeros(4,nw); inv(out.pv.Mp)*out.pv.B];
473 out.pv.g = [zeros(4,3); inv(out.pv.Mp)*out.pv.Bu];
474
475 % Define error model with rotated Cartesian position ...
```

```
    error
e = sym('e_',[8,1]); e_u = sym('e_u_',[nw 1]);  xr = ...
    sym('xr_',[8,1]); tau_r = sym('tau_r_',[nw,1]);

% Define Cartesian error in body frame
phi_e_xr = xr(3) - e(3);
e1e2 = subs(formula(R_EB_2d).', phi, phi_e_xr)*[xr(1)-p...
    (1);xr(2)-p(2)];
e1e2 = subs(e1e2, [p;v], xr-e);
e1 = e1e2(1); e2 = e1e2(2);

% Define dynamics
f_xr = subs(out.pv.dpdv, [p;v], xr);
f_xr = subs(f_xr, tau_i, tau_r);
f_xr_mns_e = subs(out.pv.dpdv, [p;v], xr-e);
f_xr_mns_e = subs(f_xr_mns_e, tau_i, tau_r-e_u);

% Define integral errors
syms e_x_int e_y_int e_phi_int real
e_z = blkdiag(subs(formula(R_EB_2d).', phi, phi_e_xr), ...
    1) * [e_x_int; e_y_int; e_phi_int];

% Define error dynamics
out.pv.de = [
-e_z(2)*f_xr_mns_e(3) + e1;
e_z(1)*f_xr_mns_e(3) + e2;
e(3);
e2*f_xr_mns_e(3) + cos(phi_e_xr)*f_xr(1) + sin(phi_e_xr...
    )*f_xr(2) - cos(phi_e_xr)*f_xr_mns_e(1) - sin(...
    phi_e_xr)*f_xr_mns_e(2);
-e1*f_xr_mns_e(3) - sin(phi_e_xr)*f_xr(1) + cos(...
    phi_e_xr)*f_xr(2) + sin(phi_e_xr)*f_xr_mns_e(1) - ...
    cos(phi_e_xr)*f_xr_mns_e(2);
f_xr(3:8) - f_xr_mns_e(3:8)
```

```
502 ];
503
504 % Linearise error dynamics about e=0
505 out.pv.Ae = subs(jacobian(out.pv.de, [e_x_int;e_y_int;...
       e_phi_int;e]), [e_x_int; e_y_int; e_phi_int; e;e_u],...
        [zeros(11,1); zeros(nw,1)]);
506 out.pv.Be = subs(jacobian(out.pv.de, e_u), [e_x_int; ...
       e_y_int; e_phi_int; e;e_u], [zeros(11,1); zeros(nw...
       ,1)]);
507
508 out.pv.Ae_fun = matlabFunction(simplify(...
       substituteParameters(out.pv.Ae),100), 'Vars', {xr, ...
       tau_r}, 'File', 'Differential_Flatness/...
       generatedFunctions/TVLQR_A_fun');
509 out.pv.Be_fun = matlabFunction(simplify(...
       substituteParameters(out.pv.Be),100), 'Vars', {xr, ...
       tau_r}, 'File', 'Differential_Flatness/...
       generatedFunctions/TVLQR_B_fun');
510 out.pv.e_fun = matlabFunction(subs([subs(e_z, e, xr-[p;...
       v]); formula(R_EB_2d.')*[xr(1)-p(1);xr(2)-p(2)]; xr...
       (3:8)-[p(3:4);v]], [p;v], state), 'Vars', {state,xr...
       ,[e_x_int;e_y_int;e_phi_int]}, 'File', '...
       Differential_Flatness/generatedFunctions/TVLQR_e_fun...
       ');
511
512 %% Feedback linearisation
513 % Uses the fully parametrised model, using Pathak's ...
       method but for the more general case
514 % This method should work with assymetric wheel ...
       configurations, odd
515 % roller angles etc.
516
517 % Calculate new input matrix P to simplify the input ...
       vector fields
```

```
518 out.pv.P = out.pv.g([5 7 8],:);
519
520 % New input vector fields g_hat - too complex to invert...
       P without some substitutions
521 out.pv.gHat = [
522 zeros(4,3);
523 1 0 0;
524 out.pv.g(6,:)*simplify(inv(substituteParameters(...
       out.pv.P, true)));
525 0 1 0;
526 0 0 1];
527
528 % Calculate feedback linearising input in x
529 w = sym('w', [3 1]);
530 out.pv.uFB = [
531 w(1) - out.pv.f(5);
532 (w(2) - out.pv.f(7));
533 w(3) - out.pv.f(8);
534 ];
535
536 out.pv.fNew = [
537 out.pv.f(1:4);
538 0;
539 out.pv.f(6) - out.pv.gHat(6,1)*out.pv.f(5) - ...
       out.pv.gHat(6,2)*out.pv.f(7) - out.pv.gHat(6,3)*...
       out.pv.f(8);
540 0;
541 0];
542
543 % Calculate change of coordinates z=T(x)
544 x = [p;v];
545 out.pz.T = [
546 x(1);
547 x(2);
```

