Model Predictive Control for Cooperative Multi-UAV Systems

ROBERTO ANTONIO CASTRO SUNDIN
Model Predictive Control for Cooperative Multi-UAV Systems

ROBERTO ANTONIO CASTRO SUNDIN

Degree Programme in Engineering Physics
Date: June 21, 2021
Supervisor: Pedro Delgado Roque
Examiner: Dimos Dimarogonas
School of Electrical Engineering and Computer Science
Swedish title: Modellprediktiv reglering för samarbetande flerdrönarsystem
Model Predictive Control for Cooperative Multi-UAV Systems / Modellprediktiv reglering för samarbetande flerdrönarsystem

© 2021 Roberto Antonio Castro Sundin
Abstract

The maneuverability and freedom provided by unmanned aerial vehicles (UAVs) make these an interesting choice for transporting objects in settings such as search and rescue operations, construction, and smart factories. A commonly proposed method of transport is by using cables attached between each UAV and the payload. However, the geometrical constraints posed by these attachments typically result in a system with highly complex dynamics. Although not an issue for conventional PID control schemes, these complex dynamics make the direct application of model predictive controllers (MPCs) infeasible for real-time usage. For this reason, much of the previous work has focused on treating the payload as a disturbance, thereby losing the ability to predict its effect on the UAVs. Contrary to this, this thesis presents an MPC that both captures the dynamics of the payload, and is capable of real-time usage. This is made possible by a parametrized linearization of the original system, and results in greatly improved performance compared to the disturbance model approach. The controller is derived for a system with two UAVs that transport a bar-like payload and verified both in simulations and physical experiments. The resulting control system is able track a multitude of setpoints, including rotations of both payload and UAVs, as well as lateral translations. Furthermore, it is able to attenuate external disturbances well, and dampens and prevents oscillations more efficiently when compared to the disturbance based approach. The resulting MPC solving time is on the order of milliseconds. Additionally, an initial attempt to decentralize the system is made, and the resulting controller experimentally tested on the UAV–bar system, resulting in a lower MPC solving time (2.5 times faster on average), but worsened performance in terms of position tracking of the bar.

Keywords

Control theory; Model Predictive Control; Unmanned Aerial Vehicles; Airborne payload transport
Sammanfattning

Den manövrerbarað och frihet som möjliggörs av användandet utav obemanndade luftfarkoster (drönare) gör dessa till tämligen intressanta kandidater för lasttransport inom områden såsom sök- och räddningsuppdrag, byggnadskonstruktion och s.k. smarta fabriker. En vanligen förespråkad transportmetod består utav att förse systemet med kablar som fästs mellan last och drönare. De geometriska restriktioner som denna lastkoppling innebär resulterar ofta i system med väldigt komplicerad dynamik och interaktionskrafter. Även om detta inte innebär något problem för konventionella PID reglersystem så omöjliggör detta det direkta applicerandet utav modellprediktiv reglering (MPC) för realtidsbruk. Av denna anledning har tidigare verk fokuserat på att behandla lasten och dess inverkan på drönarna som en störning, men med detta därmed förborrar möjligheten att förutsöpa dess effekt på drönarna. I kontrast till detta, kommer det i detta verk att presenteras en MPC som både fångar lastens dynamik och är snabb nog för realtidsanvändning. Detta gör möjligt utav en parametrizerad linjärisering utav originalsystemet och ger märkbara bättre resultat än den störningaröverförande modellen. Reglersystemet appliceras på ett system bestående utav två drönare och en stång-liknande last och resultatet verifieras både i form av numeriska simuleringar och fysiska experiment. Det resulterande systemet klarar av både rotationer utav last och drönare samt translationer i alla riktningar. Dessutom är systemet kapabelt att hantera externa störningar och både dämpar och förhindrar oscillationer bättre i jämförelse med reglersystemet baserat på störningsmodeller. Lösningsstiden för MPC-regulatorn är i storleksordningen millisekunder. Utöver detta görs ett initialt försök i att decentralisera tidigare nämnda MPC och det resulterande reglersystemet utvärderas experimentellt på samma drönarsystem som tidigare. Detta resulterar i en lägre lösningsstid (2.5 ggr snabbare i genomsnitt), men även i försämrad prestanda med avseende på reglering av stångens position.

Nyckelord

Reglerteknik; Modellprediktiv reglering; Obemannade luftfarkoster; Luftburens lasttransport
iv I Sammanfattning
Acknowledgments

First of all I’d like to thank both my examiner and supervisor for allowing me to be part of this project! Secondly, I would like to thank my supervisor Pedro for his continuous support throughout this work.

A fact that should not remain unknown, is the insane amount of support I got from Robin Baran during the experimental setup phase at the Smart Mobility Lab. With his help, the hardware configuration became a (somewhat) smooth process even for a theoretically minded person like me, so thanks for that!

Finally, a big thanks to my friends and family for making the time outside of this project a fun and relaxing one! And, of course, a special thanks to Sabina.

Stockholm, June 2021
Roberto Antonio Castro Sundin
vi | Acknowledgments
Contents

1 Introduction 1
   1.1 Literature review ................................... 5
   1.2 Outline ........................................... 7

2 Background 9
   2.1 Optimization and mathematical programming ........... 9
      2.1.1 Dynamic Programming ............................ 12
   2.2 Model Predictive Control ............................... 13
      2.2.1 A general MPC problem .......................... 14
      2.2.2 Linear Quadratic Control ....................... 16
      2.2.3 MPC stability .................................. 18
      2.2.4 Offset-free MPC ................................ 19
   2.3 Numerical simulation .................................. 23

3 System analysis and controller design 27
   3.1 UAVs/Quadrotors ..................................... 27
      3.1.1 Modeling of a single UAV ......................... 27
      3.1.2 Modeling of a UAV–bar system .................. 30
      3.1.3 Equilibrium points for the UAV–bar system .... 32
      3.1.4 Modeling of a single UAV with payload ........ 35
   3.2 Controller derivations ................................ 36
      3.2.1 MPC for a single UAV ............................. 36
      3.2.2 Offset-free MPC for a single UAV ............... 38
      3.2.3 Centralized MPC for a UAV–bar system ........ 39
      3.2.4 Decentralized MPC for a UAV–bar system .... 40

4 Methods 43
   4.1 Simulation setup ..................................... 43
      4.1.1 Simulation environment ........................... 43
      4.1.2 Simulation experiments ............................ 44
4.2 Experimental setup ........................................... 46
  4.2.1 ROS .................................................... 46
  4.2.2 PX4 .................................................... 48
  4.2.3 System input considerations ............................ 48
  4.2.4 Experimental validation, Centralized MPC ............ 49
  4.2.5 Experimental validation, Decentralized MPC .......... 51

5 Results ......................................................... 55
  5.1 Simulations .................................................. 55
    5.1.1 Damping of bar angular oscillation .................... 55
    5.1.2 Damping of bar linear oscillation ...................... 55
    5.1.3 General performance ................................... 58
  5.2 Experiments .................................................. 62
    5.2.1 Centralized MPC .................................... 62
    5.2.2 Decentralized MPC ................................... 63
    5.2.3 Comparison ............................................. 65

6 Conclusions and Future work .................................. 71
  6.1 Future work ................................................ 72

References ....................................................... 75

A Linearization and discretization ............................... 81
  A.1 Linearization ............................................. 81
  A.2 Discretization .............................................. 82

B UAV–bar linearized model ...................................... 83
# List of Figures

1.1 YuMi® – A collaborative robot developed by ABB. .................................. 2
1.2 Agents with conflicting individual objectives. ........................................... 3
1.3 The system treated in this thesis. ............................................................... 4
1.4 Payload transport without communication. ................................................. 5

2.1 The cycloid as defined in Equation (2.1). ................................................. 10
2.2 Numerical solution to a discretized Brachistochrone problem. .................... 11
2.3 MPC control law for a simple system. ....................................................... 16
2.4 Finite horizon LQR response. ................................................................. 18
2.5 Block scheme illustrating an offset-free MPC approach. ............................ 20
2.6 MPC with model errors. ........................................................................... 21
2.7 Offset-free MPC with model errors. ......................................................... 22
2.8 Simulated pendulum trajectory for different simulation techniques. ............. 26

3.1 A single UAV. ............................................................................................. 28
3.2 UAV–bar system. ......................................................................................... 30
3.3 Schematic for illustrating the definition of parametrization variables $\theta_f$ and $\theta_l$. .......................................................... 32
3.4 Solutions to Equation (3.37). .................................................................... 34
3.5 A single UAV with a payload of mass $m_p$ attached to it by the means of a cable. .......................................................... 36

4.1 Images showing the setpoints described in Table 4.4. ................................. 47
4.2 Illustrative ROS nodes and topics diagram. ............................................... 47
4.3 PX4 internal control scheme. ................................................................. 48
4.4 PX4 angular rate controller. ................................................................. 49
4.5 mRo Pixracer with ESP8266 Wi-Fi module. .............................................. 50
4.6 Connection diagram ................................................................................. 50
4.7 Graph displaying ROS architecture during experiments. .......................... 52
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Simulation evaluation of damping of bar angular oscillation, position.</td>
<td>56</td>
</tr>
<tr>
<td>5.2</td>
<td>Simulation evaluation of damping of bar angular oscillation, orientation.</td>
<td>56</td>
</tr>
<tr>
<td>5.3</td>
<td>Simulation evaluation of damping of bar angular oscillation, thrust.</td>
<td>57</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulation evaluation of damping of bar angular oscillation, torque.</td>
<td>57</td>
</tr>
<tr>
<td>5.5</td>
<td>Simulation evaluation of damping of bar linear oscillation, position.</td>
<td>58</td>
</tr>
<tr>
<td>5.6</td>
<td>Simulation evaluation of damping of bar linear oscillation, orientation.</td>
<td>59</td>
</tr>
<tr>
<td>5.7</td>
<td>Simulation evaluation of damping of bar linear oscillation, thrust.</td>
<td>59</td>
</tr>
<tr>
<td>5.8</td>
<td>Simulation evaluation of damping of bar linear oscillation, torque.</td>
<td>60</td>
</tr>
<tr>
<td>5.9</td>
<td>Simulation evaluation of general performance, position.</td>
<td>60</td>
</tr>
<tr>
<td>5.10</td>
<td>Simulation evaluation of general performance, orientation.</td>
<td>61</td>
</tr>
<tr>
<td>5.11</td>
<td>Simulation evaluation of general performance, thrust.</td>
<td>61</td>
</tr>
<tr>
<td>5.12</td>
<td>Simulation evaluation of general performance, torque.</td>
<td>62</td>
</tr>
<tr>
<td>5.13</td>
<td>Centralized MPC experimental results showing position and orientation.</td>
<td>63</td>
</tr>
<tr>
<td>5.14</td>
<td>Centralized MPC experimental results showing thrust and torque.</td>
<td>64</td>
</tr>
<tr>
<td>5.15</td>
<td>CPU time for centralized MPC experiment.</td>
<td>64</td>
</tr>
<tr>
<td>5.16</td>
<td>Decentralized MPC experimental results showing position of: bar center- and endpoints, and UAVs</td>
<td>66</td>
</tr>
<tr>
<td>5.17</td>
<td>Decentralized MPC experimental results showing angular position tracking.</td>
<td>66</td>
</tr>
<tr>
<td>5.18</td>
<td>Decentralized MPC experimental results showing thrust and torque.</td>
<td>67</td>
</tr>
<tr>
<td>5.19</td>
<td>CPU time for decentralized MPC experiment</td>
<td>67</td>
</tr>
<tr>
<td>5.20</td>
<td>Bar centerpoint position error.</td>
<td>68</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Parameter settings for the pendulum simulation.</td>
<td>25</td>
</tr>
<tr>
<td>2.2</td>
<td>Initial conditions for the pendulum simulation.</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Parameter settings for the centralized MPC in the simulation experiments.</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Parameter settings for the offset-free MPC in the simulation experiments.</td>
<td>45</td>
</tr>
<tr>
<td>4.3</td>
<td>CPU and RAM specifications for the workstation used throughout the thesis.</td>
<td>45</td>
</tr>
<tr>
<td>4.4</td>
<td>The six setpoints used in the test of general performance.</td>
<td>46</td>
</tr>
<tr>
<td>4.5</td>
<td>Parameter settings for the MPC used in the experimental validation of the centralized MPC.</td>
<td>51</td>
</tr>
<tr>
<td>4.6</td>
<td>The setpoints used during the experimental validation.</td>
<td>51</td>
</tr>
<tr>
<td>4.7</td>
<td>Parameter settings for the MPC used in the experimental validation of the decentralized MPC.</td>
<td>53</td>
</tr>
<tr>
<td>5.1</td>
<td>Bar centerpoint position error statistics.</td>
<td>69</td>
</tr>
</tbody>
</table>
### List of acronyms and abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CoM</td>
<td>Center of Mass</td>
</tr>
<tr>
<td>DAE</td>
<td>Differential-Algebraic Equation</td>
</tr>
<tr>
<td>DARE</td>
<td>Discrete Algebraic Ricatti Equation</td>
</tr>
<tr>
<td>DOF</td>
<td>Degrees of Freedom</td>
</tr>
<tr>
<td>DP</td>
<td>Dynamic Programming</td>
</tr>
<tr>
<td>FCU</td>
<td>Flight Control Unit</td>
</tr>
<tr>
<td>GGL</td>
<td>Gear–Gupta–Leimkuhler</td>
</tr>
<tr>
<td>JIT</td>
<td>Just-In-Time</td>
</tr>
<tr>
<td>LQ</td>
<td>Linear Quadratic</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear Quadratic Regulator</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear Time-Invariant</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control(ler)</td>
</tr>
<tr>
<td>NLP</td>
<td>Nonlinear Program</td>
</tr>
<tr>
<td>ODE</td>
<td>Ordinary Differential Equation</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional-Integral-Derivative</td>
</tr>
<tr>
<td>QP</td>
<td>Quadratic Program</td>
</tr>
<tr>
<td>RHC</td>
<td>Receding Horizon Control</td>
</tr>
<tr>
<td>RK4</td>
<td>Fourth Order Runge–Kutta</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>ZOH</td>
<td>Zero-Order Hold</td>
</tr>
</tbody>
</table>
xiv | List of acronyms and abbreviations
Chapter 1

Introduction

Collaboration has arguably played a central role in the long history of mankind. While collaboration in prehistoric times was essential for our mere survival, it is now a vital part for maintaining and developing the societies that we have built since. In fact, it might be hard to imagine a single feat of mankind that would be possible without collaboration: Neil Armstrong would probably not have been able to take his famous “small step”, and the Egyptian town of Giza would probably be absent of its iconic landmarks. With this in mind, it is not very surprising that much effort is currently being put into the investigation of collaboration and interaction between robots, so-called multi-agent systems. While cooperation and collaboration between humans probably is something as old as humans themselves, the same thing can not be said about robots. This fact is even reflected in the word “robotic”: when used as an adjective describing a human it is often suggestive of a person that is callous, overly rational, and unable to communicate efficiently with his peers. Fortunately, recent advances within the industry suggest that this association might change in the future; an example is the ABB manufactured robot YuMi® (fig. 1.1) which was introduced in 2015 with the intention of direct human-to-robot collaboration.

