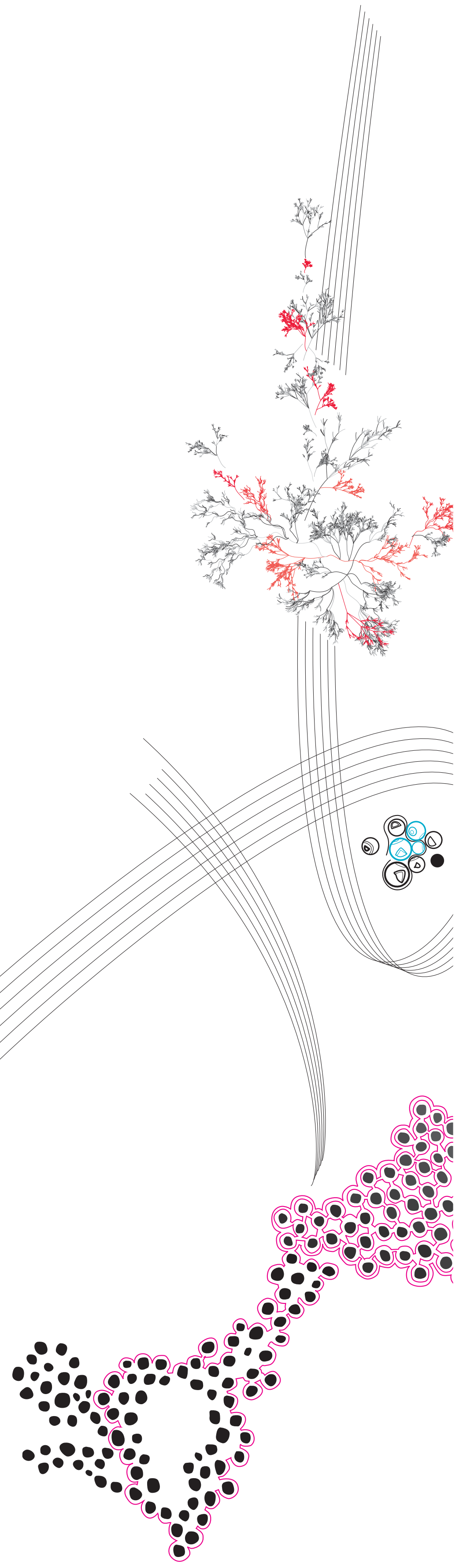
MSc Systems & Control
Thesis



MPC design for a precision, flexture based, pendulum

Jasper van der Schaaf

Examination committee:
Dr. ir. W.B.J. Hakvoort
ir. W. van Dijk
prof.dr.ir. A. Franchi

September 7, 2023

Chair of Precision Engineering
Department of Mechanics of Solids,
Surfaces and Systems (MS3)
Faculty of Engineering Technology
University of Twente

UNIVERSITY OF TWENTE.

Abstract

In this thesis a model predictive control (MPC) controller for a single arm of a precision 6-dof hexapod is developed.

The controller will be developed for a single arm of the T-flex. The T-flex is a fully flexture based precision hexapod developed by the Precision Engineering group of Universiteit Twente. It is equipped with high precision sensors and high quality motors. This results in a system for which a precise model with little error can be obtained and where measurement errors don't play a significant role.

The goal is to develop a MPC implementation that is able to control a single arm of the T-flex, using the ability of MPC to explicitly take into account the state restrictions of the joint and the thermal limits of the electric motors.

The first part of the theory chapter will explain MPC in general, to be followed by highlighting some theory and sources that proved relevant for the used implementation. The chosen MPC implementation uses MPC as a path planner for the nominal system, with a hybrid feedforward and computed torque control setup to translate the MPC planned trajectory to motor torque commands. A LQR feedback controller is used to reject disturbances and model errors. The model that is used for the hybrid feedforward and computed torque control is found using a parametric model of the system that is shown to work in previous work.

To verify the chosen control setup, simulations are performed to compare the performance of the MPC control setup with a setup that combines a feedforward and PID controller combination, using references that adhere to the steady state limits of the T-flex. These simulations demonstrate that, in unconstrained situations, the performance of both methods is comparable and that the thermal constraints of the T-flex are unlikely to become relevant in practice. A simplified system is then specified with parameters that are chosen to ensure the thermal constraints become relevant, to demonstrate that the MPC setup is superior in constraint situations. The simulations includes mild model errors and realistic disturbances to demonstrate that the chosen MPC control setup is robust under these conditions.

A second set of experiments is performed to explore controller behavior for different ways MPC can be combined with a fb controller. These experiments use the simplified model to show the behavior of the MPC controller if the fb controller is included in the model of the system it controls. Showing an example of the problems that can arise if MPC is combined with a controller whose objective functions do not share the same objective.

Contents

1 Introduction

The PE control group of Universiteit Twente has developed a fully flexture based 6-dof hexapod called the Tflex [19]. Its mechanical design combined with high precision encoders and high performance motors make it possible to fully control a platform with high speed and μm precision. Currently the reference generation used to move between setpoints is limited because it does not take into account the current state of the Tflex. Instead, it is designed to never exceed the steady state limits of the work space of the system. This means the performance might be improved by taking the current state of the system into account, such that only the limits relevant for the current state need to be considered, instead of all restrictions of the entire workspace. In this thesis the option to use MPC to control the Tflex will be explored. MPC is chosen because it is a control technique which is able to perform online optimisation while explicitly taking constraints into account [17], and can be used with nonlinear systems. This thesis will focus on generating an optimal reference, taking into account the current state of the system and the constraints.