```
548 x(3);
549 x(4);
550 x(5);
551 x(6) - x(5)*out.pv.gHat(6,1) - x(7)*out.pv.gHat(6,2) - ...
       x(8)*out.pv.gHat(6,3);
552 x(7);
553 x(8);
554 ];
555
556 % and inverse change of coordinates x=Tinv(z)
557 z = sym('z', [8 1], 'real');
558 out.pz.Tinv = [
559 z(1);
560 z(2);
561 z(3);
562 z(4);
563 z(5);
564 z(6) + subs(z(5)*out.pv.gHat(6,1) + z(7)*out.pv.gHat...
       (6,2) + z(8)*out.pv.gHat(6,3), x(4), z(4));
565 z(7);
566 z(8);
567 ];
568
569
570 % Calculate new drift and input vector fields in z
571 out.pz.f = subs([
572 cos(z(3))*z(5) - sin(z(3))*(z(6) + z(5)*out.pv.gHat...
       (6,1) + z(7)*out.pv.gHat(6,2) + z(8)*out.pv.gHat...
       (6,3));
573 sin(z(3))*z(5) + cos(z(3))*(z(6) + z(5)*out.pv.gHat...
       (6,1) + z(7)*out.pv.gHat(6,2) + z(8)*out.pv.gHat...
       (6,3));
574 z(7)
575 z(8);
```

```
576 0;
577 out.pv.f(6) - out.pv.f(5)*out.pv.gHat(6,1) - out.pv.f...
       (7)*out.pv.gHat(6,2) - out.pv.f(8)*out.pv.gHat(6,3) ...
       - z(5)*diff(out.pv.gHat(6,1),t) - z(7)*diff(...
       out.pv.gHat(6,2),t) - z(8)*diff(out.pv.gHat(6,3),t);
578 0;
579 0;
580 ], x, out.pz.Tinv);
581
582 out.pz.g = [
583 0 0 0
584 0 0 0
585 0 0 0
586 0 0 0
587 1 0 0
588 0 0 0
589 0 1 0
590 0 0 1
591 ];
592
593 % Zero dynamics should be calculated by setting dz=0 ...
       for controlled states,
594 % ie dthetap=dphi=w=0
595 % zeroDynamics = subs(substituteParameters(subs(...
       out.pz.f, [k_vr, k_vw, K_rw], [0 0 0])), z([3 4 5 7 ...
       8]), zeros(5,1));
596 zeroDynamics = subs(out.pz.f, [z([5 7 8]); w], zeros...
       (6,1));
597
598
599 % Calculate feedback linearising input in z
600 out.pz.uFB = subs(out.pv.uFB, [p;v], out.pz.Tinv);
601
602 % Derive expressions for dvy with and without ...
```

```
    acceleration due to rotation in terms of x
syms dvy real;
out.pv.dvy = out.pv.f(6) - out.pv.gHat(6,1)*out.pv.f(5)...
     - out.pv.gHat(6,2)*out.pv.f(7) - out.pv.gHat(6,3)*...
    out.pv.f(8) + out.pv.gHat(6,1)*w(1) + out.pv.gHat...
    (6,2)*w(2) + out.pv.gHat(6,3)*w(3);

% Create functions to return evaluation of fdvy - dvy ...
    for given
% state and input - used to numerically solve for ...
    theta_p_ss, i.e.
% f_dvy^-1
state = sym('state',[8 1]);
fun = subs(-dvy + substituteParameters(out.pv.dvy), x, ...
    state);
dFun = jacobian(fun,state(4));
out.pv.f_solve_fdvy_for_theta_p_x = matlabFunction(fun,...
     dFun, 'Vars', {state,w,dvy}, 'Outputs', {'fun','...
    dFun'}, 'File', 'Feedback_Linearisation/...
    generatedFunctions/f_solve_fdvy_theta_p_x');

% Define steady state (dtheta_p=w3=0) dvy's
syms dvy_ss real;
out.pv.dvy_ss = subs(substituteParameters(out.pv.dvy), ...
    {w(3) dtheta_p}, {0 0});