Another field that is currently subject to much focus is that of autonomous vehicles, and the goal of making the first fully autonomous car has attracted the attention of both traditional car manufacturers as well as technology companies. Development of autonomous vehicles is, however, not limited to ground vehicles; the prospects of unmanned aerial vehicles (UAVs) is also being considered by many companies. In February this year (2021), NASA landed the autonomous helicopter Ingenuity on Mars, where it successfully performed

* The Oxford English Dictionary gives the following definition: Resembling or suggestive of a robot; mechanical; emotionless [Oxf10].
the first powered flight on another planet [NAS15]. UAVs have also been used in search and rescue missions [QSS+15], and have been suggested for inspection and defect detection of solar panels, bridges and building facades [LYC+17; MH07; RLD+13]. Although these type of autonomous systems are intended to work and operate individually, it becomes clear that some sense of collaboration is needed even in this case. The reason is that multiple agents are likely to share space and other resources and might have objectives that therefore are in conflict with one another. Consider for example the simple case illustrated in Figure 1.2 which involves an agent that is to move from point A to B. This objective on its own does not require any collaboration. However, the addition of a second agent with the individual objective of moving from point C to D would make collaboration between the two agents a requirement if both agents are to be successful in their individual objectives. The ability for robots to collaborate will therefore become increasingly important as they become more integrated into our everyday life (or into the industry for that matter.)

The key to any of the successes in making a system behave autonomously is control theory. Control theory is in loose terms the study of the relation between actuators and system state. A system state is a collection of quantities that describe the system which we are interested in controlling and an actuator is a mediator through which we are able to intervene on the system state. If we, for example, are interested in controlling the temperature of a room, the system state may be chosen to be the current temperature, and the actuator
Figure 1.2: An agent with the objective of travelling from point A to point B (blue), and a second agent with the objective of travelling from point B to point C (green). Even though the goals are individual, collaboration is needed for both objectives to be completed.

could be the knob of a radiator inside the room. The control problem then consists of finding a relation between the current temperature and the position of the radiator knob, such that the room temperature is maintained at a desired temperature.

Traditionally, proportional-integral-derivative (PID) controllers have been the de facto method of control for a variety of systems, including UAVs (see e.g., [PD17a]). The reasoning behind the use of PID controllers is often that they are easy to apply and analyze: for regulation purposes, only the error between actual state and the desired state is considered, and a control signal is given based on the current error (proportional), the sum/integral of the error (integral), and the rate of change of the error (derivative). However, an important downside to PIDs is their inability to deal with system constraints in a straightforward manner. This might limit the type of models that can be used in the controller analysis and design.* In [PD17a], the constraint on the direction of thrust had to be modeled by the inclusion of an artificial attitude inner loop. As some of the system constraints might be safety critical, meaning that a breaking of them might mean system failure, it is arguable that a control scheme with an inherent ability to include such constraints is preferable. A promising control scheme which fulfils this criterion is model predictive control (MPC), which will be treated extensively in this thesis and introduced in the sequel.

The use of MPC also comes with drawbacks. Using an MPC in a pro-

* Although PIDs do not make explicit use of models, it is often desirable to use a model in order to analyze the performance and stability of the resulting closed-loop system. Especially since there commonly exist controller gains that render the system unstable.
cess typically means running an optimization problem at each time instant. Therefore it is not surprising that MPC in its early days was mostly used in the chemical process industry, where engineers often dealt with systems on the slower end of the scale. The slower dynamics meant that the computational load, although time-consuming, was not a limiting issue. Since then, the increase in computational power of computing devices has now made MPC a viable choice even for systems with much faster dynamics: MPC has been proposed and experimentally tested for system such as trucks [LCMW18], and UAVs [CSZ+20]. However, the computational load of MPC unfortunately still remains an issue as model complexity and the number of states increase. Such is the case for the system that will be treated in this thesis. The system is illustrated in Figure 1.3 and consists of two UAVs that collaboratively carry a payload in the form of a bar.

![Figure 1.3: The system treated in this thesis. Two UAVs collaboratively transport a bar that is connected to the UAVs through cables.](image)

The geometrical constraints that the linkage of these three rigid bodies impose, together with the high number of degrees of freedom (DOF) of the cables, make the interaction forces between the bodies complex, and strongly nonlinear. This makes direct applications of MPC impossible for real-time usage. On the other hand, an exclusion of these forces from the model (e.g. treating them as disturbances) might lead to unstable behaviour since the MPC no longer can take into account how the interaction forces will evolve. In particular, the UAVs might take decisions that counter-act each other. This is illustrated in Figure 1.4, where the UAVs efforts to reach a reference position cause unstable
behaviour since they inadvertently give the bar a too high rotational velocity, which then slings both UAVs away from their reference position after having reached it.

![Figure 1.4: Two UAVs with given reference positions: (a) Each UAV apply a force unknowing of the other UAVs decision and its effect on the bar dynamics; (b) The effect is that the bar is given a rotational velocity which slings the UAVs even further from the position in (a).](image)

The goal of this thesis is to evaluate the performance of model predictive controllers on systems with the complex, coupled dynamics often present in collaborative payload transport. Particular efforts are put into making the controllers apt for real-time usage. This means that the model complexity must be decided keeping in mind the trade-off between closed-loop performance and computational load, as previously discussed. The system visualized in Figure 1.3 is chosen since it is relatively simple, while it at the same time captures many of the aforementioned dynamics. Contrast this, for example, with a system where the payload resembles a point mass and hence not is affected by (or causing) any rotational forces: a bar clearly is more resemblant of an arbitrary 6-DOF object. Another important reason for the choice is that the system in question was studied under the framework of geometric control in [Ötä19]. This enables a direct use of some of its results.

### 1.1 Literature review

Payload transport using airborne vehicles was studied already in 1986 in [CK86], this time in the case of helicopters. Recent focus has mainly been on UAVs and, particularly, multirotors. A common denominator for much of the work is to either treat the payload as a disturbance, or to find other
workarounds in order to bypass the need for modeling the complete dynamics of the payload: [TDA+17] used Robust MPC with a disturbance term for capturing model uncertainties and all external forces, including the force generated by the payload; and [GCS17] models a UAV–bar system similar to Figure 1.3 as two UAVs with separate pendulums and use a leader-follower scheme to minimize the expression of the more complex dynamics of the bar. Successful tracking of lateral motion in a single direction is achieved with this approach. Other approaches take into account the full dynamics of the payload: Lee [Lee18] models an arbitrary 6-DOF rigid body payload in a coordinate-free fashion, and constructs a geometric PID-controller that allows asymptotical trajectory tracking of both position and attitude of the payload; Jackson et al. [JHS+20] propose a distributed batch optimization problem for generating trajectories of cable suspended point-masses, and the resulting trajectories (calculated offline) are then followed by means of a velocity-based trajectory tracking controller. The previously mentioned work by Pereira and Dimarogonas [PD17a] introduces, and experimentally verifies, a geometric control law for the system illustrated in Figure 1.3. The control law is proven to be exponentially stable around its equilibrium points and includes integral action to account for parameter uncertainties. Later works expand upon these results: in [PD17b], the control law is expanded with a degree of freedom for adjusting the relative position between the UAVs; in [PRD18], Pereira, Roque, and Dimarogonas explore the effect of system asymmetries and derive appropriate control laws, which are then experimentally verified. Lissandrini et al. [LVR+20] further explore system asymmetries by constructing a decentralized leader-follower MPC scheme which allows for object manipulation by a generic number of robots, now including not only UAVs but also ground robots. The resulting control scheme is experimentally verified for a system consisting of a ground robot and a UAV, both equipped with manipulators, that collaboratively grasp a payload in the form of a bar. Another interesting approach is found in [GTS+19], where an explicit open-loop control force is determined that is able to stabilize an arbitrary 6-DOF object. This force is dependent on a measurement of the contact force experienced by the end-effector from the rigid-body, but has the benefit that no external communication between the agents is needed: the implicit communication by the sensing of contact forces is enough.
1.2 Outline

Chapter 2 introduces relevant background such as MPC, as well as numerical simulation techniques. Section 3.1 introduces the system treated in this thesis and derives the physical models needed for the controllers, which are then introduced in Section 3.2. In Chapter 4, simulation environments and experimental setups are described, and experiments, both numerical and physical, are proposed. Finally, the experimental results are presented in Chapter 5, while concluding remarks are left for Chapter 6.
Introduction
Chapter 2

Background

2.1 Optimization and mathematical programming

It is not difficult to see why optimization might be a topic that is of interest. After all, optimizing is something that we do in our everyday life as well: choosing to take one decision over another is often something that is done after concluding that it, by some reason, is better than the other. In the engineering sciences, where mathematics is the preferred language of description, the formulations and methods for optimizing are of course also of mathematical nature. It should be mentioned that, while some problems can be solved by mere calculus, the non-continuous and condition-based nature of other problems require the use of more sophisticated, near algorithmic approaches. Such approaches have been developed since Fourier’s work in 1827, but arguably the most important contributions have been done after the work on linear programming by e.g., George Dantzig and Leonid Kantorovich [Sch98, p. 90]. With this said, it would be rather unfair to say that optimization within control theory is something that is really new. In fact, we can see traces of this already in the formulation of the brachistochrone* problem by Johann Bernoulli in 1696 [Apa12; Mir96]:

\[ \text{Given two points, } P_1 \text{ and } P_2 \text{ in a vertical plane, such that } P_2 \text{ is below } P_1, \text{ find the shape of a wire joining the points such that a bead falling along the wire under the influence of gravity and without friction, will travel from } P_1 \text{ to } P_2 \text{ in least time.} \]

The solution to this problem is well known to those who are familiar with analytical mechanics, and it is often found through calculus of variations. The

* This seemingly cryptical name is in fact nothing more than a direct translation to greek of the words shortest (brachistos) and time (chronos).
result is the so called cycloid (fig. 2.1) given by the parametric description

\[
\begin{align*}
    x &= R (\phi - \sin \phi), \\
    y &= -R (1 - \cos \phi),
\end{align*}
\]

which interestingly corresponds to the trajectory that is made by a fixed point on a circle with radius \(R\), as this circle rotates an angle \(\phi\) over a flat surface without slipping.

![Figure 2.1: The cycloid as defined in Equation (2.1).](image)

**Example 2.1.1: Numerical solution to the brachistochrone problem**

The brachistochrone problem can be solved numerically as a nonlinear program (NLP) by first performing a discretization. Let’s consider the continuous case in the first place. We know that the speed will be given by

\[
v = \sqrt{-2gy},
\]

and that the distance traveled in an infinitesimal amount of time is given by

\[
s = \sqrt{dx^2 + dy^2} = \sqrt{1 + (dy/dx)^2}dx.
\]

The total time \(T\) taken to travel from an initial point \((x_1, y_1)\) to a goal point \((x_2, y_2)\) is thereby

\[
T = \int_{x_1}^{x_2} \sqrt{\frac{1 + (dy/dx)^2}{-2gy}} \, dx,
\]
and the optimization problem can hence be formulated as

$$\min \int_{x_1}^{x_2} \frac{1 + (dy/dx)^2}{-y} \, dx . \quad (2.5)$$

Finally, after a discretization of the problem into $N$ points, this becomes the minimization of a sum

$$\min \sum_{n=1}^{N-1} \frac{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2}{y_n}$$

subject to

$$(x_0, y_0) = P_1$$
$$(x_{N-1}, y_{N-1}) = P_2$$

$$y_i < 0, \; i = 1, \ldots, N - 1 \quad (2.6)$$

The results of a solution to this problem, with $N = 20$ and a starting guess as the straight line between $P_1$ and $P_2$ are shown in Figure 2.2, where the analytical solution has been included for comparison. The solution was implemented in CasADi*, using the NLP solver ipopt.

Figure 2.2: Numerical solution to the brachistochrone problem after reformulating it as a discrete NLP.

2.1.1 Dynamic Programming

Dynamic programming is an optimization strategy which originated in the late 1950s. Essential to the strategy is the principle of optimality, coined by the dynamic programming originator Richard Bellman:

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. [Bel57, p. 83]

The important intuition provided by this principle is that if an optimization problem can be divided into stages, then each stage can be optimized over individually and the optimal policy be found as the combination of all these sub-solutions.

An area of problems in which the DP formulation comes naturally is the area of multi-stage decision processes. A classical example of this is the knapsack problem, in which we are to choose from a range of items with distinct characteristics in order to maximize value, under the constraint of keeping weight and/or size under a certain given threshold. A stage is in this case considered to be each time an item is chosen. More relevant to this thesis is of course the case of controlling a dynamic system. In this case we often seek to minimize the deviation from a certain regulation point; the decision to be made is what control signal should be used, and each stage corresponds to a certain moment in time.

2.1.1 The Bellman Equation

Considering the case of a dynamic system that evolves according to

\[ x_{k+1} = f(x_k, u_k), \]  

the optimization problem

\[
\min_u \sum_{k=1}^{N-1} v_k(x_k, u_k) + v_N(x)
\]

subject to \( x_{k+1} = f(x_k, u_k) \) \( k = 0, 1, \ldots N \)

\( u_k \in \mathcal{U} \) \( k = 0, 1, \ldots, N - 1 \)

whose solution, if existent, will be a control policy on the form

\[ \{u_k^*(x_k)\}_{k=0}^{N-1}, \]  

(2.8)
can be solved by application of the recursive formula known as the Bellman equation

\[
V_N(x) = v_N(x) \\
V_k(x) = \min_{u \in U} \{v_k(x,u) + V_{k+1}(f(x,u))\}, \quad 0 \leq k < N.
\]

(2.9)

The optimal value of the objective function \(V(\cdot)\) is then given by

\[
V_0(x_0),
\]

(2.10)

and the optimal control policy by the optimal control signal at each stage in Equation (2.9), i.e.,

\[
u_k^* = \arg \min_{u \in U} \{v_k(x,u) + V_{k+1}(f(x,u))\}.
\]

(2.11)

For a more detailed application, see Example 2.2.1.

### 2.2 Model Predictive Control

MPC, is a control paradigm with roots in optimal control [RMD19]. As the name suggests, it uses predictions generated by a model as a basis for generating appropriate control signals. To be more precise, MPC takes into account the current state (or an estimation of the current state), compares it to a desired state, and uses a system model to find the control policy which minimizes future deviations from the desired state. This minimization is done with respect to a cost (or objective) function \(J(\cdot)\), which most often is a quadratic function of the system state and inputs \(x_t, u_t\), i.e., on the form

\[
J(x,u) = \sum_{t=0}^{N-1} \left[ x_t^T Q x_t + u_t^T R u_t \right] + x_N^T Q_N x_N,
\]

for matrices \(Q, R, \) and \(Q_N^*\). The parameter \(N\) is often referred to as the horizon and decides how far into the future the model predictions should be made. The resulting output from the MPC is then a set of (optimal) control signals

\[u^* = \{u_0^*, u_1^*, \ldots, u_{N-1}^*\};\]

however, the convention is to run the MPC at each sampling instance, and to only send in the first signal \(u_0^*\) to the actual system. This practice is known as receding horizon control (RHC).

* The reasoning behind the matrix \(Q_N\) will become clear in the following sections.
2.2.1 A general MPC problem

Consider the linear discrete time model given by

\[ x_{k+1} = Ax_k + Bu_k \] (2.12)

where \( x_k \in \mathbb{R}^n \), and \( u_k \in \mathbb{R}^m \) for all \( k \), denote the system state, the control input, and the system output, respectively, and \( A, B \) are matrices of appropriate dimensions.

The MPC problem, in its most basic form, is then given as an optimization problem

\[
\begin{align*}
\min_{k=0}^{N-1} & \quad \sum_{k=0}^{N-1} \left[ x_k^T Q x_k + u_k^T R u_k \right] + x_N^T Q_N x_N \\
\text{subject to} & \quad x_{k+1} = Ax_k + Bu_k \\
& \quad x_k \in \mathcal{X} \\
& \quad u_k \in \mathcal{U},
\end{align*}
\] (2.13)

where \( Q, R, \) and \( Q_N \) are matrices that, at least for now, can be considered as design parameters, and \( \mathcal{X} \) and \( \mathcal{U} \) represent any eventual constraints on \( x \) and \( u \), respectively. The possibility of having constraints as part of the control procedure is one of the most important benefits of the MPC formulation and often why it's considered over other approaches, such as the PID controller, or the even more closely related linear quadratic (LQ) controller. As we shall see, however, it also comes with several disadvantages; whereas the more traditional approaches often result in equations from where the solution can be found by algebraic means, the optimization formulation of MPC means that numerical solvers are needed, except in the most simple cases. Further complications also arise due to the fact that the existence of a solution can not be guaranteed for all combinations of matrices \( Q, R, Q_N \) and constraint sets \( \mathcal{X} \) and \( \mathcal{U} \). A perhaps even more severe limitation is that the optimality of a solution, does not necessarily imply stability [Kal60]. It is therefore of high priority to find conditions such that, when fulfilled, can guarantee stability. This will be the topic of the sequel.