Model Predictive Control (MPC) is a control technique which started in the process industry. It solves a finite horizon open-loop optimal control problem at each sampling time, taking the measured state of the system as initial state.[21]

It is a model based control method, meaning it optimizes the control actions for the model of the system it has been provided. Recently it has gained attention for its applications in the field of robotics. This interest has increased due to advances in the amount of available computational resources and improvements in algorithms used to solve these optimisation problems. This enabled MPC implementations to meet the more stringent real time performance criteria for robotic applications, which often have much faster system dynamics compared to the process industry [17]. The ability of MPC to explicitly account for constraints makes it interesting for applications where it is important to avoid certain states, or resources are constraint in some important way. Two examples of the first are self driving cars and dynamical obstacle avoidance for UAVs [11][13]. Despite computational improvements, real time feasibility remains a major design constraint for practical MPC implementations, This has resulted in nominal or tube based MPC being the main way practical demonstrations have been achieved. This means that only the nominal system is considered in the MPC problem, where in the tube MPC cases the MPC controller also considers the uncertainty of the system. Deviations from this nominal trajectory are then considered disturbances and are dealt with using a lower level controller, ensuring that that system robustly follows the reference generate by MPC.

The main contributions of this paper will be twofold. First it will propose a MPC implementation for the T-flex, and compare it with the previous trajectory generation method. Secondly, different ways to handle constant disturbances will be discussed and compared.

A final not about this thesis, the original goal was to also perform experiments on the physical T-flex, due to issues explained in C, this did not happen.

1.1 Relevant Related Work

1.1.1 MPC stability and robustness

Mayne et al. have published several papers about proofing the stability and robustness of MPC. Two of which are especially relevant for this thesis. The first paper [17] extends the stability proof that already existed for linear systems to nonlinear systems. This paper is relevant because it shows MPC can be proven to be stable for nonlinear systems, which the T-flex is. The second paper [18] proposes and demonstrates a practical implementation of tube MPC, and demonstrates a proof that the implementation is stable for certain setups. The tube MPC implementation in this paper uses a second MPC controller to keep the system within a range of states (the tube) around the nominal system, with the nominal system being controlled by the first MPC controller. This paper is relevant because it is cited as the basis for other practical robust nonlinear MPC (NMPC) implementations, such as [14].

1.1.2 Tube MPC

Brett T. Lopez et al. [14] proposed dynamic tube MPC (DTMPC), which allows for the online parameterization and optimisation of the uncertainty tube. The tube considers state dependent uncertainty and how it is affected by changing the bandwidth of the boundary layer sliding mode controller, which is used as ancillary controller. This is notable because, as the paper notes, the state dependence is usually simplified to reduce the computational requirements of the optimisation problem. This makes it possible to optimize the resource allocation between the nominal control and disturbance rejection online. The main relevance of this paper is that it represents recent progress made in the development of robust MPC formulations. It also provided a different insight about a how a feedback controller can be combined with a MPC controller that controls the nominal trajectory.

1.1.3 Implementation and Performance

In a paper by B Lindqvist et al. [13] a MPC implementation for UAV as a local trajectory planner is demonstrated. In this application MPCs ability to explicitly take constraints into account is used to plan trajectories that avoid collisions with dynamic obstacles. The authors demonstrate it is possible to use nonlinear MPC (NMPC) in a real time application. Although used as trajectory generator without explicitly considering control actions. It still shows that even with relatively old hardware, a laptop processor from 2012, it is computationally feasible to combine the slow sample rate of mpc with a higher frequency feedback controller in a practical application.

When finalising this thesis the author became aware that the UAV control literature has a relatively large body of work describing practical MPC implementations, including implementations that combine feedback linearisation, mpc and disturbance observers. Some of the implementations are very similar to the one proposed in this thesis. A article by D Ma, Y Xia, T Li and K Chang [15] is such an example. The control problem is separated into a reference generation, which is handled by the MPC controller, and reference tracking problems, which are handled by a active disturbance rejection controller using and extended state observer. A comparison is also made with a combined MPC and H_∞ controller presented in [22]. The main relevance of these articles for this thesis is that they cover methods which are likely better ways to handle constant disturbances, compared to the method used in this thesis.

1.2 Thesis Structure

The main body of this thesis is divided into four parts. The first part is chapter 2, which will cover and reference MPC theory relevant for this thesis. Chapter 3 will detail specific choices, and explain why these have been made. Chapter 4 explains the specific control implementations for the two simulations. It will also show and discuss the results. The final chapters 5 will be used to draw a conclusion.

2 Theory

This section covers MPC theory relevant for this thesis. The information is gathered from different sources. While most sources go significantly more in depth about their respective topics, the information presented here should be sufficient to understand the chosen MPC control setup, and provide the basis understand why certain choices where made. To start, a short description of the set of general optimisation problem, of which MPC is a specific formulation, will be given.

2.1 The MPC Optimisation Problem

Model predictive control is a control technique that solves a constrained optimisation problem at each sample time. The constrained optimisation problem that has to be solved has the following general form,

$$\begin{aligned} \min_{z} \quad & \mathbb{J} = f(z) \\ \text{s.t.} \quad & h_i(z) \geq 0 \quad \text{for } i = 1, ..., m \\ & g_j(z) = 0 \quad \text{for } j = 1, ..., p \end{aligned} \tag{1}$$

where $z \in \mathbb{R}$ is the controlled variable and $\mathbb{J}^*$ is the minimum value of $\mathbb{J}$, h_i and g_j are respectively the set of inequality and equality constraints.