% Convert to function
state = sym('state',[8 1]);
out.pv.dvy_ss_fun = matlabFunction(formula(subs(...
    substituteParameters(out.pv.dvy_ss), x, state)), '...
    Vars', {state,w}, 'Outputs', {'dvy_ss'}, 'File', '...
    Feedback_Linearisation/generatedFunctions/dvy_ss_fun...
    ');

```

```
622 % Define steady state (constant dvy) derivative of dvy
623 % This should capture variation of dvy for a given ...
      theta_p due to
624 % dvx and ddphi
625 syms dtheta_p_ss real;
626 out.pv.ddvy_ss = subs(diff(out.pv.dvy_ss,t), diff(...
      theta_p), dtheta_p_ss);
627 out.pv.ddvy_ss = subs(out.pv.ddvy_ss, [diff(vx) diff(...
      dphi)], [w(1) w(2)]);
628
629 syms ddvy real;
630 out.pv.dtheta_p_ss = solve(ddvy==out.pv.ddvy_ss, ...
      dtheta_p_ss);
631
632 if nw==4
633
634 %% Velocity controller (body frame), constrained ...
      theta_p
635 syms Kv Kdphi Kr Kw1 Kw2 theta_p_max w1_max w2_max ...
      theta_pr fss_minus1_dvy0 vxr vyr dphir ...
      K_slow_dtheta_p K_slow_theta_p real;
636 body_vel_control_dtheta_pr = -subs(out.pv.dtheta_p_ss, ...
      {theta_p, ddvy}, {theta_pr, 0})*(theta_p_max^2 - ...
      theta_pr^2) / (fss_minus1_dvy0*theta_pr - ...
      theta_p_max^2) ...
637 - Kr*(theta_pr - fss_minus1_dvy0)*(fss_minus1_dvy0*...
      theta_pr - theta_p_max^2) ...
638 - ((theta_p_max^2-theta_pr^2)^2 * subs(out.pv.dvy_ss,...
      theta_p,theta_pr)*Kv*(vyr-vy)) / ((fss_minus1_dvy0*...
      theta_pr - theta_p_max^2)*(theta_pr - ...
      fss_minus1_dvy0));
639 body_vel_control_dw1 = -Kw1*w(1) + Kv*(vxr-vx)*(w1_max...
      ^2 - w(1)^2)^2;
640 body_vel_control_dw2 = -Kw2*w(2) + Kdphi*(dphir-dphi)*(...
```

```
    w2_max^2 - w(2)^2)^2;
641
642 % Slow response
643 body_vel_control_dtheta_pr = body_vel_control_dtheta_pr...
     * exp(-K_slow_dtheta_p*abs(dtheta_p) - ...
    K_slow_theta_p*abs(theta_p-theta_pr));
644
645 out.pv.body_vel_control_dw1_dw2_dtheta_pr_fun = ...
    matlabFunction( ...
646 formula(subs(formula(body_vel_control_dtheta_pr), x, ...
    state)), ...
647 formula(subs(formula(body_vel_control_dw1), x, state)),...
     ...
648 formula(subs(formula(body_vel_control_dw2), x, state)),...
     ...
649 'Vars', {state w Kv Kdphi Kr Kw1 Kw2 theta_p_max w1_max...
     w2_max theta_pr fss_minus1_dvy0 vxr vyr dphir ...
    K_slow_dtheta_p K_slow_theta_p}, 'Outputs', {'...
    dtheta_pr' 'dw1' 'dw2'}, 'File', '...
    Feedback_Linearisation/generatedFunctions/...
    body_vel_control_dw1_dw2_dtheta_pr_fun', 'Optimize',...
     false);
650
651
652 %% Velocity controller (inertial frame), constrained ...
    theta_p
653 syms Kv Kdphi Kr Kw1 Kw2 theta_p_max w1_max w2_max ...
    theta_pr fss_minus1_dvy_mvxdphi dxr dyr dphir real;
654 inertial_vel_control_dtheta_pr = -subs(...
    out.pv.dtheta_p_ss, {theta_p, ddvy}, {...
    fss_minus1_dvy_mvxdphi, -w(1)*x(7) - w(2)*x(5)})*(...
    theta_p_max^2 - theta_pr^2) / (...
    fss_minus1_dvy_mvxdphi*theta_pr - theta_p_max^2) ...
655 + exp(-K_slow_dtheta_p*abs(dtheta_p) - K_slow_theta_p*...
```