**Example 2.2.1: Explicit MPC solution for a simple system using dynamic programming (DP)**

Let \( x_{k+1} = f(x_k, u_k) := x_k + u_k, |u_k| \leq 1 \), define the system dynamics. Furthermore, let \( Q = R = Q_N = 1 \) and consider a horizon of \( N = 2 \). The
optimization problem becomes
\[
\min_{k=1}^{N-1} \left[ x_k^2 + u_k^2 \right] + x_N^2
\]
subject to \[ x_{k+1} = x_k + u_k \quad k = 0, 1, \ldots, N \]
\[ |u_k| \leq 1 \quad k = 0, 1, \ldots, N - 1 \]

Following the DP method, we identify
\[ v_N(x) = x^2, \]
and can calculate \( v_{N-1} \) as follows
\[
v_{N-1}(x) = \min \left\{ x^2 + u^2 + (x + u)^2 \right\}
= \min \left\{ 2x^2 + 2xu + 2u^2 \right\} = \left\{ u^* = -\text{sat} \frac{1}{2} x \right\}
= \begin{cases} 
\frac{3}{2}x^2, & \text{if } -2 \leq x \leq 2 \\
2x^2 + 2x + 2, & \text{if } x < -2 \\
2x^2 - 2x + 2, & \text{if } x > 2
\end{cases}
\]

The constraint on the input has in this case given us three distinct expressions for \( v_{N-1} \) which all need to be considered in the calculation of \( v_{N-2} \).

First, consider the case where \( x \in [-2, 2] \)
\[
v_{N-2}(x) = \min \left\{ x^2 + u^2 + \frac{3}{2} (x + u)^2 \right\}
= \min \left\{ \frac{5}{2}x^2 + \frac{3}{2} xu + \frac{5}{2} u^2 \right\} = \left\{ u^* = -\text{sat} \frac{3}{5} x \right\}
= \frac{8}{5}x^2.
\]

Finally, the two cases where \( x \notin [-2, 2] \) give
\[
v_{N-2}(x) = \min \left\{ 3u^2 + 2(2x \pm 1)u + 3x^2 \pm 2x + 2 \right\}
\]
which, if \( u \) were unbounded, would be minimized by
\[ u^{**} = -\frac{1}{3}(2x \pm 1). \]
However, on both of the intervals, \( u^* \not\in [-1, 1] \) and thereby the actual optimal values are

\[
u^* = \begin{cases} 
-1, & \text{if } x > 2 \\
1, & \text{if } x < -2
\end{cases}
\]

and in conclusion, the control law can be summarized as

\[
u^* = -\text{sat} \left\lfloor \frac{3}{5}x \right\rfloor.
\]

Note that, MPC finds optimal values by quadratic programs (QPs) and not DPs, but in this simple case the solutions coincide. Figure 2.3 shows a numerical solution of the problem where an MPC has been used to find the optimal control signal \( u^* \) for a range of \( x \)-values between \(-4\) and \(4\).

![Figure 2.3: MPC implementation and resulting optimal control values for a simple system with an horizon of 2. The graph follows the theoretically derived solution \( u^* = -\text{sat} \frac{3}{5}x \) well.](image-url)

### 2.2.2 Linear Quadratic Control

To gain deeper understanding of MPC it turns out to be beneficial to first study the LQ control problem, which differs from MPC in that

1. It requires the model to be strictly linear
2. The cost function must be quadratic
3. It does not (inherently) deal with constraints

The infinite horizon LQ control problem is formulated as the following
optimization problem

\[
\min \sum_{k=0}^{\infty} [x_k^T Q x_k + u_k^T R u_k]
\]

\[
s.t \quad x_{k+1} = A x_k + B u_k \\
y_k = C x_k.
\]

(2.14)

in which we now also assume that both matrices \( Q \) and \( R \) are symmetric, and furthermore that \( R \) is positive definite, \( Q \) is positive semi-definite and such that \( (Q^{1/2}, A) \) is detectable. This problem was formulated and solved by Kalman [Kal60], and has a solution in the form of a linear feedback control law \( u_k = -K x_k \), where

\[
K = (R + B^T P B)^{-1} B^T P A
\]

(2.15)

for a semi-definite \( P \) which is found by solving the discrete algebraic Riccati equation\(^*\)

\[
\begin{cases}
P_{t-1} = Q + A^T P_t A - A^T P_t B (R + B^T P_t B)^{-1} B^T P_t A \\
P_t = P_{t-1} = P_t
\end{cases}
\]

(2.16)

Essential to our case, however, is that the resulting control law is proved to render the system stable, i.e., optimality does imply stability\(^\dagger\).

**Example 2.2.2: Infinite horizon linear quadratic regulator (LQR).**

Consider the optimization problem defined in Example 2.2.1. By removing the constraint \( |u_k| < 1 \) and allowing \( u_k \) to assume any value in \( \mathbb{R} \), the problem can be reformulated as an LQR problem

\[
\min \sum_{k=1}^{N-1} [x_k^2 + u_k^2] + x_N^2
\]

\[
s.t \quad x_{k+1} = x_k + u_k \quad k = 0, 1, \ldots, N
\]

Starting by considering a horizon of \( N = 2 \), we arrive at an optimal control policy

\[
{u_0^*, u_1^*} = \{x_0 P_1/(1 + P_1), x_1 P_2/(1 + P_2)\} = \{3x_0/5, x_1/2\}
\]

(2.17)

\(^*\) For details see e.g. [RMD19] \(^\dagger\) Note that this holds only in the infinite horizon case and not for LQ control in general.
where $P_t$ for the finite horizon case has been found recursively through the equation

$$
\begin{align*}
P_{t-1} &= Q + A^T P_t A - A^T P_t B (R + B^T P_t B)^{-1} B^T P_t A, \\
N &= Q_N.
\end{align*}
$$

(2.18)

Note the similarity between this solution and the one in Example 2.2.1, and especially how the two solutions would be identical provided that $x_0 \in [-1, 1]$ and that the LQR would be run in a RHC fashion.

Proceeding to the infinite horizon case, the optimal control is found after solving the discrete algebraic ricatti equation (DARE) given by Equation (2.16), which gives $P \approx 1.618$ and consequently $u_k^* \approx -0.618 x_k$ for all $k$.

In Figure 2.4, the closed-loop response of the infinite horizon LQR has been plotted against the derived finite horizon LQR for the case $x_0 = 1$. The case $N = 1^*$ has been included for comparison. The control law seems to converge quickly towards the infinite horizon solution as $N$ increases.

Figure 2.4: Closed-loop response under finite horizon LQR, and receding, finite horizon LQR. The rightmost graph shows the difference from the infinite horizon solution at each time-step.

2.2.3 MPC stability

Rewriting the cost function in Equation (2.14) on the form

$$
\left( \sum_{k=0}^{N-1} + \sum_{k=N}^{\infty} \right) \left[ x_k^T Q x_k + u_k^T R u_k \right]
$$

(2.19)

* This gives the control law $u_k = x_k / 2$. 
we utilize the fact that the infinite sum, under the absence of constraints, along with previous assumptions on matrices $A$, $B$, $Q$, and $R$, has an explicit minimal value given through the solution of the infinite horizon DARE. In other words, the equivalence

$$
\min \sum_{k=0}^{\infty} [x_k^T Q x_k + u_k^T R u_k] = \min \sum_{k=0}^{N-1} [x_k^T Q x_k + u_k^T R u_k] + x_N^T P x_N
$$

holds, if $P$ is the solution to Equation (2.16). A convenient way to achieve the same stability guarantees as for the infinite horizon LQR in the MPC case would hence be to set $Q_N = P$ in Equation (2.13). However, nonlinearities and/or constraints in the MPC case might break the aforementioned equivalence (eq. (2.20)). The solution is then often to force the system to a suitable region near the origin before time-step $N$, where all constraints are automatically fulfilled and the system is approximately linear; then Equation (2.20) holds, at least approximately [MRRS00, pp. 795–796]. This is generally done by adding a terminal constraint set $\mathcal{X}_N$ and the constraint

$$
x_N \in \mathcal{X}_N
$$

(2.21)

to the optimal control problem formulation. The set $\mathcal{X}_N$ can favourably be chosen as the maximal output admissible set in order to maximize the set of starting states from which the system remains feasible at all times.

### 2.2.4 Offset-free MPC

The fact that a model is needed in MPC, suggests that the performance of the MPC will depend on the quality and precision of the model. In fact, a mismatch in the model is likely to lead to static errors under reference tracking [BM02]. A way to combat this is offset-free MPC, in which both state and disturbance are modeled, estimated, and fed into the controller, resulting in a control scheme as illustrated in Figure 2.5.

Denoting the disturbance $d$, the reference $r$, and letting $\hat{\cdot}$ denote the estimate of a state ($\hat{\cdot}$), the optimal control problem becomes a slightly modified
Figure 2.5: Blockscheme illustrating an offset-free MPC approach.

version of Equation (2.13):

\[
\min \sum_{k=0}^{N-1} \left[ \Delta x_k^T Q \Delta x_k + \Delta u_k^T R \Delta u_k \right] + \Delta x_N^T Q_N \Delta x_N \\
\text{subject to } \begin{align*}
x_{k+1} &= A x_k + B u_k + B_d \hat{d} \\
y_k &= C x_k \\
x_k &\in \mathcal{X} \\
u_k &\in \mathcal{U} \\
dot{\hat{x}} &= A \tilde{x} + B \tilde{u} + B_d \hat{d} \\
r &= C \tilde{x} \\
\tilde{d} &= \hat{d} \\
\Delta(\cdot)_k &= (\cdot)_k - (\cdot) \\
\Delta x_0 &= \hat{x}_0 - \tilde{x}
\end{align*}
\] (2.22)

Example 2.2.3: Offset-free MPC for height tracking.
Consider the system

\[
\ddot{x} = \frac{u}{m} - g
\] (2.23)

corresponding to a body of mass \( m \) that is affected by a gravitational force \( mg \) and a force \( u \) which can be considered as an input. The system can be transformed into a first order ODE on the form

\[
\begin{cases}
\dot{x}_1 = x_2 \\
\dot{x}_2 = \frac{u}{m} - g
\end{cases}
\] (2.24)

In order to apply a linear MPC for position reference tracking, the system is linearized around \( x_{eq} = (x_0, 0) \), \( u_{eq} = mg \), resulting in the linear time-invariant (LTI) system

\[
\frac{d \Delta x}{dr} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \Delta x + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \Delta u,
\] (2.25)
where $\Delta(\cdot) = (\cdot) - (\cdot)_{eq}$. Finally, a zero-order hold (ZOH) discretization [Ast97] with a sampling time of $h$ yields a discrete time LTI system of the form

$$
\Delta x_{k+1} = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \Delta x_k + \begin{bmatrix} h/2 \\ 1/m \end{bmatrix} \Delta u_k, \quad A
$$

where we now have made explicit the measurement of the position by defining $C$. Now to the real issue: suppose that our model assumes a mass of $m_{est}$. A static error is then present under the control by an MPC unless $m_{est} = m$ (fig. 2.6).

To get rid of the static error we can generate a disturbance model, where we include a disturbance signal $d$ which has the purpose of capturing any unincorporated dynamics of the system. Consequently, we define a new model as follows

$$
\Delta x_{k+1} = A \Delta x_k + B \Delta u_k + B_d d_k. \quad (2.27)
$$

How $B_d$ is chosen is not crucial for successful disturbance elimination [PR03, pp. 430–431], but since the disturbance in this case is related to a force (more force than estimated, or less force than estimated) it is reasonable to believe that
$B_d$ should be parallel to $B^*$. If we assume a constant disturbance $d_{k+1} = d_k$, Equation (2.27) can be reformulated as an augmented system.

\[
\begin{align*}
\xi_{k+1} := \begin{bmatrix} \Delta x_{k+1} \\ d_{k+1} \end{bmatrix} &= \begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ d_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \Delta u_k \\
&=: A_{\text{aug}} \xi_k + B_{\text{aug}} \Delta u_k \\
y_k &= \begin{bmatrix} C & 0 \end{bmatrix} \xi_k
\end{align*}
\]

(2.28)

This augmented system is observable under the condition that [MBM09]

\[
\text{rank } \Theta_{\text{aug}} = \text{rank } \begin{bmatrix} A - I & B_d \\ C & 0 \end{bmatrix} = n + n_d
\]

(2.29)

where $n_d$ is the number of dimensions of the disturbance signal. Our particular system has this property, and hence it is possible to design an estimator on the form

\[
\hat{\xi}_{k+1} = A_{\text{aug}} \hat{\xi}_k + B_{\text{aug}} \Delta u_k + \begin{bmatrix} K_x \\ K_d \end{bmatrix} (y_k - \hat{y}_k)
\]

(2.30)

Setting up the MPC problem as in Equation (2.22), the stationary errors seen in Figure 2.6 are effectively eliminated (fig. 2.7).

![Figure 2.7: Results using offset-free MPC, where the state and disturbances are estimated and used in the optimization process.](image)

Note that the estimator in Equation (2.30) could favourably be replaced by a Kalman filter if needed.

*Parallel in the algebraic sense of the word, so that $B_d = cB$ for some $c \in \mathbb{R}$. In the following example we have chosen $B_d = m_{\text{est}}B$. 


## 2.3 Numerical simulation

The numerical practices for solving ordinary differential equations (ODEs) are numerous, and will typically weigh numerical precision and stability against computational complexity and scalability. While the study of these numerical techniques is beyond the scope of this thesis, an area that needs to be covered is differential-algebraic equations (DAEs). Simply put, a DAE can be seen as an ODE with algebraic constraints, meaning that it includes variables for which we have no explicit time evolution.* DAEs are particularly common within multibody dynamics since any linkage between two (or more) rigid bodies introduces geometrical constraints.

A problem with DAEs, as compared to ODEs, is that many commercially available solvers do not allow for the direct solving of these. Fortunately, there are methods to reduce DAEs to ODEs, or at least, methods of transforming these into formulations that are accepted by the solver in question. This is generally done by differentiation of the algebraic constraints. A highly relevant feature of any DAE in this case is the index. The index of a DAE is, more or less, the number of differentiations needed to transform it into ODE form. While index-reduced DAEs have the same analytical solutions, the numerical precision of the formulations may vary. In this thesis the Gear–Gupta–Leimkuhler (GGL) formulation [GLG85] is considered. The GGL formulation has the advantage that both position and velocity level constraints are fulfilled at each time step, but at the cost of having to add additional Lagrange multipliers [SL15]. Furthermore, the formulation results in an index-2 DAE, which requires solvers that are able to handle this [EF98]. For a practical example and comparison of several techniques, see the following example.