To make solving the problem computationally tractable, the problem is solved only for a certain horizon, which can be fixed or changing. The problem is solved iteratively, meaning the problem is updated using new information and re-solved every t_s (sample time) seconds. [23]

A key design problem with MPC appears to be the balance between the accuracy of the problem that is solved compared to the computational complexity. Because ignoring any computation constraints, there is no reason to not solve the full, potentially nonlinear and non convex, control problem for the full or infinite horizon using a vanishingly short sample time. The subsection where the trade-offs between accuracy and performance are clearest is 2.5, but it will be a theme throughout this chapter.

2.2 MPC and Control

Equation (1) is a general formulation for a constrained optimization problem. It can be rewritten to be more suitable for control applications. For control the full optimisation problem is usually a discretised version of the true real-time problem, because the controller needs to be implemented usually using a digital controller. The control problem also uses the system states $x \in \mathbb{X} \subseteq \mathbb{R}^{n_x}$ and controller actions $u \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$, where $\mathbb{X}$ and $\mathbb{U}$ are the sets of feasible states and actions respectively. As already mentioned, only a limited horizon of the full optimal control problem is considered to limit the computational requirements. The optimisation problem can be solved either online or offline. With an offline solution the state space is split into regions, with each region having its own solution. While finding a solution offline might seem attractive, as the optimisation problem no longer needs to be solved online. For nonliner control problems with a large horizon, it might be slower to find in which region of the solution the current state lies, then it is to solve the nlp problem for the current location [26].

$$\begin{aligned} \min_{u} \quad & \mathbb{J} = \sum_{k=0}^{N} f_k(x_k, u_k) \\ \text{s.t.} \quad & h_i(x, u) \geq 0 \quad \text{for } i = 1, ..., m \\ & g_j(x, u) = 0 \quad \text{for } j = 1, ..., p \end{aligned} \tag{2}$$

A description of the function f from equation (2) is still missing. It consists of two parts, the actual cost-function and information about the dynamics of the problem, which are needed to predict the states that are used by the cost function. First, the plant dynamics will be addressed, which covers how they are integrated into the optimisation problem. Two methods to include the system dynamics in the objective will be described in the next section.

2.2.1 The MPC Objective Function

Before the system dynamics can be integrated into the optimisation problem they first need to be derived. For this chapter it is assumed that this model is already available in this form: $x_{k+1} = p(x_k, u_k)$, with $p(x, u)$ being the function describing the system dynamics. It is also assumed a cost function in the form of $g(x_k, u_k)$ already exists. For now g can be any function that takes a state and action pair and returns the cost associated with it. The integration of the dynamics into the cost function can be done in several ways, but two common methods are single and multiple shooting. These and a third set of methods are shortly described in chapter 4 of [23]. For single shooting the plant dynamics are directly inserted into the cost function.

$$\begin{aligned}
&f_0 = g(p(x_0, u_0),\ u_0)\\
&f_1 = g(p(p(x_0, u_0), u_1),\ u_1)\\
&f_2 = g(p(p(p(x_0, u_0), u_1), u_2),\ u_2)\\
&...
\end{aligned} \tag{3}$$

Multiple shooting instead treats each time step a separate optimisation problem. This is done by making the states at each time step as a decision variable and adding the constraints that the states of subsequent time steps adhere to the system dynamics.

$$\begin{aligned}
&f_0 = g(x_0, u_0)\\
&f_1 = g(x_1, u_1)\\
&...\\
\text{s.t.}\quad &x_0 = x_{measured}\\
&x_{k+1} - p(x_k, u_k) = 0 \quad \text{for } k = 1, ..., N-1
\end{aligned} \tag{4}$$

To better show what is happening, figure (1) uses a double integrator system to illustrates what it looks like if subsequent shooting nodes don't adhere to the system dynamics yet.

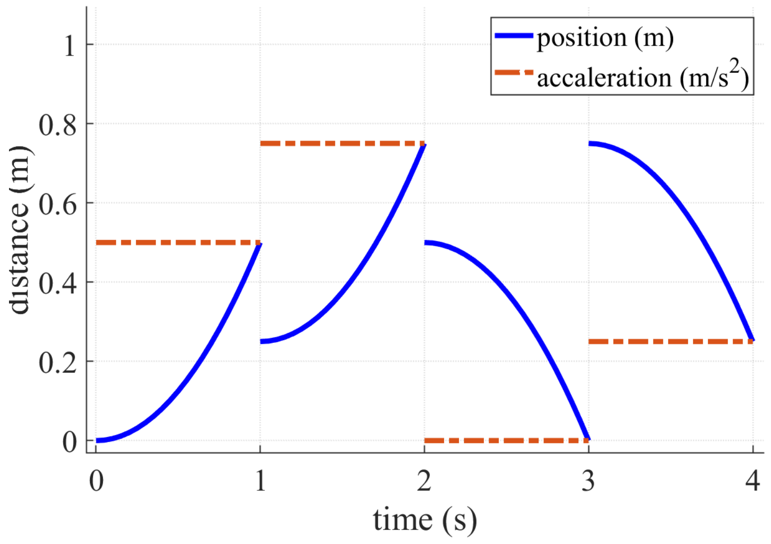


Figure 1: Figure showing multiple shooting without system constraints. Acceleration is relative to the start position, $ts = 1$ and $k = 1 : 4$.