```
    abs(theta_p-theta_pr))*( ...
656 - Kr*(theta_pr - fss_minus1_dvy_mvxdphi)*(...
    fss_minus1_dvy_mvxdphi*theta_pr - theta_p_max^2) ...
657 - ((theta_p_max^2-theta_pr^2)^2 * (subs(out.pv.dvy_ss, ...
    theta_p, theta_pr) + vx*dphi)*Kv*(vyr-vy)) / ((...
    fss_minus1_dvy_mvxdphi*theta_pr - theta_p_max^2)*(...
    theta_pr - fss_minus1_dvy_mvxdphi)) ...
658 );
659
660 %inertial_vel_control_dw1 = (dphi*out.pv.dvy_ss + vy*w...
    (2)) -Kw1*(w(1) - dphi*vy) + Kv*(vxr-vx)*(w1_max - w...
    (1) + vy*dphi)^2*(w1_max + w(1) - vy*dphi)^2 / ...
    w1_max;
661 inertial_vel_control_dw1 = (dphi*out.pv.dvy_ss + vy*w...
    (2)) -Kw1*(w(1) - dphi*vy) + Kv*(vxr-vx)*(w1_max - w...
    (1) + vy*dphi)^2*(w1_max + w(1) - vy*dphi)^2 / ...
    w1_max;
662 inertial_vel_control_dw2 = -Kw2*w(2) + Kdphi*(dphir-...
    dphi)*(w2_max^2 - w(2)^2)^2;
663
664 % Substitute vxr for dxr etc
665 inertial_vel_control_dtheta_pr = subs(...
    inertial_vel_control_dtheta_pr, [vxr; vyr], R_EB_2d....
    '*[dxr;dyr]);
666 inertial_vel_control_dw1 = subs(...
    inertial_vel_control_dw1, [vxr; vyr], R_EB_2d.'*[dxr...
    ;dyr]);
667 inertial_vel_control_dw2 = subs(...
    inertial_vel_control_dw2, [vxr; vyr], R_EB_2d.'*[dxr...
    ;dyr]);
668
669
670 out.pv.inertial_vel_control_dw1_dw2_dtheta_pr_fun = ...
    matlabFunction( ...
```

```
671 formula(subs(formula(inertial_vel_control_dtheta_pr), x...
       , state)), ...
672 formula(subs(formula(inertial_vel_control_dw1), x, ...
       state)), ...
673 formula(subs(formula(inertial_vel_control_dw2), x, ...
       state)), ...
674 'Vars', {state w Kv Kdphi Kr Kw1 Kw2 theta_p_max w1_max...
        w2_max theta_pr dxr dyr dphir ...
       fss_minus1_dvy_mvxdphi K_slow_dtheta_p ...
       K_slow_theta_p}, 'Outputs', {'dtheta_pr' 'dw1' 'dw2'...
       }, 'File', 'Feedback_Linearisation/...
       generatedFunctions/...
       inertial_vel_control_dw1_dw2_dtheta_pr_fun', '...
       Optimize', true);
675
676
677 % Inertial velocity controller with differential ...
       flatness derived perturbations
678 syms Kvx Kvy Kdphi Kr Kw1 Kw2 theta_p_max w1_max w2_max...
        theta_pr fss_minus1_dvy_f_dvyp dxr dyr dphir real;
679
680 % Circular dependency here
681 [dxp, dyp, vxp, vyp, dvxp, dvyp, ddvxp, ddvyp] = ...
       calculateInertialVelocityPerturbations(dxr, dyr, phi...
       , dphir);
682
683 inertial_vel_control_dtheta_pr = ...
684 -subs(out.pv.dtheta_p_ss, {theta_p, ddvy}, {...
       fss_minus1_dvy_f_dvyp, ddvyp})*(theta_p_max^2 - ...
       theta_pr^2) / (fss_minus1_dvy_f_dvyp*theta_pr - ...
       theta_p_max^2) ...
685 + exp(-K_slow_dtheta_p*abs(dtheta_p) - K_slow_theta_p*...
       abs(theta_p-theta_pr))*( ...
686 - Kr*(theta_pr - fss_minus1_dvy_f_dvyp)*(...
```