**Example 2.3.1: Simulating a DAE formulation of a pendulum**

Consider the pendulum consisting of a point mass of mass $m$ attached to a rigid, massless link of length $L$. Defining $x = (x, y)$ as the position in a Cartesian coordinate system with $(0, 0)$ corresponding to the pendulum’s point of attachment and $v = (u, v)$ as the velocity of the point mass. The algebraic constraint reads

$$g(x) = \| x \| - L = 0,$$

and the index-3 DAE characterizing the system is

$$\begin{align*}
\dot{x} &= v \\
m \ddot{v} + G^T \lambda &= (0, -mg) \\
g(x) &= 0,
\end{align*}$$

* In fact, an ODE is a special form of DAE.
with $g : \mathbb{R}^2 \to \mathbb{R}$ as defined previously, $G = \partial g / \partial x$, and $g \in \mathbb{R}$ denoting the gravitational constant. The constraint Jacobian $G$ is calculated as follows

$$
\frac{\partial g}{\partial x} = \frac{\partial}{\partial x} \|x\| = \frac{\partial}{\partial x} \sqrt{x^T x} = \frac{1}{2L} \frac{\partial}{\partial x} (x^T x) = \frac{1}{L} x^T, \quad (2.33)
$$

and consequently Equation (2.32) can be expanded to obtain

$$
\dot{x} = v \quad (2.34)
$$

$$
m\ddot{v} + \frac{x}{L} \lambda = (0, -mg) \quad (2.35)
$$

$$
g(x) = \|x\| - L = 0. \quad (2.36)
$$

This index-3 DAE can be reduced to different degrees, resulting in DAEs that can be solved in e.g. SUNDIALS’ IDAS*. To reduce this index it is first necessary to calculate the time derivatives of the constraint $g(\cdot)$. We find

$$
\frac{dg}{dt} = \frac{x\dot{x} + y\dot{y}}{L} = \frac{1}{L} (xu + yv), \quad (2.37)
$$

$$
\frac{d^2g}{dt^2} = \frac{1}{L} \left( x\ddot{u} + y\ddot{v} + u^2 + v^2 \right) = \frac{1}{L} \left( -\frac{L}{m} yg + u^2 + v^2 \right), \quad (2.38)
$$

and

$$
\frac{d^3g}{dt^3} = \frac{1}{L} \left[ -\lambda \frac{L}{m} - 3vg - \lambda \frac{2}{m} \left( \frac{ux + vy}{L} \right) \right], \quad (2.39)
$$

where the term containing $ux + vy$ in the last equation, alternatively, could be crossed out because of the constraint $dg/dt = 0$. However, since this constraint is not necessarily fulfilled at all times during simulation, it may be better to keep this term.

We now consider the three different formulations stated below, each of different index.

**Index-0 DAE**

An expression for the time evolution of the Lagrange multiplier $\lambda$ can be extracted from Equation (2.39), yielding an index-0 DAE

$$
\dot{x} = v
$$

$$
m\ddot{v} + \frac{x}{L} \lambda = (0, -mg) \quad \text{(Index-0 DAE)}
$$

$$
\dot{\lambda} = -\frac{m}{L} \left[ 3vg + \lambda \frac{2}{mL (ux + vy)} \right].
$$

or in other words, an ODE which can be directly simulated.

* HBG+05.
Index-1 DAE

In this formulation, a single algebraic constraint is used, and $\lambda$ is considered to be an algebraic variable instead of a state variable as in the index-0 formulation. The resulting equation system is

$$\begin{align*}
\dot{x} &= v \\
m\dot{v} + \frac{x}{L}\lambda &= (0, -mg) \\
\frac{d^2g}{dt^2} &= 0.
\end{align*}$$

(Index-1 DAE)

Stabilized index-2 DAE (GGL)

In this stabilized variant, a second Lagrange multiplier $\mu$ is used as an algebraic variable to also enforce the positional constraint $g(x) = 0$. The resulting equation system becomes

$$\begin{align*}
\dot{x} &= v + \frac{x}{L}\mu \\
m\dot{v} + \frac{x}{L}\lambda &= (0, -mg) \\
g(x) &= 0 \\
\frac{dg}{dt} &= 0.
\end{align*}$$

(Index-2 DAE)

This solution of this formulation is equivalent to that of the original DAE, provided that $\mu = 0$. The variable $\mu$ could thus be seen as a correction factor that enforces the constraint $g(\cdot)$, and that ideally should be zero.

The different formulations in Equations (Index-0 DAE) to (Index-2 DAE) have been solved using SUNDIALS’ IDAS through its CasADi interface, and the results are given in Figure 2.8, where both trajectories and errors in pendulum length are graphed. The GGL variant clearly demonstrates its superiority in terms of not breaking the pendulum length constraint. For parameter settings and initial conditions, see Tables 2.1 and 2.2.

Table 2.1: Parameter settings for the pendulum simulation.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$g$</th>
<th>$L$</th>
<th>$dt$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>9.8066</td>
<td>1.0</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Figure 2.8: Resulting pendulum trajectory (upper) with corresponding squared length difference $x^2 + y^2 - L^2$ (lower) for numerical simulations using a number of different formulations of the same DAE. The superiority of the GGL-formulation (right) is evident since the squared length difference is on the order of $10^{-10}$, compared to $10^{-1}$ for the other two cases.

Table 2.2: Initial conditions for the pendulum simulation.

<table>
<thead>
<tr>
<th>$x_0$</th>
<th>$y_0$</th>
<th>$u_0$</th>
<th>$v_0$</th>
<th>$z_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1/\sqrt{2}$</td>
<td>$1/\sqrt{2}$</td>
<td>0</td>
<td>0</td>
<td>$\sqrt{2}mg$</td>
</tr>
</tbody>
</table>
Chapter 3

System analysis and controller design

The purpose of this chapter is to give an overview of the system that is treated in this thesis (Figure 1.3), and to derive suitable model predictive controllers. Starting with a model for a single UAV, the full UAV–bar model is derived, after which a set of equilibrium points for the system is found. Furthermore, with the intention of decentralizing the controller structure while maintaining some of the cable dynamics, a model of a single UAV with payloads is developed. Finally, the MPCs are designed by slight adaptations of the models.

3.1 UAVs/Quadrotors

3.1.1 Modeling of a single UAV

Let \{\mathcal{E}\} and \{\mathcal{B}\} denote an inertial frame, and UAV body frame, respectively (fig. 3.1), with \{e_x, e_y, e_z\} and \{b_x, b_y, b_z\} being the basis vectors of each frame. Also, let the frame \{\mathcal{B}\} be attached and kept fixed to the UAV center of mass (CoM). The frames are relatable through a rotation matrix \( R \) and a translation by \( p \), with \( p \) thereby corresponding to the current position of the UAV. The rotation matrix \( R \) can be represented by numerous forms, but in this work we will consider Euler angles under the ZYX-convention, and \( R \) therefore takes the form

\[
R(\phi, \theta, \psi) = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix}
  c_{\psi}c_{\theta} & c_{\psi}s_{\theta}s_{\phi} - s_{\psi}c_{\phi} & c_{\psi}s_{\theta}c_{\phi} + s_{\psi}s_{\phi} \\
  s_{\psi}c_{\theta} & s_{\psi}s_{\theta}s_{\phi} + c_{\psi}c_{\phi} & s_{\psi}s_{\theta}c_{\phi} - c_{\psi}s_{\phi} \\
  -s_{\theta} & c_{\theta}c_{\phi} & c_{\theta}s_{\phi}
\end{bmatrix},
\]

(3.1)
where $R_i, i \in \{x, y, z\}$ is the elementary rotation matrix around axis $i$, as defined in [SSVO09, p. 42] and the abbreviated forms

$$s_X = \sin X$$
$$c_X = \cos X$$

have been used. The angles $\phi$, $\theta$, and $\psi$ will in the sequel be referred to as roll, pitch, and yaw, respectively. Furthermore, we consider $m$ as the mass of the UAV, and $J$ as its associated inertial matrix.

![Figure 3.1: A single UAV with body frame $\{B\}$ located at a position $p$ with respect to the inertial frame $\{E\}$.](image)

Although the forces and moments that the UAV is able to generate is an effect of the combination of the thrusts generated by each propeller, a frequently used technique is to combine these thrusts into a single thrust in the direction of $b_z$ (here $F_t$), and a torque vector (here $\tau_B$)*. This technique is motivated by that each pair $(F_t, \tau_B)$, uniquely defines the thrust of each propeller [LLM10, p. 5421].

After denoting $\dot{\Theta} = (\dot{\phi}, \dot{\theta}, \dot{\psi})$ as the Euler angle rates, and $\omega_B = (p, q, r)$ as the angular velocities in the body frame, the equations of motion of the system can be derived using the Newton-Euler formulation, and result in the

* See e.g. [MKC12, p. 21; LLM10]
following set of equations

\[ \dot{p} = v \]  

\[ \ddot{p} = \frac{1}{m} (Re_z F_t - mge_z) \]  

\[ \dot{\Theta} = H \omega_\mathcal{B} \]  

\[ \dot{\omega}_\mathcal{B} = J^{-1} (\tau_\mathcal{B} - \mathcal{S}(\omega_\mathcal{B}) J \omega_\mathcal{B}) \]  

where \( \mathcal{S} : \mathbb{R}^3 \to \mathbb{R}^{3 \times 3} \) is the skew-symmetric matrix operator such that

\[ \mathcal{S}(X)Y = X \times Y. \]  

with \( \times \) denoting the vector cross-product, and \( H \) is the matrix that relates the Euler angular rates to the angular velocities in the body frame and is given by

\[
H = \begin{bmatrix}
    e_x & R_x(\phi)^{-1} e_y & R_x(\phi)^{-1} R_y(\theta)^{-1} e_z \\
1 & s_\phi t_\theta & c_\phi t_\theta \\
0 & c_\phi & -s_\phi \\
0 & s_\phi / c_\theta & c_\phi / c_\theta
\end{bmatrix}^{-1}
\]  

where the relationship \( t_X = \tan X \) has been added to list of trigonometrical abbreviations. Finally, the system can be written in scalar form as the ODE

\[
\begin{align*}
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{z} &= v_z \\
\dot{v}_x &= F_t m^{-1} (s_\phi s_\psi + c_\phi c_\psi s_\theta) \\
\dot{v}_y &= F_t m^{-1} (c_\phi s_\psi s_\theta - c_\psi s_\theta) \\
\dot{v}_z &= F_t m^{-1} (c_\phi c_\theta) - g \\
\dot{\phi} &= p + r (c_\phi t_\theta) + q (s_\phi t_\theta) \\
\dot{\theta} &= q (c_\phi) - r (s_\phi) \\
\dot{\psi} &= [r (c_\phi) + q (s_\phi)] / c_\theta \\
\dot{p} &= J_x^{-1} [(J_y - J_z) r q + \tau_x] \\
\dot{q} &= J_y^{-1} [(J_z - J_x) p r + \tau_y] \\
\dot{r} &= J_z^{-1} [(J_x - J_y) p q + \tau_z]
\end{align*}
\]  

where \( \tau_\mathcal{B} := (\tau_x, \tau_y, \tau_z) \), and \( J \) has been assumed to be a diagonal matrix \( J = \text{diag}(J_x, J_y, J_z) \).
3.1.2 Modeling of a UAV–bar system

We now consider a system where a rigid bar is tethered to two UAVs by cables, as illustrated in Figure 3.2. Each UAV is given a corresponding body frame \( \{B_i\} \) that is attached to the UAV CoM, and to where it is assumed that the cables are attached. The bar, however, is chosen to be modeled solely with respect to the inertial frame \( \{E\} \).

![Figure 3.2: UAV–bar system.](image)

Each cable is of length \( l_1 \) and \( l_2 \), respectively, where the subindex is given depending on to which UAV the cable is directly attached to: \( l_1 \) is the length of the cable attached to UAV 1, etc. The bar also comes with a number of quantities that need to be defined: we consider \( m_b \) to be the mass of the bar, and \( d_i \) to be the distance between the bar CoM and the end of the bar to which cable \( i \) is attached. Without loss of generality, \( d_1 \) is chosen to be negative, and thus the total length of the bar is \( (d_2 - d_1) \).

The cables constitute the media through which forces between the UAVs and the bar are mediated, resulting in internal forces \( T_1 \) and \( T_2 \). It becomes clear that the directions of the cables

\[
    n_i = \frac{p_i - (p_b + d_i n_b)}{l_i}, \quad i = 1, 2
\]

are important. However, as \( n_i \) is completely decided by other states, it is introduced solely for matters of simplification.
The same derivation as for the single UAV case results in the following set of equations

\[
\begin{align*}
\dot{p}_i &= v_i, & i = 1, 2, b \\
\ddot{p}_i &= m_i^{-1} \left( (F_i R_{\theta_i} - m_i g I) e_z - T_i n_i \right), & i = 1, 2 \\
\dot{\omega}_{B,i} &= J_i^{-1} \left( \mathbf{\tau}_{B,i} - \mathbf{\omega}_{B,i} J_i \mathbf{\omega}_{B,i} \right), & i = 1, 2 \\
\dot{\omega}_b &= m_b^{-1} \left( T_1 n_1 + T_2 n_2 - m_b g \right) \\
\dot{\Theta}_b &= H_{\Theta_b} \omega_b \\
\dot{\omega}_b &= J_b^{-1} S(n_b)(d_1 T_1 n_1 + d_2 T_2 n_2)
\end{align*}
\]

where \( R_{\theta_i}, H_{\theta_i} \) are found by making the substitutions

\[
\phi \leftarrow \phi_i, \quad \theta \leftarrow \theta_i, \quad \psi \leftarrow \psi_i
\]

in Equation (3.1) and Equation (3.7), respectively. The matrix \( H^{(\xi)} \), relates angular velocities in the inertial frame to Euler angles rates and has the form

\[
H^{(\xi)} = \begin{bmatrix}
R_x(\psi) R_y(\theta) e_x & R_z(\psi) e_y & e_z
\end{bmatrix}^{-1}
\]

\[
= \begin{bmatrix}
c \psi / c \theta & s \psi / c \theta & 0 \\
-s \psi & c \psi & 0 \\
c \psi t \theta & s \psi t \theta & 1
\end{bmatrix}.
\]

Equation (3.16) is valid under the assumption that the connections to the UAVs only imply positional constraints, and \( J_b \) can in this case be chosen to be a scalar, corresponding to the moment of inertia in the axes perpendicular to the bar symmetry line (\( n_b \)).

For purposes of reducing the model complexity, each cable is modeled as a massless, rigid, and rotational link, which means that the following constraint

\[
\| p_i - (p_b + d_i n_b) \| = l_i,
\]

must be fulfilled, and the forces \( T_1, T_2 \) take the form [Ótá19, p. 175]

\[
\begin{bmatrix}
T_1 \\
T_2
\end{bmatrix} = M^{-1} \Lambda,
\]

where

\[
M = \begin{bmatrix}
\frac{m_b + m_1}{m_1} & \frac{\mathbf{n}_1 \cdot \mathbf{n}_2}{m_2} \\
\frac{m_1}{\mathbf{n}_1 \cdot \mathbf{n}_2} & \frac{m_b + m_2}{m_2}
\end{bmatrix} + \frac{m_b}{J_b} \begin{bmatrix}
(d_1 n_1^2) d_1 / d_2 & \mathbf{n}_b \times \mathbf{n}_1 \cdot \mathbf{n}_b \times \mathbf{n}_2 \\
\mathbf{n}_b \times \mathbf{n}_1 \cdot \mathbf{n}_b \times \mathbf{n}_2 & (n_b \times n_2)^2 d_2 / d_1
\end{bmatrix},
\]

(3.21)
and
\[
\Lambda = \begin{bmatrix}
    n_1 \cdot F^1_{mb} \frac{m_b}{m_1} + \frac{m_b}{T_1} (v_1 - (v_b + d_1 \omega_b \times n_b))^2 \\
    n_2 \cdot F^2_{mb} \frac{m_b}{m_2} + \frac{m_b}{T_2} (v_2 - (v_b + d_2 \omega_b \times n_b))^2 \\
    + \frac{m_b d_1 n_1 \cdot n_b \omega_b^2}{m_b d_2 n_2 \cdot n_b \omega_b^2}
\end{bmatrix}.
\]

(3.22)

### 3.1.3 Equilibrium points for the UAV–bar system

Equilibrium points for an identical system have already been studied in [Ótá19, p. 172]. However, the derivations were made under the assumption that both UAVs were fully actuated, meaning that they were able to generate thrust in any given direction, regardless of their pose. Here, we instead consider the more realistic case where thrust only can be generated in the \( z \)-direction of the local frame of each UAV.