As [23] explains, although multiple shooting adds additional decision variables and constraints, in practice it is usually faster for longer horizons, while in the authors experience not being slower for smaller horizons. This is because the single shooting method can have a complicated relation between the input u and the state at a later time. Obvious examples of systems where the states at some time step $ts + N$ can be very sensitive to changes to some past time ts are chaotic systems. This high sensitivity between decision variable and output often causes problems for the numerical solvers used to find solutions. Multiple shooting on the other hand has a relatively much simpler relation between the decision variables and costs. The speedup from a faster convergence outweighs the additional computation required from the additional states. The increase in constraints and decision variables is also not as bad as one might initially expect. The reason for this is that multiple shooting results in a sparse problem. Meaning that the matrices describing the costs and constraints as a function of the decision variables contains a lot of zeros, this structure can be exploited by solvers reducing the computation requirements for these kind of problems [5]

2.3 Optimization

This thesis has so far only described how a MPC problem can be written as an optimisation problem, but this optimisation problem still needs to be solved. This is where the connection with mathematical optimisation becomes relevant. Because it are mathematical solvers that are used to solve the optimization problem posed by MPC. Although it is not required to fully understand how the solvers work, having a basic understanding about how they function, and the trade offs they make is can be very useful. This sections aim is to provide a starting point to the reader about the some of the concepts that can be relevant for MPC. For more information about numerical optimisation algorithms, the book [27] is an excellent source. Most of the information in this chapter is sourced from this book.

To start, a clear description of the problem the optizers try to solve. There is some cost that needs to be minimised, which is given by the cost function. The problem is that, although the function is known, no analytical solution for the minimiser of the function is available. This minimizer has to be found by trial and error. Exhaustively sampling the function and then picking the minimal value is possible, but generally not computationally feasible. Instead some procedure needs to be used to efficiently look for the minimizer. There are two broad classes of procedures, Hueristic based methods and gradient based methods. Heuristic methods follow some global heuristic to try to efficiently find the global optimum. Gradient based methods instead try to follow the cost landscape to find a local minimum.[27]

Depending on the problem either a linear, quadratic or non-linear programming problem needs to be solved. With the problem being a nonlinear one if either the system dynamics or boundary constraints are nonlinear. Both linear and quadratic problems can usually be solved quickly and solutions found are, or within in some bound of, the global optimum. the reason for this is because linear problems only have one minimum, the same is true for quadratic problems if the cost matrix is positive semi-definite [8].

The book [8] states that: this is not generally true for nonlinear problems. And although is possible to obtain the global minimum for nonlinear problems, this is generally considered to be to computationally expensive for MPC applications. Instead local minima need to, and generally do, suffice. Not all nonlinear problems have multiple local minima, but it can be difficult to determine if this is the case for any specific problem.

Most solvers used for MPC application seem to be numerical local ones. The book [27] states that these start from some initial guess, and information about the solution at that initial guess to take a step in a direction. With the step hopefully improving the guess. This process is then repeated until a stable point is reached. Ideally the stable point is the global minima, but it can also be a local minima or saddle point.

Generally any information about the function for which a minimizer is desired is not freely obtained, meaning the goal of algorithms is to reliably find good steps, without relying on information that is computationally expensive to obtain [27]. Algorithms that might be robust for one set of problems, might not be for a different set. This means that which algorithm is the best choice depends on the application and problem [5]. With MPC, getting a solution quickly is important, but it should also find a solution reliably. If the application is not real time, a slower algorithm but one that is more likely to find a better local minimum might be preferred.

2.3.1 Warm Starting

Because the optimizers perform some form of gradient decent from some initial guess, their performance can be improved by providing a good starting point from which they search for a solution. This is often referred to as warm starting. The book [23] provides roughly the following explanation. It can often be assumed that the problem solved at each iteration is almost the same problem. At $ts = t$ the problem $\sum_{k=ts}^{N} f_k$ is being solved, if the goal does not change, the same problem has to be solved at $ts = t+1$, just with a shifted horizon. Assuming the prediction at ts is perfect and the new horizon does not influence the previous solutions, the solution for time steps $ts = 2 : N$ have not changed. Meaning that using the old solutions for time steps $ts = 2 : N$, as the solutions for the current time steps $1 : N-1$, already provides the solver with a good guess from which to start its search. This can greatly reduce the computation time required to find a solution.

Different algorithms benefit from warm starting to a more or lesser extend [23]. There are several strategies to initialise the new time step like using time step $N-1$, but in practice it makes little difference how this state is initialised. The procedure that implements the warm starting is usually called the shifting operation.

2.3.2 discretization

MPC usually optimizes a discretised version of a continuous time problem. Often the system dynamics are known as a continuous time ordinary differential equation (ode). What is required for the MPC problem is a function that takes the state at time t, and returns the state at time $t{+}1$. This is a common problem and multiple methods are available, with two well known examples being the forward Euler method and the Runge-Kutta methods. It is important that the discretization used is both accurate, and does not require too much compute. It needs to be accurate as otherwise the solver might arrive at an incorrect solution or fail to converge [24].

To close, the statements made in the paragraph above are also true for calculating the cost to go. Generally the same method for the cost calculation is used as the system dynamics [24].

2.4 Stability, Safety and Robustness

Stability, safety and robustness are three important aspects of control. It is important the system behaves in a predictable way and does not damage itself or its surroundings. The review paper [17] provides an overview of the, then known, stability and robustness proofs, while Chapter 5 of the book [9] also covers more recent developments, such as in nonlinear MPC. The purpose of this chapter is repeat some of the key concepts behind the stability and robustness guarantees, and use these concepts to explain some of the possible options that are available to obtain these guarantees.

2.4.1 Stability

To start with the explanation why MPC is stable, a repeat from [21] of the requirements for stable infinite horizon optimal control. The first requirement is that the problem needs to be feasible. This can be difficult to prove, but for the experiments in this thesis this is true, unless stated otherwise. The second assumption is that the cost function is positive definite. Because usually the cost function is a design choice, as it is in this thesis, this can be made to be true. Note that although the cost function is positive definite, the costs are positive semi-definite. The cost function will be described in section 3.2.3.