```
    fss_minus1_dvy_f_dvyp*theta_pr - theta_p_max^2) ...
687 - ((theta_p_max^2-theta_pr^2)^2 * (subs(out.pv.dvy_ss, ...
    theta_p, theta_pr) - dvyp)*Kvy*(vyr+vyp-vy)) / ((...
    fss_minus1_dvy_f_dvyp*theta_pr - theta_p_max^2)*(...
    theta_pr - fss_minus1_dvy_f_dvyp)) ...
688 );
689
690 inertial_vel_control_dw1 = ddvxp - Kw1*(w(1) - dvxp) + ...
    Kvx*(vxr+vxp-vx)*(w1_max+dvxp - w(1))^2*(w1_max - ...
    dvxp + w(1))^2 / w1_max^2;
691 inertial_vel_control_dw2 = -Kw2*w(2) + Kdphi*(dphir-...
    dphi)*(w2_max^2 - w(2)^2)^2;
692
693 % Substitute vxr for dxr etc
694 inertial_vel_control_dtheta_pr = subs(...
    inertial_vel_control_dtheta_pr, [vxr; vyr], R_EB_2d....
    '*[dxr;dyr]);
695 inertial_vel_control_dw1 = subs(...
    inertial_vel_control_dw1, [vxr; vyr], R_EB_2d.'*[dxr...
    ;dyr]);
696 inertial_vel_control_dw2 = subs(...
    inertial_vel_control_dw2, [vxr; vyr], R_EB_2d.'*[dxr...
    ;dyr]);
697
698
699 out.pv.inertial_vel_control_dw1_dw2_dtheta_pr_fun = ...
    matlabFunction( ...
700 simplify(subs(formula(inertial_vel_control_dtheta_pr), ...
    formula(vertcat(x{:})), state)), ...
701 simplify(subs(formula(inertial_vel_control_dw1), ...
    formula(vertcat(x{:})), state)), ...
702 simplify(subs(formula(inertial_vel_control_dw2), ...
    formula(vertcat(x{:})), state)), ...
703 'Vars', {state w Kvx Kvy Kdphi Kr Kw1 Kw2 theta_p_max ...
```

```
    w1_max w2_max theta_pr dxr dyr dphir ...
    fss_minus1_dvy_f_dvyp K_slow_dtheta_p K_slow_theta_p...
    }, ...
'Outputs', {'dtheta_pr' 'dw1' 'dw2'}, ...
'File', 'Feedback_Linearisation/generatedFunctions/...
    inertial_vel_control_2_dw1_dw2_dtheta_pr_fun', ...
'Optimize', true);


%% Position controller
syms K_vr K_p K_dphir K_phi v_max dphi_max  xr yr phir ...
    K_slow real;
dxdy = formula(R_EB_2d*[x(5); x(6);]);
ddxr = -K_vr*dxr + (v_max^2 - dxr^2 - dyr^2)^2 * K_p*(...
    xr - x(1));
ddyr = -K_vr*dyr + (v_max^2 - dxr^2 - dyr^2)^2 * K_p*(...
    yr - x(2));
ddphir = -K_dphir*dphir + (dphi_max^2 - dphir^2)^2 * ...
    K_phi*(phir - x(3));

ddxr = ddxr * exp(-K_slow * abs(dxdy(1) - dxr));
ddyr = ddyr * exp(-K_slow * abs(dxdy(2) - dyr));
ddphir = ddphir * exp(-K_slow * abs(x(7) - dphir));

out.pv.inertial_pos_control_ddxr_ddyr_ddphir_fun = ...
    matlabFunction( ...
subs(ddxr, x, state), ...
subs(ddyr, x, state), ...
subs(ddphir, x, state), ...
'Vars', {state K_vr K_p K_dphir K_phi v_max dphi_max ...
    dxr dyr dphir xr yr phir K_slow}, 'Outputs', {'ddxr'...
     'ddyr' 'ddphir'}, 'File', 'Feedback_Linearisation/...
    generatedFunctions/...
    inertial_pos_control_ddxr_ddyr_ddphir_fun', '...
```

```
    Optimize', true);
end

%% Nonlinear controllability
runNonlinearControllability = false;
if runNonlinearControllability
% Put system into nonlinear input-affine form dx = f(x)...
     + sum_{j=1}^4 g_j(x)u_j
x = sym('x',[8 1],'real');
f = substituteParameters(subs([out.pv.Rv*v; inv(...
    out.pv.Mp)*(out.pv.F*v - out.pv.Cpv*v - out.pv.Gp)],...
     [p;v], x));
g = substituteParameters(subs([zeros(4,3); (inv(...
    out.pv.Mp)*out.pv.Bu)], [p;v], x));
%     f = substituteParameters(subs(subs([out.pv.Rv*v; ...
    inv(out.pv.Mp)*(out.pv.F*v - out.pv.Cpv*v - ...
    out.pv.Gp)], [p;v], x),[h_p k_vw k_vr],[0 0 0]));
%     g = substituteParameters(subs(subs([zeros(4,3); (...
    inv(out.pv.Mp)*out.pv.Bu)], [p;v], x),[h_p k_vw k_vr...
    ],[0 0 0]));
f = subs(f, [p;v], x);
g = subs(g, [p;v], x);

% Define Lie bracket operation
lb = @(g1,g2,x) jacobian(g2,x)*g1 - jacobian(g1,x)*g2;


% Confirm G_0 is the involutive closure of G - not the ...
    case unless dvx
% and ddphi are decoupled
lieBracket(g,g,x)