Heuristically, for system equilibrium we need that

1. The UAVs together apply a thrust in the global \( z \)-direction such that the weight of the whole system is cancelled
2. The string tension force acting on each UAV must be cancelled
3. The global \( z \)-component of the string tensions must cancel the weight of the bar.
4. The projection of the string tensions onto the bar normal vector must be zero, meaning that the contribution from each string must be of equal magnitude, but of opposing directions.

\[F^1, \theta_f^{f}, \theta_l \]
\[F^2, -\theta_f^{f}, -\theta_l \]

Figure 3.3: Schematic for illustrating the definition of parametrization variables \( \theta_f \) and \( \theta_l \).
We will now consider a symmetrical system in which

\[ d_1 = -d_2 = -d_b \]  
\[ l_1 = l_2 = l \]  
\[ m_1 = m_2 = m_{\text{uav}} \]

where \( d_b \) is half the bar length. We will also assume that the orientation of the bar is orthogonal to the z-direction and without loss of generality that \( n_b \) is parallel to \( e_x \), and that the bar is placed with its center of mass in the origin. Condition 4 above hints at that the pose of each UAV should be symmetrical with respect to \( e_z \). We therefore, in accordance with Figure 3.3, parametrize the thrust forces as

\[
F_1 = F \cdot (\sin \theta_f, 0, \cos \theta_f) \quad (3.26)
\]
\[
F_2 = F \cdot (\sin \theta_f, 0, \cos \theta_f) \quad (3.27)
\]

and the string tension forces as

\[
T_1 = T \cdot (\sin \theta_t, 0, \cos \theta_t) \quad (3.28)
\]
\[
T_2 = T \cdot (\sin \theta_t, 0, \cos \theta_t) \quad (3.29)
\]

where \( \theta_f = \theta_1 = -\theta_2 \) and \( \theta_t \) uniquely determine the pose of both UAVs, so that

\[
p_1 = (-d_b - l \sin \theta_t, 0, \cos \theta_t) \quad (3.30)
\]
\[
p_2 = (d_b + l \sin \theta_t, 0, \cos \theta_t) \quad (3.31)
\]

We are now ready to further specify the conditions that were heuristically derived before. To fulfill condition 1, we need

\[
(F_1 + F_2) \cdot e_z = 2F \cos \theta_f = g (m_b + 2m_{\text{uav}}). \quad (3.32)
\]

Condition 2 implies

\[
(F_1 - T_1) \cdot e_x = 0
\]
\[
(F_2 - T_2) \cdot e_x = 0
\]

and since an equilibrium point requires all state derivatives to be zero, the string tension forces \( T_i \) assume the relatively simple form

\[
T_i = \frac{m_b g \cos (\theta_f - \theta_t) (m_b + 2m_{\text{uav}})}{2 \cos (\theta_f) (2m_{\text{uav}} \cos^2 (\theta_t) + m_b)} n_i \quad (3.34)
\]
after substituting $F$ with the $F$ that is defined by Equation (3.32). Moreover, the symmetry of the problem means that Equation (3.33) can be reduced to the single equation

$$F \sin \theta_f - T \sin \theta_l = 0.$$  
(3.35)

For condition 3 to be fulfilled, it is necessary that

$$\frac{m_b g}{2} - T \cos \theta_l = 0.$$  
(3.36)

Finally, by adding Equations (3.35) and (3.36) we arrive at the equation

$$0 = \frac{g \sin (\theta_f) (m_b + 2m_{uav})}{2 \cos(\theta_f)} - \frac{m_b g}{2} + \frac{m_b g \cos(\theta_f - \theta_l) \cos(\theta_l) (m_b + 2m_{uav})}{2 \cos(\theta_f) (2m_{uav} \cos^2(\theta_l) + m_b) - \frac{m_b g \cos(\theta_f - \theta_l) \sin(\theta_l) (m_b + 2m_{uav})}{2 \cos(\theta_f) (2m_{uav} \cos^2(\theta_l) + m_b)}.$$  
(3.37)

Although closed form solutions might exist for this equation, the true point of interest is rather if any solutions at all exist. A numerical solution yields the results seen in Figure 3.4, from which it can be concluded that there in fact for every $\theta_l$ on the interval exists at least one $\theta_f$ such that Equation (3.37) is fulfilled (and vice versa). Note that these solutions seemingly depend on the masses $m_b$ and $m_{UAV}$ as well and therefore will tend to vary depending on the system configuration.

![Figure 3.4: Solutions to Equation (3.37).](image)

Out of all solutions, the trivial solution $\theta_f = \theta_l = 0$ constitutes the most efficient equilibrium point as the thrust direction is then completely aligned.
with the direction of gravity. The other equilibria, however, remain interesting in situations where the space in the $z$-direction is limited, and also enables the possibility of raising the bar while the $z$-position of the UAVs remains relatively intact (or lowering the UAVs while the bar remains in position.)

### 3.1.4 Modeling of a single UAV with payload

If a (point mass) payload is attached to the single UAV system described in Section 3.1.1 by the means of a cable (fig. 3.5), the resulting cable tension must also be taken into account. Similarly to what was done for the UAV–bar system in Section 3.1.2, the cable is approximated as a rigid link, meaning that it assumes a constant length (here $l$). Let the positions $p$ and $p_p$, the velocities $v$ and $v_p$, and the masses $m$ and $m_p$, correspond to those of the UAV and the payload, respectively. The cable tension $T$ is then given by the expression

\[
T = \frac{m_p}{m_p + m} \left( F_t R e_z \cdot n + m \frac{\|v - v_p\|^2}{l} \right)
\]  

(3.38)

where the cable direction vector

\[
n = \frac{p - p_p}{l},
\]

(3.39)

has been added for matters of simplification, and $R = R(\phi, \theta, \psi)$ is a rotation matrix, as defined in Section 3.1.1.

The equations of motion of the complete system are as follows

\[
\dot{p} = v
\]

(3.40)

\[
\ddot{p} = \frac{1}{m} \left( R e_z F_t - m g e_z - T n \right)
\]

(3.41)

\[
\dot{\Theta} = H \omega_B
\]

(3.42)

\[
\dot{\omega}_B = J^{-1} \left( \tau_B - \mathcal{S}(\omega_B) J \omega_B \right)
\]

(3.43)

\[
\dot{p}_p = v_p
\]

(3.44)

\[
\ddot{p}_p = \frac{1}{m_p} \left( T n - m_p g e_z \right)
\]

(3.45)

and are similar to the equations derived in Section 3.1.1 in every regard apart from the added effects of the cable tension, and the consequential need to add the equations of motion of the payload.

* Please note, however, that the given reference contains a typo, where $ML \|V - v\|^2$ should read $ML^{-1} \|V - v\|^2$. 
3.2 Controller derivations

3.2.1 MPC for a single UAV

Consider the model derived for a single UAV in Section 3.1.1. The ODE defined in Equation (3.8) describes the continuous time evolution of the system, but contains nonlinearities. For matters of simplifying the optimization problem\(^*\) the system is linearized around a hover point for an arbitrary position \(p\) and yaw \(\psi_0\)

\[
\begin{align*}
x_{eq} &= (p, 0, 0, 0, 0, \psi_0, 0, 0, 0) \\
u_{eq} &= (mg, 0, 0, 0). 
\end{align*}
\]  
(3.46)

This parametrized hover point results in a less complex model, but is general enough to capture many of the possible operating points. The linearization

\(^*\) Although not considered here, there are solvers capable of handling nonlinear MPC at acceptable speeds by different means of approximations in the solving phase. See [CSZ+20] for an example.
results in the (linear) ODE

\[
\begin{align*}
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{z} &= v_z \\
\dot{v}_x &= g(\theta \cos \psi_0 + \phi \sin \psi_0) \\
\dot{v}_y &= -g(\phi \cos \psi_0 - \theta \sin \psi_0) \\
\dot{v}_z &= F_i m^{-1} - g \\
\dot{\phi} &= p \\
\dot{\theta} &= q \\
\dot{\psi} &= r \\
\dot{\tau}_x &= I_x^{-1} \tau_x \\
\dot{\tau}_y &= I_y^{-1} \tau_y \\
\dot{\tau}_z &= I_z^{-1} \tau_z
\end{align*}
\] (3.47)

and can be written in matrix form as

\[
\dot{x} = A_c \Delta x + B_c \Delta u.
\] (3.48)

Since \( A_c \) is nilpotent with index four\(^*\), the ZOH transformation described in Appendix A is particularly straightforward and results in

\[
x_{k+1} = A \Delta x_k + B \Delta u_k.
\] (3.49)

with

\[
A = \\
\begin{bmatrix}
1 & 0 & 0 & h & 0 & 0 & g h^2 \sin (\psi_0) & g h^2 \cos (\psi_0) & 0 & g h^3 \sin (\psi_0) & g h^3 \cos (\psi_0) & 0 \\
0 & 1 & 0 & h & 0 & -g h^2 \cos (\psi_0) & g h^2 \sin (\psi_0) & 0 & -g h^3 \cos (\psi_0) & g h^3 \sin (\psi_0) & 0 \\
0 & 0 & 1 & 0 & h & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & g h \sin (\psi_0) & g h \cos (\psi_0) & 0 & g h^2 \sin (\psi_0) & g h^2 \cos (\psi_0) & 0 \\
0 & 0 & 0 & 0 & 1 & -g h \cos (\psi_0) & g h \sin (\psi_0) & 0 & -g h^2 \cos (\psi_0) & g h^2 \sin (\psi_0) & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & h & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & h & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & h \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\(^*\) In other words, \( A^4 = 0 \).
and

\[
B = \begin{bmatrix}
0 & \frac{gh^4 \sin(\psi_0)}{24I_x} & \frac{gh^4 \cos(\psi_0)}{24I_y} & 0 \\
0 & \frac{-gh^4 \cos(\psi_0)}{24I_x} & \frac{-gh^4 \sin(\psi_0)}{24I_y} & 0 \\
\frac{h^2}{2m} & 0 & 0 & 0 \\
0 & \frac{gh^3 \sin(\psi_0)}{6I_x} & \frac{gh^3 \cos(\psi_0)}{6I_y} & 0 \\
0 & \frac{-gh^3 \cos(\psi_0)}{6I_x} & \frac{-gh^3 \sin(\psi_0)}{6I_y} & 0 \\
\frac{h}{m} & 0 & 0 & 0 \\
0 & \frac{h^2}{2T_c} & 0 & 0 \\
0 & 0 & \frac{h}{T_c} & 0 \\
0 & 0 & 0 & \frac{h}{T_c} \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

for a sampling time \( h \). A model predictive controller can now be designed by using Equation (3.49) in the general MPC problem defined in Equation (2.13) as well as choosing appropriate weight matrices \( Q, R, Q_N \), and constraint sets \( \mathcal{X}, \mathcal{U}, \mathcal{X}_N \). The parameter \( \psi_0 \) is set at each sampling instance to be equal to the yaw at that specific instance. The terminal cost \( Q_N \) may favourably be set to be equal to the solution of Equation (2.16) as discussed in Section 2.2.3. In this case, however, it is worthwhile noting that this solution depends on the yaw parameter \( \psi_0 \), and \( Q_N \) must therefore be redefined online if setpoints with distinct yaw references are to be used.

### 3.2.2 Offset-free MPC for a single UAV

Section 2.2.4 highlighted the importance of model accuracy, but demonstrated that a slight reformulation of the optimization problem could combat stationary errors due to model mismatches: a strategy known as offset-free MPC. This strategy could certainly prove helpful for asserting that a single UAV is able to reach a desired height even though the estimated weight of the UAV differs slightly from the actual, but it could also be helpful for treating external disturbances, such as the weight from a payload attached to the UAV. For this purpose the model is reformulated as

\[
x_{k+1} = Ax_k + Bu_k + B_d d_k. \tag{3.50}
\]

with \( B_d \) receiving a slightly different form than in Example 2.2.3 since we are now considering more states and consequently have to augment \( B_d \) with
zeros at the appropriate places. Following the ordering of states that has been used throughout the thesis, $B_d$ is a column vector of zeros except for rows three and six, that read $-h^2/2$ and $-h$, respectively. Everything that is needed for the controller in Equation (2.22) is to construct an estimator, and to find appropriate weights for both the controller and said estimator.

### 3.2.3 Centralized MPC for a UAV–bar system

This controller makes use of the full model derived in Section 3.1.2, meaning that it will take into account both UAVs and their dynamic interaction with the bar. The high complexity of these interaction forces makes the linearization step increasingly important for this system. Just as in the single UAV case, a parametrized linearization is favoured for its generalizability. While this method might take more time during the development stage, it saves precious time in the online control process where the linearization would otherwise have had to be performed. The considered linearization point is the hover position of both UAVs characterized by $\theta_l = \theta_f = 0$ in Figure 3.3. One parameter is considered for each rigid body: $\psi_0^{(b)}$, $\psi_0^{(1)}$, and $\psi_0^{(2)}$. The bar is given an arbitrary position $\mathbf{p}_b$, but the position of UAV 1 and 2 is decided completely based on the yaw and position of the bar if the system is to remain in the decided equilibrium position. Consequently, the equilibrium state is given by

$$
x_{\text{eq}} = (x_{\text{eq}}^{(b)}, x_{\text{eq}}^{(1)}, x_{\text{eq}}^{(2)}),
$$

where

$$
x_{\text{eq}}^{(b)} = (p_b, 0, 0, 0, 0, 0, 0, 0, 0)
$$

$$
x_{\text{eq}}^{(1)} = (p_1, 0, 0, 0, 0, 0, 0, 0, 0)
$$

$$
x_{\text{eq}}^{(2)} = (p_2, 0, 0, 0, 0, 0, 0, 0, 0)
$$

and

$$
p_1 = p_b + (-d_1 \sin \psi_0^{(b)}, d_1 \cos \psi_0^{(b)}, l_1)
$$

$$
p_2 = p_b + (-d_2 \sin \psi_0^{(b)}, d_2 \cos \psi_0^{(b)}, l_2).
$$

The corresponding input equilibrium is given by the conditions of force and moment equilibrium which result in

$$
u_{\text{eq}} = m_b g \left( \frac{d_2}{d_2 - d_1}, 0, 0, \frac{d_1}{d_1 - d_2}, 0, 0, 0 \right).
$$
The resulting ODE has been forwarded to Appendix B. We now favour the slightly different form of the matrix description used previously

\[ \dot{x} = A_c x + B_c u + c_c, \]  
where the deviations from the linearization point now are captured by the term

\[ c_c = -A_c x_{eq} - B_c u_{eq}. \]

Unfortunately, the matrix \( A_c \), if nilpotent at all, has a very high index. The ZOH-discretization used previously is therefore not a preferable choice since it would include high-order terms of the parameters. For matters of keeping down the model complexity, its first order approximation is instead used*, yielding the system

\[ x_{k+1} = Ax_k + Bu_k + c_d, \]

with

\[ A = I + A_c h \]
\[ B = B_c h \]
\[ c_d = c_c h. \]

### 3.2.4 Decentralized MPC for a UAV–bar system

For this approach, the UAV–bar system is approximated further by instead considering it as two independent UAVs with payloads. Hence, the endpoints of the bar are tracked, and their position and velocity considered as the positions and velocities of imaginary payloads attached to each UAV by cables. The weight given to each (imaginary) payload is such that it matches with the input equilibrium defined in Equation (3.57). Thus the weight of the payload attached to UAV 1 is

\[ F_{g1} := m_b g \frac{d_2}{d_2 - d_1}, \]

while the weight of the payload attached to UAV 2 is

\[ F_{g2} := m_b g \frac{d_1}{d_1 - d_2}. \]

* See Appendix A.2 for details.
With these modifications in place, the equations of motion of the $i$:th UAV and payload are given as follows

$$\dot{p}_i = v_i$$  \hspace{1cm} (3.66)
$$\ddot{p}_i = \frac{1}{m_i} \left( R_{\Theta_i} e_z F_i - m_i g e_z - T_i n_i \right)$$  \hspace{1cm} (3.67)
$$\dot{\Theta}_i = H_{\Theta_i} \omega_{B,i}$$  \hspace{1cm} (3.68)
$$\dot{\omega}_{B,i} = J_i^{-1} (\tau_{B,i} - \delta(\omega_{B,i}) J_i \omega_{B,i})$$  \hspace{1cm} (3.69)
$$\dot{p}_{p(i)} = v_{p(i)}$$  \hspace{1cm} (3.70)
$$\ddot{p}_{p(i)} = \frac{1}{m_p} \left( T_i n_i - F_{g(i)} e_z \right)$$  \hspace{1cm} (3.71)

where $T_i = T_i(p_i, p_{p(i)}, v_i, v_{p(i)})$ in this case is given by Equation (3.38). The most important part of these equations is that there no longer exists any algebraic coupling between the UAVs; each UAV, and corresponding payload, can therefore be treated as an independent system. In an MPC setting, this also means that each UAV can be controlled by an independent MPC, with both MPCs running in a parallel fashion. Furthermore, this formulation does not rely on any explicit information exchange: the only information exchange between both controllers is implicit and is contained in the disturbances that one UAV might experience by cause of the other.