If the system is detectable and stabilizable and optimal infinite horizon controller is stabilizing[21]. To understand why a few ingredients are required. First there exists some policy π which generates actions u based on the current state x. In this thesis this policy is the MPC controller, meaning it is time and state invariant. To be clear, the actions u the MPC controller generates do differ for each state, but the policy that is generating those state dependent actions is not. The policy π_* is the optimal policy, with the $*$ indicating it is optimal. Meaning it chooses the optimal action u_* in each state. Also recall $\mathbb{J}$ is the cost to go according to the MPC algorithm. In this chapter $\mathbb{J}_\pi$ will be used to indicate the cost to go actually depends on the policy use and V^∞ will be used with the ∞ superscript to indicate it refers to the infinite horizon cost. This allows the construction of the function $V_\pi(x)$ which is relates to $\mathbb{J}_\pi$,

$$V_\pi^\infty(x) = \mathbb{J}^N(x,u) + V_\pi^\infty(x+N) \tag{5}$$

the function $\mathbb{J}_\pi^N(x)$ returns the cost to go using policy π from state x for N time steps. From Bellman's principle of optimality it is known that if $V_\pi^\infty(x+N)$ is known, optimal and the policy π chooses a u which minimize $\mathbb{J}_\pi^N(x) + V_\pi^\infty(x)$, that the policy π is optimal even if $N = 1$ [6] [7].

The above can be combined with the requirement that the problem is feasible and the cost function is positive definite, to conclude that $l(x, u^*) \geq l(x_{goal}, u^*) \; \forall \; x \neq x_{goal}$. In words it means that, as long as the current state is not the goal, there exists an action which reduces infinite horizon cost. Because the lowest cost is at the goal, it means that an infinite horizon optimal policy approaches the goal and can be used as a Lyapunov function.[17]

$$V_\pi^\infty(x+1) \leq V_\pi^\infty(x) \tag{6}$$

Note that it is only required for the policy to chose an action reducing the infinite horizon costs, meaning this proof does not require the chosen policy to be optimal.

Sadly the infinite horizon stability guarantee does not automatically extended to a finite horizon optimal control problem. The challenge in proving stability for the finite horizon optimal control problem stem from its limited horizon. Because the horizon is limited and finding $V^{\infty *}$ is not trivial, if it was there would be no or little need for the finite horizon formulation, the controller can only be assumed to have knowledge of $\mathbb{J}_\pi^N(x)$. This is an issue because there is no guarantee that an action minimizing the cost within the horizon does not result in a bigger cost beyond the horizon. An example of the limited horizon causing a constraint violations can be seen at the end of the following subsection in figure 2. In other words $\pi_*^\infty \neq \pi_*^N$ without additional guarantees. This means that the inequality 6, which is the basis for the stability guarantee for the infinite horizon, no longer holds. There are several ways to reintroduce this guarantee for limited horizon optimal control [**mayne_Stab_robust**].

The first solution is to simply add the condition that the goal state has to be reached at the terminal state of the horizon, resulting in the equality.

$$V_{\pi^N}^\infty = \mathbb{J}_{\pi^N}^N \tag{7}$$

Because $\mathbb{J}_{\pi^N}^N$ is known it can be substituted in the original inequality equation 5 and the proof is restored. The policy π has also been replaced by the policy π^N, because all policies not reaching the goal in N steps are excluded. It should be clear that this can lead to less optimal solutions, especially in cases where actions are more costly.

Other downsides are that it can significantly reduce the set of initial states for which the problem is feasible, as only states that can reach the goal within the horizon are still feasible. The reachability problem can be partially mitigated by extending the horizon until the goal lies within it. But this can become impractical, as it can take significant time to work with a long horizon. [23]

The second solution is a variation on the previous method. But instead of requiring the terminal state to be the goal state, the terminal state is restricted to a set of states that are known to be stabilizing to the goal. A way this can be achieved is to design a stabilising controller that stabilises the system around the goal. With this setup MPC is used to steer the system towards the region of attraction of the controller, at which point the controller takes over. [23]

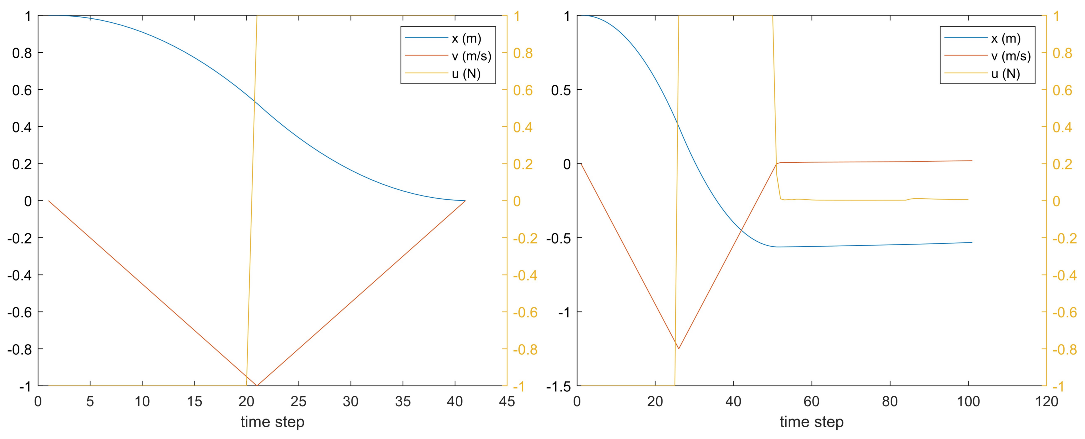
A third solution is by introducing a terminal cost that approximates the infinite horizon optimal control cost of the terminal state. In short, it works because the terminal cost ensure the current actions won't cause bigger future costs, because these future costs are captured in the value of the terminal state. It is also possible to directly incorporate this knowledge about the infinite horizon cost into the cost function used for the problem. This difference between the used terminal value function and the actual terminal state value does mean that the solution is no longer guaranteed to be optimal. [12]