Q_0 = g;
Q_0_closure = Q_0;
```

```
749 Q_1 = [Q_0_closure lb(f,g(:,1),x) lb(f,g(:,2),x) lb(f,g...
       (:,3),x)];
750 Q_1_closure = [Q_1 lieBracket(Q_1,Q_1,x)];
751
752 % Controllability indices
753 r_0 = rank(Q_0);
754 r_1 = rank(Q_1) - rank(Q_0);
755
756
757 % Strong accessibility / STLA
758 Delta{1} = [f g];
759 disp(['rank(Delta_1) = ' num2str(rank(Delta{1}))]);
760 for i=2:100
761 Delta{i} = [Delta{i-1} lieBracket(Delta{1},Delta{i-1},x...
       )];
762 disp(['rank(Delta_' num2str(i) ') = ' num2str(rank(...
       Delta{i}))]);
763 if rank(Delta{i}) == rank(Delta{i-1})
764 break;
765 end
766 end
767 end
768
769
770 %% Internal dynamics derivation
771 % Eliminate tau by subtracting second and fourth ...
       expressions
772 out.internalDynamics = collect(out.pv.model(2)*r_w - ...
       out.pv.model(4), tau_i);
773
774 if any(ismember(tau_i, symvar(out.internalDynamics)))
775 error('Internal dynamics still contain tau');
776 end
777
```

```
778
779 %% Generate C code
780 codegen f_solve_fdvy_theta_p_x  -d ./...
       Feedback_Linearisation/codegen/...
       f_solve_fdvy_theta_p_x -c -args {zeros(8,1), zeros...
       (3,1), 0}
781
782 temp = sym('temp', [8 1]);
783 matlabFunction(subs(substituteParameters(out.pz.T), x, ...
       temp), 'Vars', {temp}, 'Outputs', {'z'}, 'File', '...
       Feedback_Linearisation/generatedFunctions/T_x_to_z')...
       ;
784 matlabFunction(substituteParameters(out.pz.uFB), 'Vars'...
       , {z,w}, 'Outputs', {'v'}, 'File', '...
       Feedback_Linearisation/generatedFunctions/uFB');
785
786
787 vin = sym('vin', [3 1]);
788 matlabFunction(subs(inv(substituteParameters(out.pv.P))...
       *vin, x, substituteParameters(out.pz.Tinv)), 'Vars',...
        {z,vin}, 'Outputs', {'u'}, 'File', '...
       Feedback_Linearisation/generatedFunctions/Pz_v_to_u'...
       );
789
790
791 codegen T_x_to_z -d ./Feedback_Linearisation/codegen/...
       T_x_to_z -c -args {zeros(8,1)}
792 codegen uFB -d ./Feedback_Linearisation/codegen/uFB -c ...
       -args {zeros(8,1), zeros(3,1)}
793 codegen Pz_v_to_u -d ./Feedback_Linearisation/codegen/...
       Pz_v_to_u -c -args {zeros(8,1), zeros(3,1)}
794 codegen dvy_ss_fun -d ./Feedback_Linearisation/codegen/...
       dvy_ss -c -args {zeros(8,1), zeros(3,1)}
795 codegen body_vel_control_dw1_dw2_dtheta_pr_fun -d ./...
```

```
    Feedback_Linearisation/codegen/...
    body_vel_control_dw1_dw2_dtheta_pr_fun -c -args {...
    zeros(8,1) zeros(3,1) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0}
796 codegen inertial_vel_control_dw1_dw2_dtheta_pr_fun -d ....
    /Feedback_Linearisation/codegen/...
    inertial_vel_control_dw1_dw2_dtheta_pr_fun -c -args ...
    {zeros(8,1) zeros(3,1) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
    0}
797 codegen inertial_pos_control_ddxr_ddyr_ddphir_fun -d ./...
    Feedback_Linearisation/codegen/...
    inertial_pos_control_ddxr_ddyr_ddphir_fun -c -args {...
    zeros(8,1) 0 0 0 0 0 0 0 0 0 0 0 0 0}
798 codegen dvy_tau -d ./Feedback_Linearisation/codegen/...
    dvy_tau_fun -c -args {zeros(8,1) zeros(nw,1)}
799
800
801
802 %% Generate optimised simulink blocks and matlab ...
    functions
803 stateInput = sym('stateInput', [8 1]);
804 if generateBlocks
805 % Create matlab function for pv dynamics model
806 out.dpdv = matlabFunction(subs(substituteParameters(...
    out.pv.dpdv), [p;v], stateInput) ...
807 , 'vars',   {stateInput, tau_i} ...
808 , 'outputs', {'dpdv'});
809
810 % Create/open simulink library
811 if ¬exist('model_library', 'file')
812 new_system  ('model_library', 'Library');
813 else
814 load_system('model_library');
815 end
816 set_param('model_library','Lock','off');
```