As done continuously throughout this work, the system is linearized leaving the UAV yaw as a parameter $\psi_0$. This results in the following linear ODE belonging to $\mathbb{R}^{18}$

$$\begin{align*}
\begin{bmatrix}
v \\
gm^{-1} \left( \theta c_{\phi_0} (m + m_p) + \phi s_{\phi_0} (m + m_p) + l^{-1} m_p (p_{px} - p_x) \right) \\
gm^{-1} \left( \phi s_{\phi_0} (m + m_p) - \phi c_{\phi_0} (m + m_p) + l^{-1} m_p (p_{py} - p_y) \right) \\
F (m + m_p)^{-1} - g + 2gm_p m^{-1} (1 + l^{-1} (p_{pz} - p_z)) \\
\omega_{B} \\
J^{-1} \tau_{B} \\
v_p \\
-g l^{-1} (p_{px} - p_x) \\
-g l^{-1} (p_{py} - p_y) \\
F (m + m_p)^{-1} - 3g - 2gl^{-1} (p_{pz} + p_z)
\end{bmatrix}
\end{align*}$$

(3.72)

where the indices of UAV and payload have been dropped in favour of simpler notation. With a model in place, the MPC formulation is complete after setting appropriate weights and constraint sets.
System analysis and controller design
Chapter 4

Methods

4.1 Simulation setup

4.1.1 Simulation environment

The simulation environment is set up in CasADi\(^*\) via its python application programming interface (API) by using the nonlinear equation system defined by Equations (3.10) to (3.16). The cables are simulated as massless, rigid links and the length constraint in each of them is maintained by using the GGL approach. The tension forces \(T_i\) are thus considered as Lagrange multipliers \(\lambda_i\), and the corresponding constraint Jacobians \(G_i\) can be calculated to be equal to \(n_i\). The GGL approach also requires additional Lagrange multipliers \(\mu_i\) that are added to the kinematic equations of the UAVs. The simulated system is thus

\[
\begin{align*}
\dot{p}_i &= v_i + \mu_i n_i & i &= 1, 2 \\
\dot{v}_i &= m_i^{-1} \left( (F_i R \dot{\theta}_i - m_i g I) e_z - \lambda_i n_i \right) & i &= 1, 2 \\
\dot{\theta}_i &= H_{\theta_i} \omega_{\mathcal{B},i} & i &= 1, 2 \\
\dot{\omega}_{\mathcal{B},i} &= J_i^{-1} \left( \tau_{\mathcal{B},i} - \mathcal{S}(\omega_{\mathcal{B},i}) J_i \omega_{\mathcal{B},i} \right) & i &= 1, 2 \\
\dot{p}_b &= v_b \\
\dot{v}_b &= m_b^{-1} (\lambda_1 n_1 + \lambda_2 n_2 - m_b g) \\
\dot{\theta}_b &= H_{\theta_b}^{(g)} \omega_b \\
\dot{\omega}_b &= J_b^{-1} \mathcal{S}(n_b) (d_1 \lambda_1 n_1 + d_2 \lambda_2 n_2)
\end{align*}
\]

\(^*\) AGH+18. \(^\dagger\) Version 3.5.5, built from source and including LLVM 3.4.2 for just-in-time (JIT) compilation support.
along with the algebraic constraints

\[ \| p_i - (p_b + d_i n_b) \|^2 - l_i^2 = 0 \]

\[ [p_i - (p_b + d_i n_b)] \cdot [v_i - (v_b + d_i \delta (\omega_b n_b))] = 0, \]

and the notation following that of Section 3.1.2.

SUNDIALS’ IDAS* is chosen as a solver since it is able to handle index-2 DAEs. An important note, however, is that the ability of IDAS to calculate consistent initial conditions is non-existent in the case of index-2 DAEs. These initial conditions must therefore be calculated before-hand, and the option calc_ic be set to false. A set of consistent initial conditions is

\[ x = x_0 \]

\[ \lambda = (T_1(x_0, u_0^*(x_0)), T_2(x_0, u_0^*(x_0)) \]

\[ \mu = (0, 0) \]

with \( T_i \) from Equation (3.20), \( x_0 \) being a state vector consistent with the algebraic equations and \( u_0^*(x_0) \) the applied control signal.

### 4.1.2 Simulation experiments

The centralized MPC using the linearized, complete state-space model introduced previously, is tested by setpoint tracking. For comparison, a naive dual-controller setup in which each UAV is unknowing of the other is included. In this setup, the cable tension is treated as a disturbance in the \( z \)-direction using an offset-free MPC approach based on a linearized single-UAV model as detailed in Section 3.2.2, and the disturbance estimate is initialized to the disturbance caused by the weight of the bar in equilibrium. In order to not overtune the MPC weights for each specific experiment, the same weights were used throughout all simulation experiments. These weights along with other relevant MPC parameters such as horizon and sampling time are found in Tables 4.1 and 4.2. The input constraints were set in order to guarantee a positive thrust \( (F > 0) \) and such that the thrust magnitudes were less than or equal to one \( (|\tau_i| \leq 1, \ \forall i) \).

All experiments were performed on a computer with the specifications found in Table 4.3 running Ubuntu 18.04.5 LTS x86_64 with Linux kernel version 5.4.0-70-generic.

* HBG+05.
Table 4.1: Parameter settings for the centralized MPC in the simulation experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>diag($Q'<em>{b}$, $Q'</em>{UAVs}$)</td>
</tr>
<tr>
<td>$Q'_{b}$</td>
<td>diag(50, 50, 50, 30, 30, 10, 10, 10, 10)</td>
</tr>
<tr>
<td>$Q'_{UAVs}$</td>
<td>diag(50, 50, 10, 10, 10, 100, 100, 100, 100)</td>
</tr>
<tr>
<td>$R$</td>
<td>diag(1, 10, 10, 10, 10, 10)</td>
</tr>
<tr>
<td>$Q_N$</td>
<td>100$Q$</td>
</tr>
<tr>
<td>$h$</td>
<td>0.02 s</td>
</tr>
<tr>
<td>$N$</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 4.2: Parameter settings for the offset-free MPC in the simulation experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$</td>
<td>diag(100, 100, 100, 100, 10, 10, 10, 1, 1, 1)</td>
</tr>
<tr>
<td>$R$</td>
<td>diag(1, 10, 10, 10)</td>
</tr>
<tr>
<td>$Q_N$</td>
<td>100$Q$</td>
</tr>
<tr>
<td>$h$</td>
<td>0.02 s</td>
</tr>
<tr>
<td>$N$</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 4.3: CPU and RAM specifications for the workstation used throughout the thesis.

<table>
<thead>
<tr>
<th>CPU</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel i7-6700 at 4 GHz (8 threads)</td>
<td>32 GB DDR4 at 2133 MHz</td>
</tr>
</tbody>
</table>
4.1.2 Damping of bar angular oscillation

In this experiment, the system starts in the linearization point with $\psi_b = \psi_1 = \psi_2 = 0$, and all velocities set to zero except that the bar is given an initial angular velocity

$$\omega_b = (0, 0, 3) \text{ rad s}^{-1}. \quad (4.4)$$

4.1.2 Damping of bar linear oscillation

Similar to the angular oscillation damping experiment, this experiment has the system starting around the linearization point with all yaw angles set to zero. However, in this case the bar is given an initial linear velocity

$$v_b = (3, 0, 0) \text{ m s}^{-1}. \quad (4.5)$$

All other velocities are set to zero.

4.1.2 General performance

This experiment intends to test the overall performance and therefore includes a variety of setpoints which includes yaw rotations of all three rigid bodies as well as motion in the $xy$-plane. It also includes one of the setpoints derived in Section 3.1.3 with $\theta_l \neq 0$. See Table 4.4 and Figure 4.1 for a full description.

Table 4.4: The six setpoints used in the test of general performance.

<table>
<thead>
<tr>
<th>$x_b$</th>
<th>$\Theta_b$</th>
<th>$x_1$</th>
<th>$\Theta_1$</th>
<th>$x_2$</th>
<th>$\Theta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0.5, 0)</td>
<td>(0, 0, 0)</td>
<td>(0, 0.1)</td>
<td>(0, 0)</td>
<td>(0.1, 1)</td>
<td>(0, 0.0)</td>
</tr>
<tr>
<td>(0, 0.5, 0)</td>
<td>(0, 0, $-\pi/2$)</td>
<td>($-0.5, 0.5, 1$)</td>
<td>(0, 0, $-\pi/2$)</td>
<td>(0.5, 0.5, 1)</td>
<td>(0, 0.0)</td>
</tr>
<tr>
<td>(0, 1.5, 0)</td>
<td>(0, 0, $-\pi/2$)</td>
<td>($-0.5, 1.5, 1$)</td>
<td>(0, 0)</td>
<td>(0.5, 1.5, 1)</td>
<td>(0, $-\pi/2$)</td>
</tr>
<tr>
<td>(0, 0.5, 0)</td>
<td>(0, 0.0)</td>
<td>(0, 0.1)</td>
<td>(0, 0)</td>
<td>(0.1, 1)</td>
<td>(0, 0.0)</td>
</tr>
<tr>
<td>(0, 0.5, 0.5)</td>
<td>(0, 0.0)</td>
<td>(0, $-0.87, 1$)</td>
<td>(0.156, 0, $\pi$)</td>
<td>(0.1, 1.87, 1)</td>
<td>$-(0, 1.56, 0, \pi$)</td>
</tr>
<tr>
<td>(0, 0.5, 0)</td>
<td>(0, 0.0)</td>
<td>(0, 0.1)</td>
<td>(0, 0)</td>
<td>(0.1, 1)</td>
<td>(0, 0.0)</td>
</tr>
</tbody>
</table>

4.2 Experimental setup

4.2.1 ROS

The Robot Operating System (ROS)* is an open source middleware that provides powerful tools for development of robots and eases the initial setup.

* QGC+09.
process. The core concept of ROS are so called *nodes*: applications that are able to interchange information by subscribing or publishing to *topics* (fig. 4.2). This feature makes ROS highly modular and enables easy reuse of previously written code as well as seamless incorporation of third-party modules into an existing code base.

Figure 4.2: ROS nodes (elliptical) that publish and listen to information channels called topics (rectangular).

Another relevant feature of ROS is that it is able to communicate with the Gazebo simulator*. This enables direct testing of the developed ROS nodes in

* http://gazebosim.org/
a simulation environment, given that an appropriate Gazebo-compliant model has been constructed.

### 4.2.2 PX4

PX4* is an open source flight control stack that is usable with a large number of different UAVs ranging from quadcopters and hexacopters to fixed-wing UAVs. Although commonly used for high-level control, it provides low-level, offboard control through the MAVROS† interface where it is possible to send in e.g. reference positions, or raw servo values for direct actuator control.

The PX4 internal flight controller, illustrated in Figure 4.3, is built in a cascaded fashion with faster inner loops for controlling attitude and angular rate and a slower outer loop for position control. Since PX4 is designed to work with a wide range of UAVs, the control structure includes a mixer with the purpose of translating the signals $\delta_\text{ref}$ to relevant motor servo values $\Omega_\text{ref}$ (fig. 4.3). Depending on the type of reference that is sent to MAVROS, certain parts of the controller structure can be bypassed. In the experiments, angular rates and thrust are sent into MAVROS and the relevant controller in this case is the angular rate controller part of Figure 4.3, seen in Figure 4.4.

![Figure 4.3: PX4 internal control scheme.](PX420)

### 4.2.3 System input considerations

An important inconsistency with the controllers proposed in Section 3.2 and the PX4 flight stack is that the controller structure provided by PX4 does not accept torques as input values. One solution to this issue would be to find the inverse relationship between one of the quantities that PX4 accepts as input (and potentially other relevant state variables) and the resulting torque. A far

---

* https://px4.io/ † https://github.com/mavlink/mavros
more straightforward solution is to use the predictions generated as part of the MPC process. Since optimal control inputs lead to optimal states, the optimality is kept, provided that the low-level controllers give a fast enough response. For example, it is possible to extract the angular rates that the controller is expecting to have reached at time step $k$ by the following formula*

$$\omega_{b}(x_0, \{u_i^{*}\}_{i=0}^{k-1}, k) = \left[ A^k x_0 + \sum_{i=1}^{k-1} A^i B u_{k-1-i}^i \right]_{10:12} \quad (4.6)$$

and use these rates as input to the PX4 controller (fig. 4.4). An important thing to consider in this case is how to choose $k$ in Equation (4.6): under ideal conditions, the natural choice would be to set $k$ to 1, but since the corresponding PX4 controller might introduce some lag into the system a value $k > 1$ might be preferable.

### 4.2.4 Experimental validation, Centralized MPC

Two Storm SRD370 quadrotors (one with the original Storm M2212 980kv motors, one with replacement T-Motor MN2213 950kv motors) were used in the experiment. Both quadrotors were equipped with onboard mRo Pixracer flight control units (FCUs) (fig. 4.5). These include sensors such as accelerometers and gyros, and also enable communication over Wi-Fi through a ESP8266 microchip (Espressif Systems). The FCUs were flashed with PX4 firmware‡, and communicated with the host computer (table 4.3) over Wi-Fi using the MAVLink§ messaging protocol. For this purpose, the host computer used MAVLink¶ accompanied with MAVROS®, which enabled the translation of ROS messages to

* This, of course, assuming a linear model $x_{k+1} = Ax_k + Bu_k$.  
† $[X]_{J:K}$ denotes an extraction of elements $J$ to $K$ from $X$, so that $[X]_{J:K} = (X_J, X_{J+1}, ..., X_K)$ where $X_J$ denotes the $J$:th element of $X$.  
‡ version 1.10.1  
§ version release/kinetic/mavlink/2020.9.10-1  
¶ version 1.4.0

![Figure 4.4: PX4 angular rate controller. [PX420]](image)
the relevant MAVLink messages. A motion capture system (Qualisys) was used for receiving position, and velocity estimates. See Figure 4.6 for a schematic illustration of the connections.