2.4.2 Safety

To avoid confusion, in this thesis safety refers to constraints that are present to avoid states that are undesired for some reason. This could be the location of other cars if the application is self driving cars or humans and other obstacles for robotic arms, but also thermal states that would damage the system being controlled. Safety should be ensured by safety constraints added to the optimisation problem. However the limited horizon also limits the guarantees these provide. To illustrate the problem, take a simple moving mass system as an example. The system is described by the set of ode equations (8). With m being the mass and F the force exerted on the mass, which is chosen by controller.

$$\begin{aligned} x &= \dot{x} \\ \dot{x} &= F/m \end{aligned} \tag{8}$$

The cost function is $(x - x_{setpoint})^2$, the constraint is $x \geq 0$, the start position is $x = 1$ and the goal is $x = 0.01$. In figure 2 trajectories with and without kinetic safety constraints are shown. The reason why the safety constraint is required is due to the short horizon, which is only 5 time steps. This causes a problem because the optimal solution with this cost function and constraints is a second order motion profile. But the mpc controller can only detect that it will overshoot the setpoint and violate the constraint after the deceleration point of the optimal motion profile. The safety constraint will prevent this problem by restricting states to the states that are still able to be decelerated before hitting the constraint as shown in figure 2b. Casadi v3.5.5 [5] with the Ipopt solver is used for this example. The kinetic energy had to be restricted with a position limit of $x = 0.02$, it seems that with the settings used small constraint violations are allowed.



(a) A trajectory with kinetic safety constraints (b) A trajectory without kinetic safety constraints

Figure 2: Comparison between trajectories with and without safety constraints with $N = 5$

Exactly how the safety constraints are implemented depends on how stability and the disturbances are handled. If the terminal condition $x_N = x_{goal}$ is used safety is already guaranteed. The reason is the same as how stability is ensured. If the terminal state is the goal state, the system is guaranteed to end in a safe state, as long as the goal is save, which it is if the problem is feasible. A different solution to prevent safety constraint violations, is to find an horizon independent description of the set of safe states, and add the constraint that the final state must lie in this set. The appendix **??** contains a derivation of these safe sets, it also gives a specific example and explanation about the difficulties associated with this approach. Further care has to be taken if uncertainty is considered, but how this affects the constraints will be discussed in the next section.

2.5 Robustness

Having provided methods to ensure both stability and safety, a method that ensures this remains true under some uncertainty is also required. As in practice there is always some noise and the system parameters are never exactly what they are expect to be. The purpose of this section is to provide a basic understand of the difficulties of robust MPC implementations. This will be done in two parts. First the stochastic optimisation problem will be described, followed by an explanation why it increases the computational complexity of the optimisation problem. The second half of this section will reference some sources to quickly describe ways the increased computational complexity can be reduced to allow for real time solutions.

To understand why uncertainty is problematic for MPC, remember the sources of uncertainty and their effects on the prediction of future states, also recall the goal of MPC is to find the actions u that will maximise the value $\mathbb{J}$. Because of the uncertainty the optimisation problem becomes a stochastic optimisation problem, with the objective to maximize the expected value. This means the goal is to maximise the value of the sum of all possible realisations, multiplied by their likelihood of occurring.

For now the question as to what happens with constraints will be left open, instead the focus will be on the computational difficulties of this stochastic optimisation problem.

There are several strategies that can be used to reduce the computational burden stochasticity introduces. The Stochastic Model Predictive Control chapter of [23] lists a few and is a good source if the reader is interested in a more detailed overview. This thesis only considers tube MPC, because more papers describing its implementation where found, and the method lends itself better to the engineering approach eventually used as in section , where the tube size is constant and its size is based on the parameter estimation experiments.

Before providing a description of tube MPC the stochastic system needs to be properly defined first. The assumed disturbance model is,

$$\dot{x} = p(x, u) + \hat{p}(x, u, w) \tag{9}$$

where $\hat{p}(x, u, w)$ represents the effects of unmodeled behavior, due to modeling errors, and random disturbances represented by the stochastic variable $w \in \mathbb{W}$, on the evolution of the system.

Tube MPC considers the nominal system p and tries to find a parameterization that describes the outer edge of the states that can be reached due to the uncertainty $\hat{p}$ around this nominal system.

An article by Mayne et al. [16] also contains a short overview of the stochastic MPC research landscape at the time. The paper adds distinctions between the assumptions that can be made about the influence of the controller on the system uncertainty evolution. The paper describes three ways of the MPC controller on the tube shape can be categorized. The first is to ignore the influence and treat the controller as an open loop controller, assuming it takes no actions to try to counteract disturbances. This is computationally the easiest approach, but it is pessimistic because the controller will take actions to reduce the influence of disturbances. This means the tube can quickly grow large enough that the problem becomes infeasible. The remaining two categories do account for the effect feedback has on the tube geometry, but do it in different ways. One directly considers the actions the controller takes to reduce the tube size, while the last method considers a policy which will take actions to reduce the tube size.

An different paper by Mayne et al. [18] uses tube MPC to separate the design of the controller controlling the nominal system x_{ref} and the controller rejecting disturbances $x - x_{ref}$. In the proposed solution the fb controller is assumed to be a second mpc controller sharing the same cost function as the main controller, but for the error between the nominal trajectory and the actual trajectory. This does not solve one of the problems that nmpc can suffer from, namely its long computation time.