```
817
818 %     % Create pdp model block
819 %     matlabFunctionBlock('model_library/dynamics_dpddp...
      ', subs(substituteParameters(out.pdp.dpddp), [p;dp],...
       stateInput) ...
820 %         , 'vars',   {stateInput, tau_i} ...
821 %         , 'outputs', {'dpddp'});
822
823 % Create pv model block
824 matlabFunctionBlock('model_library/dynamics_dqdv', subs...
      (substituteParameters(out.pv.dpdv), [p;v], ...
      stateInput) ...
825 , 'vars',   {stateInput, tau_i} ...
826 , 'outputs', {'dpdv'});
827
828 % Create pv inverse kinematics block
829 matlabFunctionBlock('model_library/...
      qv_inverse_kinematics',subs(substituteParameters(...
      out.pv.inverseKinematics), [p;v], stateInput) ...
830 , 'vars',   {stateInput} ...
831 , 'outputs', {'dtheta_i'});
832
833 %     % Create zeta->u block
834 %     matlabFunctionBlock('model_library/pz_zeta_to_u',...
       subs(substituteParameters(out.pz.uFB), [p;z], ...
      stateIn) ...
835 %         , 'vars',   {stateIn, zeta} ...
836 %         , 'outputs', {'u'});
837
838 % Create feedback linearising input block
839 matlabFunctionBlock('model_library/uFB_z',...
      substituteParameters(out.pz.uFB) ...
840 , 'vars',   {z,w} ...
841 , 'outputs', {'v'});
```

```
842
843 % Create x->z transform z=T(x)
844 matlabFunctionBlock('model_library/T_x_to_z',subs(...
       substituteParameters(out.pz.T), x, stateInput) ...
845 , 'vars',   {stateInput} ...
846 , 'outputs', {'z'});
847
848 %     % Create z->x transform x=Tinv(z)
849 %     matlabFunctionBlock('model_library/T_z_to_x',...
       substituteParameters(out.pz.Tinv) ...
850 %         , 'vars',   {z} ...
851 %         , 'outputs', {'x'});
852
853 %     % Create u->v mapping function v=Pu in terms of x
854 %     u = sym('u', [3 1], 'real');
855 %     matlabFunctionBlock('model_library/Px_u_to_v',...
       subs(substituteParameters(out.pv.P*u), x, stateInput...
       ) ...
856 %         , 'vars',   {stateInput, u} ...
857 %         , 'outputs', {'v'});
858
859 %     % Create u->v mapping function v=Pu in terms of z
860 %     matlabFunctionBlock('model_library/Pz_u_to_v',...
       subs(substituteParameters(out.pv.P*u), x, ...
       substituteParameters(out.pz.Tinv)) ...
861 %         , 'vars',   {z, u} ...
862 %         , 'outputs', {'v'});
863
864 %     % Create v->u mapping function v=Pu in terms of x
865 %     vin = sym('vin', [3 1], 'real');
866 %     matlabFunctionBlock('model_library/Px_v_to_u',...
       subs(inv(substituteParameters(out.pv.P)*vin), x, ...
       stateInput) ...
867 %         , 'vars',   {stateInput, vin} ...
```

```
868 %          , 'outputs', {'u'});
869
870 % Create v->u mapping function v=Pu in terms of z
871 vin = sym('vin', [3 1], 'real');
872 matlabFunctionBlock('model_library/Pz_v_to_u',subs(inv(...
       substituteParameters(out.pv.P))*vin, x, ...
       substituteParameters(out.pz.Tinv)) ...
873 , 'vars',   {z, vin} ...
874 , 'outputs', {'u'});
875
876 save_system ('model_library',[],'...
       OverwriteIfChangedOnDisk',true);
877 close_system('model_library');
878 end
879
880 %% Generate linear systems
881 dqdvSubbed = subs(subs(substituteParameters(out.pv.dpdv...
       ), [p;v], stateInput), {cos(phi), sin(phi)}, {1 0});
882 A = double(subs(jacobian(dqdvSubbed, stateInput), [...
       stateInput; tau_i], zeros(8+nw,1)));
883 B = double(subs(jacobian(dqdvSubbed, tau_i), [...
       stateInput; tau_i], zeros(8+nw,1)));
884 C = eye(8);
885
886 out.linsys = ss(A,B,C,0);
887
888 %% Store function handles for later use
889 out.substituteAlphas = @(x) (substituteAlphas(x));
890 out.substituteParameters = @(x) (substituteParameters(x...
       ));
891 out.removeWheelMassInertiaTerms = @(x) (...
       removeWheelMassInertiaTerms(x));
892 out.stripTimeDependence = @(x) (stripTimeDependence(x))...
       ;
```