![Figure 4.5: The mRo Pixracer FCU used in the experimental validation fitted with an Espressif Systems ESP8266 Wi-Fi module. [PX420]](image)

![Figure 4.6: An illustration of the experimental setup and the connection between the ingoing devices, where solid lines represent wired connections, and dashed lines wireless connections.](image)

The MPC was formulated as a ROS node and the optimal control problem formulated in CasADi and solved using the solver hpmpc*. In order to reduce the effect of the delay introduced by the PX4 controllers, \( k \) (as described in Equation (4.6)) was given the value of 5. The parameters of the controller was set as seen in Table 4.5, and similarly to the simulation experiments the input

* [https://github.com/giaf/hpmpc](https://github.com/giaf/hpmpc)
methods were set in order to guarantee a positive thrust ($F_i > 0$) and such that the thrust magnitudes were less than or equal to one ($|τ_j^{(i)}| \leq 1, \forall j$).

The experiment consisted in setpoint-tracking using the setpoints as given in Table 4.6 (corresponding to transitions one, two and three in Figure 4.1) and was performed at the $6 \times 6 \times 3$ m$^3$ arena at the KTH Smart Mobility Lab. As payload, a hollow bar weighing 0.39 kg and of length 1.47 m was used, and the Storm SRD370s were modeled as having masses $m_i = 1.15$ kg, and moments of inertia $J_i = \text{diag}(0.0347563, 0.0458929, 0.0977)$ kg m$^2$.

An overview of the ROS architecture can be viewed in Figure 4.7.

Table 4.5: Parameter settings for the MPC used in the experimental validation of the centralized MPC.

<table>
<thead>
<tr>
<th>$Q$</th>
<th>diag($Q_b$, $Q_{\text{UAV}1}$, $Q_{\text{UAV}2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_b$</td>
<td>diag(50, 50, 50, 50, 50, 10, 10, 1, 1, 1, 5)</td>
</tr>
<tr>
<td>$Q_{\text{UAV}1}$</td>
<td>diag(30, 30, 30, 30, 30, 10, 10, 10, 10, 10, 100)</td>
</tr>
<tr>
<td>$R$</td>
<td>diag(1, 100, 100, 50, 1, 100, 100, 50)</td>
</tr>
<tr>
<td>$Q_N$</td>
<td>100$Q$</td>
</tr>
<tr>
<td>$h$</td>
<td>0.02 s</td>
</tr>
<tr>
<td>$N$</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 4.6: The setpoints used during the experimental validation.

<table>
<thead>
<tr>
<th>$x_b$</th>
<th>$\Theta_b$</th>
<th>$x_1$</th>
<th>$\Theta_1$</th>
<th>$x_2$</th>
<th>$\Theta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0, 0.3)</td>
<td>(0, 0, 0)</td>
<td>(0, 0, 0.3)</td>
<td>(-$d_b$, 1.3)</td>
<td>(0, 0, 0)</td>
<td>(0, 0, 0.3)</td>
</tr>
<tr>
<td>(0, 0, 0.3)</td>
<td>(0, 0, 0, 0)</td>
<td>(-$d_b$, 0.13)</td>
<td>(0, 0, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
</tr>
<tr>
<td>(0, 1, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
<td>(-$d_b$, 1.3)</td>
<td>(0, 0, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
</tr>
<tr>
<td>(0, 0, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
<td>(-$d_b$, 1.3)</td>
<td>(0, 0, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
<td>(0, 0, 0, 0)</td>
</tr>
</tbody>
</table>

4.2.5 Experimental validation, Decentralized MPC

An experimental validation of the decentralized MPC introduced in Section 3.2.4 was performed using a setup that was identical to the validation of the centralized MPC, with the obvious exception of changing the centralized MPC to two independent controllers (again, as described in Section 3.2.4). Furthermore, the bar endpoints were to be tracked, and the bar was therefore set up with new motion capture markers. During initial tests it was also discovered that the previous assumption that the bar was homogenous with a CoM in the
Figure 4.7: A graph displaying the ROS architecture, where elliptical shapes correspond to nodes, and rectangular to topics. Some less important nodes have been left out for clarity, as has one of the UAVs (since the setup is identical.)
center of its extension along the axis of symmetry was not accurate enough for the decentralized setup. Instead, \(d_1\) and \(d_2\) were re-identified and found to be

\[
d_1 = -70.5 \text{ cm}, \quad d_2 = 76.5 \text{ cm},
\]

and the values of \(F_g^{(i)}\) updated according to Equations (3.64) and (3.65). The weights and parameters of both controllers were set as seen in Table 4.7. The relatively low penalization of velocity deviations of the payload were set after simulation trials, but can be retroactively motivated as follows: putting high penalization on the velocity of the payload will make the controller try to damp the payload more, but since both controllers work independently, the damping of one of the payloads is likely to imply a disturbance of the other payload.

Table 4.7: Parameter settings for the MPC used in the experimental validation of the decentralized MPC. The controllers of both MPC used the same weights. Parameter \(k\) corresponds to the one defined in Equation (4.6).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Q)</td>
<td>diag ((Q_{UAV}', Q_p'))</td>
</tr>
<tr>
<td>(Q_{UAVs}')</td>
<td>diag (20, 20, 20, 30, 30, 30, 10, 10, 10, 10, 10, 30)</td>
</tr>
<tr>
<td>(Q_p')</td>
<td>diag (20, 20, 20, 1, 1, 1)</td>
</tr>
<tr>
<td>(R)</td>
<td>diag (1, 100, 100, 100)</td>
</tr>
<tr>
<td>(Q_N)</td>
<td>50 (Q)</td>
</tr>
<tr>
<td>(h)</td>
<td>0.02 s</td>
</tr>
<tr>
<td>(N)</td>
<td>30</td>
</tr>
<tr>
<td>(k)</td>
<td>7</td>
</tr>
</tbody>
</table>
Methods
Chapter 5

Results

5.1 Simulations

In this section, the results from the simulation experiments described in Section 4.1.2 are presented. All figures have their description and units written in the title, and all x-axes correspond to time in seconds unless stated otherwise. Figures that contain three graphs follow the ordering bar, UAV 1, UAV 2 from top to bottom.

5.1.1 Damping of bar angular oscillation

The results are presented in Figures 5.1 to 5.4. The centralized MPC efficiently is able to damp the oscillations: Figure 5.2 shows that the oscillations are, at least visibly, completely damped after around 10 s. The offset-free MPC, on the other hand, was not able to dampen to the same degree, even over a 25 s time interval. An interesting feature seen in Figure 5.1a is that the centralized MPC seems to favour a (rather non-intuitive) configuration in which both UAVs go outwards in the y-direction before returning back to their initial position. Similarly, their x-positions are not immediately driven down to zero, but rather in a gradual, seemingly controlled, fashion. Put this in contrast to Figure 5.1b, where the UAVs are rapidly going towards zero, but continuously overshooting due to the unexpected influence of the bar.

5.1.2 Damping of bar linear oscillation

The results are presented in Figures 5.5 to 5.8. The overall behaviour of each controller is similar to what was described in the previous simulation
Figure 5.1: Damping of bar angular oscillation (section 4.1.2.1) results showing linear position for the: (a) Centralized MPC; (b) Offset-free MPC variant.

Figure 5.2: Damping of bar angular oscillation (section 4.1.2.1) results showing angular position for the: (a) Centralized MPC; (b) Offset-free MPC variant.
Figure 5.3: Damping of bar angular oscillation (section 4.1.2.1) results showing the resulting thrust output from the: (a) Centralized MPC; (b) Offset-free MPC variant.

Figure 5.4: Damping of bar angular oscillation (section 4.1.2.1) results showing the resulting torque output from the: (a) Centralized MPC; (b) Offset-free MPC variant.
experiment: while the offset-free MPC puts its focus into just driving the UAVs back to their initial position, the centralized MPC is more allowing of initial deviations. Figure 5.5 shows that the centralized MPC achieves approximataly the same amount of damping in 10 s as the offset-free MPC does in 25 s. Compared to the previous simulation experiment, the offset-free MPC seems to damp more efficiently this time. A reasonable explanation to this could be that the effect that each UAV has on the other is less counter-productive in this case: in the damping of angular oscillation, the UAVs, in their effort to reach the regulation point inadvertently give the bar more rotational energy (see e.g. Figure 1.4).

Figure 5.5: Damping of bar linear oscillation (section 4.1.2.2) results showing linear position for the: (a) Centralized MPC; (b) Offset-free MPC variant.

### 5.1.3 General performance

The results are presented in Figures 5.9 to 5.12. The centralized MPC clearly outperforms the offset-free MPC, demonstrating a much less oscillatory response, and comparably fast and precise transitions between setpoints, with low overshoot. Furthermore, all setpoints are reached without static errors, except for the non-trivial setpoint (see Section 3.1.3) between the 20 and 25 second mark where the UAVs are not able to reach the intended $y$-position, but instead compensate by increasing their $z$-position so that the bar reaches its setpoint in the $z$-axis.
Figure 5.6: Damping of bar linear oscillation (section 4.1.2.2) results showing angular position for the: (a) Centralized MPC; (b) Offset-free MPC variant.

Figure 5.7: Damping of bar linear oscillation (section 4.1.2.2) results showing the resulting thrust output from the: (a) Centralized MPC; (b) Offset-free MPC variant.
Figure 5.8: Damping of bar linear oscillation (section 4.1.2.2) results showing the resulting torque output from the: (a) Centralized MPC; (b) Offset-free MPC variant.

Figure 5.9: General performance (section 4.1.2.3) results showing linear position for the: (a) Centralized MPC; (b) Offset-free MPC variant.
Figure 5.10: General performance (section 4.1.2.3) results showing angular position for the: (a) Centralized MPC; (b) Offset-free MPC variant.

Figure 5.11: General performance (section 4.1.2.3) results showing the resulting thrust output from the: (a) Centralized MPC; (b) Offset-free MPC variant.
Figure 5.12: General performance (section 4.1.2.3) results showing the resulting torque output from the: (a) Centralized MPC; (b) Offset-free MPC variant.

5.2 Experiments

5.2.1 Centralized MPC

In this section, the results of the experimental validation described in Section 4.2.4 are presented. The figures follow the same specifications explained for the simulation experiments in Section 5.1. Figure 5.13 shows position and orientation of all rigid bodies, as well as the reference signals (in dashed lines). Figure 5.14 shows the resulting optimal control signals. The CPU time taken to solve the MPC problem can be seen in Figure 5.15.

The overall behaviour of the controller is consistent with the simulation results: the system is able to track all setpoints, both in terms of position and orientation. With this said, the regulation around setpoints is clearly less stable than for the simulation case (especially for UAV 2). Whether this is an effect of the low-level controller provided by PX4, the MPC, or both is hard to answer without rigorous testing, but for the record, similar behaviour could be seen under position control of a single UAV while completely relying on the PX4 internal controller (fig. 4.3).

An interesting phenomenon occurs between approximately seconds 10 to 17 for UAV 1, and between seconds 25 to 30 for UAV 2 in Figure 5.14b, where $\tau_y^{(i)}$ oscillates and saturates at $-1$ and $1$. This happens predominately when the UAVs are rotated by $-90$ degrees, and a likely cause could be a failure
in attaching the cables at the UAV center of mass,* which would mean that
the bar would provide the UAVs with an external torque. However, this on its
own does not explain why this effect is mostly visible when the UAVs are in
rotated positions.

The somewhat noisy appearance of the input signals seen in Figure 5.14 is
likely to be attributed to the linear velocity part of the motion capture system
which itself seemed to be affected by noise.

In Figure 5.14a it can be noted that the thrust of UAV 1 seems to be
of consistently larger magnitude than that of UAV 2. This is line with the
non-symmetric placement of the bars CoM discussed in Section 4.2.5, but it
is uncertain how this information would enter into the optimization process.
A likely cause could be that the angular acceleration that the asymmetric
placement of the bar CoM causes gets picked up by the MPC as an angular
velocity, and consequently results in a higher thrust for UAV 1 in order to
counteract this velocity.†

![Figure 5.13: Experimental validation results showing: (a) position; (b) orientation.](image)

5.2.2 Decentralized MPC

In this section, the results of the experimental validation described in Sec-
section 4.2.5 are presented. The figures follow the same specifications explained

---

* Symmetry around the UAV x-axis makes center of mass identification trivial in the x-
direction, but a lack of symmetry in y-direction makes this harder in this direction.  † See
Equation (B.1) for more details on the relation between thrust and bar angular acceleration/ve-
locity.
Figure 5.14: Experimental validation results showing the achieved optimal input signals: (a) thrust; (b) torque.

Figure 5.15: CPU time taken to solve the centralized MPC problem. Although the average is well below the sampling time 0.02 s, occasional spikes exceeding this number can be observed.
for the simulation experiments in Section 5.1. Figure 5.16a shows position of all rigid bodies, as well as the reference signals (in dashed lines); however, as the bar centerpoint was no longer being tracked, its estimated position was calculated as

\[
p_b = p_p^{(1)} + \frac{d_2 - d_1}{2} \left( p_p^{(2)} - p_p^{(1)} \right).
\]  

(5.1)

Figure 5.16b shows the position of the endpoints of the bar. Figure 5.17 shows the orientations of the UAVs, as well as the estimated yaw angle of the bar, calculated by

\[
\arctan2(y_p^{(2)} - y_p^{(1)}, x_p^{(2)} - x_p^{(1)}).
\]  

(5.2)

where \(x_p^{(i)}, y_p^{(i)}\) are the x and y positions of payload \(i\), and \(\arctan2 : \mathbb{R}^2 \to (-\pi, \pi]\) is the function commonly found in many programming languages which, in contrast to \(\arctan\), returns values in the range \((-\pi, \pi]\). Figure 5.18 shows the resulting optimal control signals, where it should be noted that the relatively large difference between the thrust signals of UAV 1 and 2 likely stems from the differences in \(F_g\) (eqs. (3.64) and (3.65)) as discussed in Section 4.2.5. Finally, the CPU time taken to solve the MPC problem can be seen in Figure 5.19; however, since two controllers were used and run in parallel, a more interesting quantity is the maximum of time taken by the two controllers. Therefore, Figure 5.19 shows

\[
\max_i T_{CPU}^{(i)}(t)
\]  

(5.3)

where \(T_{CPU}^{(i)}(t)\) is the CPU time taken for controller \(i\) at a time step \(t\). The resulting average of 2.8 ms is about 2.5 times faster than for the centralized MPC case, and furthermore, all CPU times are below the sampling time of 0.02 s.

Similar to the validation of the centralized MPC, the noisy appearance of the input signals seen in Figure 5.18 is likely to be caused by the noise in the linear velocity part of the motion capture system.

### 5.2.3 Comparison

The purpose of this section is to provide quantitative comparisons between the performance of both the centralized and decentralized MPC. Figure 5.20 shows the difference

\[
(e_x, e_y, e_z) := p^{(b)} - p_{ref}^{(b)}
\]

between the position of the bar centerpoint \(p^{(b)}\) and its desired position \(p_{ref}^{(b)}\) for both the centralized and decentralized case. Important to mention is that,
Figure 5.16: Experimental validation results for the decentralized MPC showing: (a) the (calculated) position of bar centerpoint and UAVs; (b) the position of the endpoints of the bar.

Figure 5.17: Experimental validation results for the decentralized MPC showing the angular position tracking. Note that the bar yaw angle (uppermost figure) has been calculated by considering the angle between the two endpoints.
Figure 5.18: Experimental validation results for the decentralized MPC showing the achieved optimal input signals: (a) thrust; (b) torque.

Figure 5.19: The maximum CPU time taken to solve the MPC problem by the two decentralized controllers. All times are below the sampling time of 0.02 s.
although the decentralized case does not explicitly track the bar centerpoint position, this tracking is implicitly enforced by the bar endpoint references. In fact, a perfect tracking of the bar endpoint positions automatically implies a perfect tracking of the bar centerpoint.

![Graphs (a) and (b)](image)

**Figure 5.20:** The errors in position of the centerpoint of the bar for the: (a) centralized MPC; (b) decentralized MPC. Note that the decentralized case is based on the calculation in Equation (5.1).