A more recent paper [14] proposes to parameterise a feedback control policy, instead of a second MPC controller. It then parameterises the tube based on the fb parameters, and lets the MPC controller controlling the reference also control the fb parameters. To understand why being able to co-optimise the fb controller and reference is usefull, how constraints are handled in the presence of uncertainty needs to be finally explained.

Starting with a repetition of the purpose of the constraints. The reason the constraints are present is because some sates or actions should never be visited or chosen. If the system is stochastic, it can be the case that the probability of a constraint violation is never fully 0. If this is the case the constraint can be rephrased to state that the probability of a constraint violation is less then a chosen probability. If tube MPC is used this can be enforced by tightening the constraints of the nominal trajectory by the width of the tube. Figure 3 illustrates what this might look like.

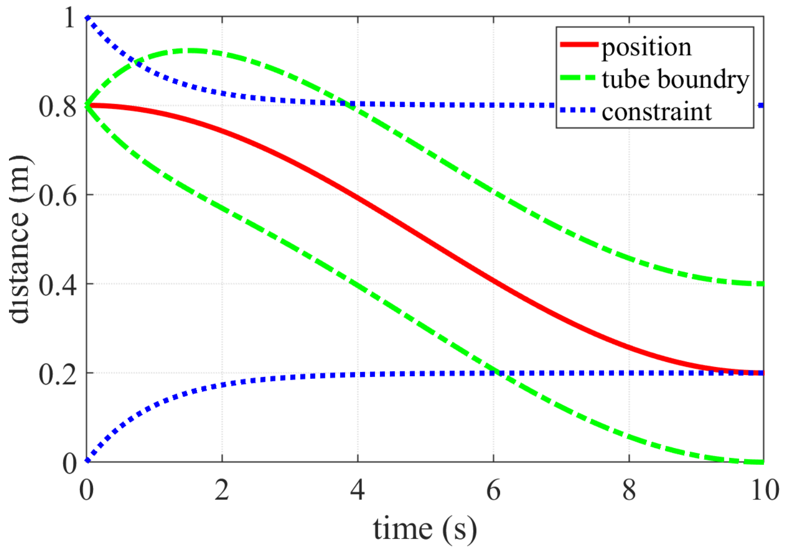


Figure 3: Ilustration of the state, tube and constraint evolution over a prediction horizon

If the control problem is split into a reference generation and disturbance rejection problem, with each having its own controller as in [18] and [14], further care has to be taken to guarantee controller constraints are satisfied. Additional care has to be taken because generally the MPC controller is not aware of the actions, or even the existence, of the second controller. Meaning that if the MPC controller would have access to the full constraint actions space, the actions of the second controller might lead to a constraint violation. To prevent this both papers split the constraint controller resources between the reference and feedback controllers.

This leaves the question, what happens if the fb and mpc objective are not the same? As the mpc controller controls the reference the fb controller uses to reject disturbances, it indirectly also controls the control signal the fb controller produces. This means that if the fb controller is included in the model the mpc controller uses, it will generate references that track its objective as close as possible, effectively bypassing the fb controllers objective. This behavior is demonstrated using a simulation and shown in figure 15a.

As far as the author is aware there are three solutions to this problem. The first solution is to not include the fb controller in the mpc model. This should work under the assumption that the fb controller is fast and "good" enough that the mpc controller will not significantly notice the disturbances. The problem is that this effectively is just adding additional unmodeled disturbances hoping they cancel out. Although this can work in practice there are some cases where this method runs into issues.

The second method is to also include any additional objectives used for fb in the cost function the mpc controller uses. This seems like a better solution but can run into practical problems as explained in section 3.2.3.

The final solution is to recognise that any disturbances requiring a different objective function are just due to model errors, and instead of attempting to deal with it using feedback, it can also be handled by updating the model the MPC controller uses online.

3 Methodology

After discussing general MPC theory in the previous chapter, this chapter will be used to discuss its application to the T-flex. This is done by proposing explaining and justifying a method to apply model-predictive control to the T-Flex. First, the feedforward (ff) parameter estimation will be explained. Because the control scheme that is proposed in this chapter relies on a ff that performs well, the results of the estimation will already be shown and discussed. After verifying the accuracy of the parameter estimation the MPC based controller will be explained, followed by a more detailed description of the individual components, and their functions. The next chapter will detail the exact implementation, and evaluate the performance using several simulations.

3.1 Feedforward Parameter Estimation

From earlier work it is known that it is possible to create a feedforward that takes care of most of the required control input to follow a reference [25]. This means that assuming a near perfect feedforward is not unreasonable for the T-flex. Allowing for control schemes that rely on an accurate feedforward, with any mismatches being considered additional disturbances that need to be corrected by the feedback controller. In this thesis a similar procedure as described in section III, D from [25] is followed. It uses plant inversion to get a least square estimate of a actuator dynamics. In this section the procedure will be briefly explained again and considerations specific to this thesis will be covered.