```
893
894 end
895
896 % Function to calculate all possible Lie brackets of ...
       two distributions
897 function out = lieBracket(F,G,x,depth)
898
899 if ¬exist('depth','var')
900 depth = 1;
901 end
902
903 % Define Lie bracket operator
904 lb = @(g1,g2,x) jacobian(g2,x)*g1 - jacobian(g1,x)*g2;
905
906 out = sym([]);
907 for i=1:size(F,2)
908 for j=i+1:size(G,2)
909 out = [out lb(F(:,i),G(:,j),x)];
910 if depth>1
911 for k=1:depth-1
912 out = [out lb(out(:,end),G(:,j),x)];
913 end
914 end
915 end
916 end
917 end
918
919
920 function [out] = substituteParameters(in, ...
       simplificationForInversion)
921
922 if ¬exist('simplificationForInversion', 'var')
923 simplificationForInversion = false;
924 end
```

```
925
926 syms m_p m_w I_pbx I_pby I_pbz I_wx I_wyz h_p l_1 l_2 ...
      l_3 l_4 r_w r_r g alpha_1 alpha_2 alpha_3 alpha_4 ...
      k_vw k_vr K_rw real;
927 names={}; values = {};
928 names{end+1} = alpha_1; values{end+1} = pi/4;
929 names{end+1} = alpha_2; values{end+1} = -pi/4;
930 names{end+1} = alpha_3; values{end+1} = pi/4;
931 names{end+1} = alpha_4; values{end+1} = -pi/4;
932 if ¬simplificationForInversion
933 names{end+1} = I_pbx;    values{end+1} = 0.0315; % ...
      Measured in pendulum experiment
934 names{end+1} = I_pby;    values{end+1} = 0.0534;
935 names{end+1} = I_pbz;    values{end+1} = 0.0271;
936 names{end+1} = I_wx;    values{end+1} = 5.12e-5; %5.5e...
      -05; % Thoroughly estimated at 500Hz using ...
      fit_wheel_inertia_model_accel
937 names{end+1} = I_wyz;   values{end+1} = 0.00011;
938 names{end+1} = h_p;     values{end+1} = 4*0.145*0.07/(3...
      .22 - 4*0.145) + 0.07;
939 names{end+1} = m_p;     values{end+1} = 3.22 - 4*0.145;
940 names{end+1} = m_w;     values{end+1} = 0.145;
941 names{end+1} = l_1;     values{end+1} = -0.105;
942 names{end+1} = l_2;     values{end+1} = -0.063;
943 names{end+1} = l_3;     values{end+1} = 0.063;
944 names{end+1} = l_4;     values{end+1} = 0.105;
945 names{end+1} = r_w;     values{end+1} = 0.0595/2;
946 names{end+1} = r_r;     values{end+1} = 0.0055;
947 names{end+1} = g;       values{end+1} = 9.81;
948 names{end+1} = k_vw;     values{end+1} = 2.33e-5;
949 names{end+1} = k_vr;     values{end+1} = 1.01e-4;% 9.7e...
      -06;
950 names{end+1} = K_rw;     values{end+1} = 1.97e-4;%(3e...
      -04 - 1.5e-5)*0.65;
```

```
951 else
952 names{end+1} = l_3;     values{end+1} = -l_2;
953 names{end+1} = l_4;     values{end+1} = -l_1;
954 end
955 out = subs(in, names,  values);
956 end
957
958 function [out] = substituteAlphas(in)
959 syms alpha_1 alpha_2 alpha_3 alpha_4 real;
960 names={}; values = {};
961 names{end+1} = alpha_1; values{end+1} = pi/4;
962 names{end+1} = alpha_2; values{end+1} = -pi/4;
963 names{end+1} = alpha_3; values{end+1} = pi/4;
964 names{end+1} = alpha_4; values{end+1} = -pi/4;
965 %     names{end+1} = alpha_1; values{end+1} = pi/4;
966 %     names{end+1} = alpha_2; values{end+1} = -pi/4;
967 %     names{end+1} = alpha_3; values{end+1} = -pi/4;
968 %     names{end+1} = alpha_4; values{end+1} = pi/4;
969 out = simplify(subs(in, names,  values));
970 end
971
972 function [ out ] = stripTimeDependence(in)
973 syms(symvar(in));
974 if contains(char(in), 'diff')
975 error('Cannot remove time dependence when expression ...
      contains differentials')
976 end
977 removeTimeDep = @(expr) str2sym(strrep(char(expr), '(t)...
      ', ''));
978 out = arrayfun(removeTimeDep, in);
979 end
```