Looking at Figure 5.20, the centralized case, at least visibly, seems to result in better overall tracking. To quantify this notion, we can make use of the fortunate coincidence that the bar centerpoint reference in the $x$-direction is equal to zero at all times during the experiments. The error $e_x$ is thereby free from the transients otherwise occurring at each change in reference, and $e_x$ being non-zero is therefore solely a consequence of imperfect tracking.

Calculating the arithmetic mean* [BEE+05]

$$
\mu(X) = \frac{1}{T} \sum_{k=0}^{T-1} X_k,
$$

and the (sample) standard deviation [BEE+05]

$$
\sigma(X) = \sqrt{\frac{1}{T-1} \sum_{k=0}^{T-1} (X_k - \mu(X))^2}
$$

* $X$ in this case being a discrete time signal of length $T$, and $X_k$ the value of $X$ at time $k$. 

of $e_x$ leads to the results in Table 5.1. Perfect tracking would mean that $e_x$ were to be zero at all times. This is logically equivalent to $\mu(e_x) = \sigma(e_x) = 0$. By Table 5.1, neither of the two control systems is able to achieve such a tracking. However, the centralized MPC achieves a mean that is more than four times closer to zero than the decentralized MPC. Furthermore, the standard deviation of the centralized MPC is close to 50 percentage points lower, indicative of a less oscillatory behaviour. The superior performance of the centralized MPC is likely to be attributed its model, which captures more of the bar dynamics, including the effect of the other UAV.

Table 5.1: Statistics of the error in bar centerpoint position $e_x$.

<table>
<thead>
<tr>
<th></th>
<th>$\mu(e_x)$</th>
<th>$\sigma(e_x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized MPC</td>
<td>$-0.055\ 14\ m$</td>
<td>$0.058\ 46\ m$</td>
</tr>
<tr>
<td>Decentralized MPC</td>
<td>$0.234\ 68\ m$</td>
<td>$0.125\ 54\ m$</td>
</tr>
</tbody>
</table>
701 Results
Chapter 6

Conclusions and Future work

The aim of this thesis was to investigate the effectiveness and performance of model predictive controllers when applied to the case of collaborative payload transport using UAVs, and in particular as low-level control loops. For this purpose, two physical models were derived: one where the interaction forces caused by the payload were modeled as a disturbance (Offset-free MPC, section 3.2.2), and one where the interaction force had been explicitly calculated under the assumption that the payload was lifted using rigid links (centralized MPC, section 3.2.3). In order to reduce the computational complexity and enable real-time, online MPC calculations, both models were linearized, but with the yaw angles of each rigid body being left as parameters for increased generality. Although the system considered in this thesis were limited to two UAVs carrying a bar, a similar process should be possible for systems with more UAVs, and for arbitrary 6-DOF payloads.*

Simulation results were in line with what was hinted at in the introduction: by including an explicit expression of the interaction forces, the MPCs can generate control signals which lead to better performance. The disturbance model based MPC, although stable, resulted in a far more oscillatory trajectory of the payload, and furthermore, was substantially less effective in terms of damping and disturbance attenuation.

Motivated by the results from the simulation, the controller containing the explicit expressions of interaction forces was set up on an experimental platform and successfully was able to track a number of setpoints, which included translations in the $xy$-plane, as well as yaw rotations of all three rigid bodies. The MPC problem took on average 7 ms to solve, although occasional

* Care must be taken so that the UAVs are placed in a configuration so that cables remain stretched at all times. Otherwise, the rigid link assumption might not be equally valid.
spikes of times exceeding the 20 ms sample time were observed during the experiment.

As a final contribution, the possibilities of decentralizing the MPC were investigated, and a model in which the UAV–bar system was approximated as two independent UAVs with payload was developed and linearized in the same manner as done for the previous models. While still capturing the explicit expression of the cable tension, this formulation resulted in a far less complex expression. This controller was experimentally tested on the same platform as the centralized MPC, and although it was able to track and somewhat stabilize around setpoints, it did so with a more oscillatory and biased tracking of the bar position compared to the centralized MPC. However, it did result in an 2:5 times faster CPU time average, with all values being lower than the 20 ms sample time.

6.1 Future work

Although this work proved that MPC can in fact be utilized to successfully control a UAV–bar system like the one in Figure 1.3, there is still much that is left to be explored; both in terms of adjustments and additions in order to make the system usable under more realistic settings, and in terms of investigating if performance carries over to larger systems with more UAVs and more complicated payloads.

A first step towards a more realistic case would be to develop techniques of state estimation: the UAVs could be equipped with sensors in order to estimate their relative position or distance between one another. Furthermore, the state of the payload could be estimated using e.g. cameras, as done in [GCS17]. It is very likely, however, that the loss of precision caused by these state estimation techniques would mean that the controller proposed and utilized in this thesis would need to be modified in order to have a satisfactory performance. This modification could perhaps be to add disturbance terms to the model, or to robustify it using any of the techniques from the field of Robust MPC.

Similarly, some modifications likely have to be done to the controller developed in this thesis (apart from model modifications) if a system with more UAVs and a more complicated payload is to be treated. Further simplifications might have to be done in order to be able to run the controller at acceptable rates. Furthermore, the rigid link assumption might not be equally valid when more UAVs are added, since this increases the risk the cables not being taut at all times.

Finally, substantial progress could be made on the decentralization or
distribution of the original MPC problem. This area has currently seen few contributions for dynamically coupled systems: since all agents are interlinked, the optimal control signal for one agent will depend on the decisions of all the other agents. Although a decentralized attempt was done in this thesis, the performance could likely be improved by further investigation into e.g. if there are any quantities that should be transmitted between agents, and/or looking at if it is possible to incorporate more of the dynamics caused by the cable tension without making the models too complex.
Conclusions and Future work
References


<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title and Details</th>
</tr>
</thead>
</table>


Appendix A

Linearization and discretization

A.1 Linearization

In general, a system with state $Z$, inputs $u$ and an associated state evolution

$$\dot{Z} = f(Z, u), \quad (A.1)$$

can be linearized around a point $(Z_{eq}, u_{eq})$, by first performing a Taylor expansion

$$\dot{Z} = f(Z, u) = f(Z_{eq}, u_{eq}) + \left[ \frac{\partial f(Z, u)}{\partial Z} \right]_{Z_{eq}, u_{eq}} (Z - Z_{eq}) + \left[ \frac{\partial f(Z, u)}{\partial u} \right]_{Z_{eq}, u_{eq}} (u - u_{eq}) + O((Z - Z_{eq})^2, (u - u_{eq})^2). \quad (A.2)$$

Near the linearization point, we have

$$f(Z, u) \approx f(Z_{eq}, u_{eq}) + A_c (Z - Z_{eq}) + B_c (u - u_{eq}) =: f_{lin}(Z, u) \quad (A.3)$$

and if $(Z_{eq}, u_{eq})$ is an equilibrium point of the system, then

$$f_{lin}(Z, u) = A_c (Z - Z_{eq}) + B_c (u - u_{eq}) = A_c \Delta Z + B_c \Delta u \quad (A.4)$$

which is indeed a linear system.
A.2 Discretization

MPC, in most practical applications, makes use of discrete time models, and the linearized model in Equation (A.4) should consequently be discretized. A common discretization method is the ZOH [Åst97] method, in which the system input is assumed to be constant in between sampling times. The resulting discrete time model is

$$\dot{Z} = \Phi \Delta Z + \Gamma \Delta u,$$  \hspace{1cm} (A.5)

in which

$$\Phi = \Phi_{ZOH} = \exp(A_c h)$$

$$\Gamma = \Gamma_{ZOH} = B_c \int_0^h \exp(A_c s) \, ds,$$  \hspace{1cm} (A.6)

and $h$ is the sampling time.

In some applications, the ZOH sampling can become impractical because of the need to calculate the matrix exponential. This can, for example, be the case when $A_c$, for purposes of generality, is desired to be dependent on a parameter. The ZOH discretization can then instead be approximated up to an order $K$ by utilizing the matrix identity

$$\exp(X) = \sum_{n=0}^{\infty} \frac{X^n}{k!}$$  \hspace{1cm} (A.7)

in Equation (A.6) and discarding any terms that are in $O(h^{K+1})$. This results in the approximations

$$\tilde{\Phi}^{(K)} := \sum_{n=0}^{K} \frac{A_c^k h^k}{k!}$$

$$\tilde{\Gamma}^{(K)} := B_c h \sum_{n=0}^{K-1} \frac{A_c^k h^{k+1}}{(k+1)!}.$$  \hspace{1cm} (A.8)

Interestingly enough, the first order approximation coincides with the Euler forward method

$$\tilde{\Phi}^{(1)} = \Phi_{EF} = I + A_c h$$

$$\tilde{\Gamma}^{(1)} = \Gamma_{EF} = B_c h,$$  \hspace{1cm} (A.9)

and the fourth order approximation with the fourth order Runge–Kutta (RK4) method

$$\tilde{\Phi}^{(4)} = \Phi_{RK4} = I + A_c h + A_c^2 h^2 + \frac{A_c^3 h^3}{2} + \frac{A_c^4 h^4}{24}$$

$$\tilde{\Gamma}^{(4)} = \Gamma_{RK4} = I h + \frac{A_c h^2}{2} + \frac{A_c^2 h^3}{6} + \frac{A_c^3 h^4}{24}.$$  \hspace{1cm} (A.10)
Appendix B

UAV–bar linearized model

The linearization described in Section 3.2.3 yields the ODE
\[
\begin{aligned}
\dot{\psi}_b &= g(p_{1x} + p_{2x} - 2p_{bz}) \\
\dot{\psi}_b &= g(p_{1y} + p_{2y} - 2p_{bz}) \\
F_1 + F_2 + g m p_{1z} + g m p_{2z} + g m_b p_{1z} + g m_b p_{2z} - 2g m_b p_{bz} \\
&= \frac{l(2m + m_b)}{2l} \\
R_z^{-1} (\psi_0^{(b)}) &\omega_b \\
&= \frac{d_b \cos(\psi_0^{(b)}) (F_1 - F_2 + g p_{1z} + g p_{2z} + g m_b p_{1z} - g m_b p_{2z} - 2d_b g m_b \phi_b + 2d_b g m_b \phi_b)}{l(2md_b^2 + I_b)} \\
&- \frac{d_b \sin(\psi_0^{(b)}) (F_1 - F_2 + g p_{1z} + g p_{2z} + g m_b p_{1z} - g m_b p_{2z} + 2d_b g m_b \phi_b + 2d_b g m_b \phi_b)}{l(2md_b^2 + I_b)} \\
&- \frac{d_b g m_b (2d_b \psi_0^{(b)} - p_{1x} \cos(\psi_0^{(b)}) + p_{2x} \cos(\psi_0^{(b)}) + p_{1x} \sin(\psi_0^{(b)}) + p_{2x} \sin(\psi_0^{(b)}))}{2I_b l} \\
\dot{v}_1 &= \ddot{\phi}_1 \\
\dot{\phi}_1 &= \psi_1 \\
\omega_{\phi,1} &= \text{diag}(I_x, I_y, I_z)^{-1} \tau_1 \\
\dot{v}_2 &= \ddot{\phi}_2 \\
\dot{\phi}_2 &= \psi_2 \\
\omega_{\phi,2} &= \text{diag}(I_x, I_y, I_z)^{-1} \tau_2 \\
\quad \text{(B.1)}
\end{aligned}
\]

where a symmetric system

\[
\begin{aligned}
d_1 &= -d_2 = -d_b \\
l_1 &= l_2 = l \\
m_1 &= m_2 = m
\end{aligned}
\quad \text{(B.2)} \quad \text{(B.3)} \quad \text{(B.4)}
\]

has been assumed, and
\begin{align}
\dot{\phi}_1 &= \frac{g m_b P_{bx}}{2 l m} - \frac{g m_b P_{1x}}{2 l m} + g \theta_1 \cos \left( \psi_0^{(1)} \right) \frac{m + m_b}{2} \\
&\quad + \frac{g \phi_1 \sin \left( \psi_0^{(1)} \right) \left( m + \frac{m_b}{2} \right)}{m} + \frac{d_b g m_b \psi_0^{(b)} \cos \left( \psi_0^{(b)} \right)}{2 l m} \tag{B.5} \\
\dot{\theta}_1 &= \frac{g m_b P_{by}}{2 l m} - \frac{g m_b P_{1y}}{2 l m} - g \theta_1 \cos \left( \psi_0^{(1)} \right) \frac{m + m_b}{2} \\
&\quad + \frac{g \phi_1 \sin \left( \psi_0^{(1)} \right) \left( m + \frac{m_b}{2} \right)}{m} - \frac{d_b g m_b \psi_0^{(b)} \sin \left( \psi_0^{(b)} \right)}{2 l m} \tag{B.6} \\
\dot{\psi}_1 &= \frac{F_1 \left( I_b + 4 d_b^2 m + d_b^2 m_b \right)}{2 m d_b^2 + I_b} \frac{(2 m + m_b)}{(2 m + m_b)} + \frac{F_2 \left( I_b - d_b^2 m_b \right)}{(2 m d_b^2 + I_b)} \frac{(2 m + m_b)}{(2 m + m_b)} \\
&\quad + \frac{g p_{2z} \left( I_b - d_b^2 m_b \right) \left( m + m_b \right)}{l \left( 2 m d_b^2 + I_b \right) (2 m + m_b)} + \frac{g m_b P_{bz} \left( m + m_b \right)}{l m \left( 2 m d_b^2 + I_b \right) (2 m + m_b)} \\
&\quad + \frac{g p_{1z} \left( m + m_b \right) \left( m m_b d_b^2 + I_b m + I_b m_b \right)}{l m \left( 2 m d_b^2 + I_b \right) (2 m + m_b)} \\
&\quad - \frac{I_b d_b g \phi_b \left( m + m_b \right)}{l m \left( 2 m d_b^2 + I_b \right)} \tag{B.7} \\
\dot{\phi}_2 &= \frac{g m_b P_{bx}}{2 l m} - \frac{g m_b P_{2x}}{2 l m} + g \theta_2 \cos \left( \psi_0^{(2)} \right) \frac{m + m_b}{2} \\
&\quad + \frac{g \phi_2 \sin \left( \psi_0^{(2)} \right) \left( m + \frac{m_b}{2} \right)}{m} - \frac{d_b g m_b \psi_0^{(b)} \cos \left( \psi_0^{(b)} \right)}{2 l m} \tag{B.8} \\
\dot{\theta}_2 &= \frac{g m_b P_{by}}{2 l m} - \frac{g m_b P_{2y}}{2 l m} - g \theta_2 \cos \left( \psi_0^{(2)} \right) \frac{m + m_b}{2} \\
&\quad + \frac{g \phi_2 \sin \left( \psi_0^{(2)} \right) \left( m + \frac{m_b}{2} \right)}{m} - \frac{d_b g m_b \psi_0^{(b)} \sin \left( \psi_0^{(b)} \right)}{2 l m} \tag{B.9}
\end{align}
\[
\psi_2 = \frac{F_2 \left( I_b + 4d_b^2 m + d_0^2 m_b \right)}{(2md_b^2 + I_b) (2m + m_b)} + \frac{F_1 \left( I_b - d_0^2 m_b \right)}{(2md_b^2 + I_b) (2m + m_b)} \\
+ \frac{g p_{1z} \left( I_b - d_0^2 m_b \right) (m + m_b)}{l (2md_b^2 + I_b) (2m + m_b)} + \frac{g m_b p_{bz} (m + m_b)}{lm (2m + m_b)} \\
- \frac{g p_{2z} (m + m_b) \left( m m_b d_0^2 + I_b m + I_b m_b \right)}{lm (2md_b^2 + I_b) (2m + m_b)} \\
+ \frac{I_b d_b g \phi_b (m + m_b)}{lm (2md_b^2 + I_b)}.
\]