Parameter estimation on the T-flex is not very complicated due to the presence of the current sensor and highly accurate encoders. The current measurements combined with the known Torque constant of the motors make it possible to measure the current going trough the motors and derive the applied torque on the system using the equation 10, where I is the current and K_i is the motor toque constant which is 5.57 Nm/A [1].

$$\tau = I\,K_{\text{i}} \tag{10}$$

For the parameter estimation the position θ in rad, velocity $\dot{\theta}$ in $\frac{\text{rad}}{\text{s}}$ and angular acceleration $\ddot{\theta}$ in $\frac{\text{rad}}{\text{s}^2}$ are required. These can be obtained from the encoder and taking the first and second-order discrete derivative of it. Taking derivatives of measurements can be problematic as high frequency noise is amplified. The encoder used is the Heidenhain LIC 4007-411 encoder which offers a resolution in nano radians, so position sensor noise or error is not a real concern. The same filter and method of obtaining the derivative states $\dot{\theta}$ and $\ddot{\theta}$ are used as described in section IV, D of [25]. The filter is a 101 Hann-window based FIR filter.

$$w(n) = 0.5(1 - \cos(2\pi\frac{n-2}{101-1})) \quad \text{for } 1 \leq n \leq 101 \tag{11}$$

To be clear, all the measurements θ, $\dot{\theta}$, $\ddot{\theta}$ and τ are filtered. The feedforward parameter estimation uses a parametric model of the plant. The model terms themselves are nonlinear with respect to the measured signals, but system is assumed to be a linear combination of these nonlinear terms. The model equation is,

$$\tau = J\ddot{\theta} + d\dot{\theta} + k\theta + o + Cz + \sum_{i=1}^{9} a_i\cos(iw\theta) + b_i\sin(iw\theta) \tag{12}$$

where the system vector $\bar{p}(x)$ are the fixed or state dependent variables $[\ddot{\theta}, \dot{\theta}, \theta, 1, z, \cos(iw\theta), \sin(iw\theta)]$, while $\phi = [J, d, k, o, C, a_i, b_i]$ is the parameter vector that needs to be estimated. The z term is a dhal hysteresis state for which the zoh discretised version

$$z_k = z_{k-1} + \frac{\theta_k - \theta_{k-1}}{2 \cdot 1.5 \cdot 10^{-2}}(1 - \text{sign}(\theta_k - \theta_{k-1})z_{k-1}) \tag{13}$$

is used. The dhal state is estimated using the filtered states. While [25] uses a Kalman filter to estimate the parameters online, this thesis estimates the parameters offline. The reason for this is that the Kalman filter was designed for the full T-flex configuration, while this thesis only considers a single arm. The offline estimation is performed by letting the system follow some reference, during which measurements of the states and torque at each time step can be collected. The torque measurements and states can be written in the shape of $Ax = b$. Where $x = \phi$, $A = P(x,t)$ is a matrix where each row is the vector $\bar{p}(x,t)$ for a different measurement, and $b = \tau$ is the torque applied to the system. This system of equations can be solved to obtain a least square estimate of the parameter vector by rewriting the equation as $x = A^{-1}b$. The non-cogging terms can be seen (14), the cogging parameters can be found in the appendix D.

$$\begin{aligned} J &= 0.0818 \\ d &= 0.0161 \\ k &= 1.76 \\ o &= 1.07 \\ C &= 0.0732 \end{aligned} \tag{14}$$

The ff is implemented by generating a reference consisting of $x_{\text{ref}} = \ddot{\theta}_{\text{ref}}, \dot{\theta}_{\text{ref}}, \theta_{\text{ref}}$, the ff torque is then calculated using the following equation.

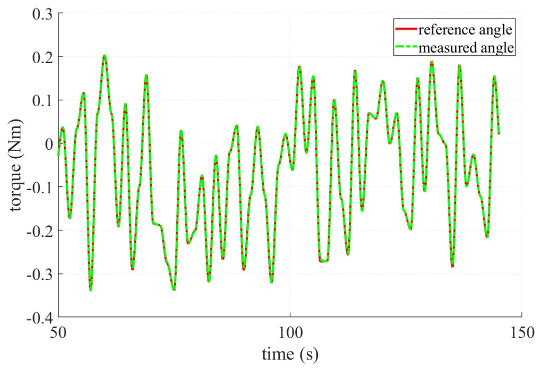
$$\tau_{\text{ff}} = \bar{\phi} \cdot \bar{p}(x_{\text{ref}}) \tag{15}$$

Because of a mistake, the dhal model in the Measurements subsection uses the filtered measured angles instead of the reference angles.

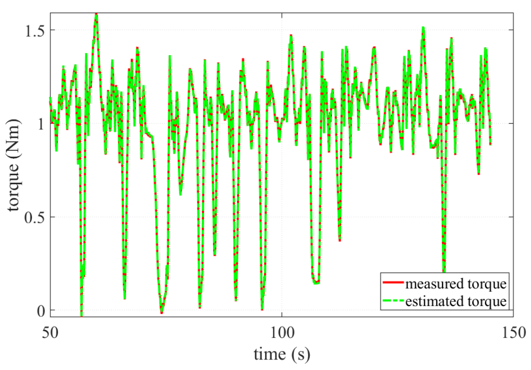
3.1.1 Measurements

To perform and validate the parameter estimation and ff implementation, two experiments were performed . Both experiments involve measurements from 50 to 150 seconds at 4000 hz, the start time is a result of required startup time. In both experiments the references are 3rd order profiles that move between randomly chosen setpoints that are between -20 and 12 degrees of the 0 position. The asymmetric limits are to avoid the angle limits. The difference between the experiments is that in the first experiment the setpoint changes each 1.5 seconds while in the second one it changes every 0.66 seconds.

The first experiment shown in figure 4 was performed to estimate the parameters. Figure 4a shows the reference and measured trajectory. Note that for these experiments all measurements came from data going trough the filter, while the reference did not, so there is a slight time shift between them. This is just an issue when comparing the reference with the trajectory, because the reference is not used in the estimation itself. The second figure shows the measured torque and the estimated torque using the ff. As can be seen in figure 4b the estimated torque is close to the measured torque, meaning the model is a good fit for the data.



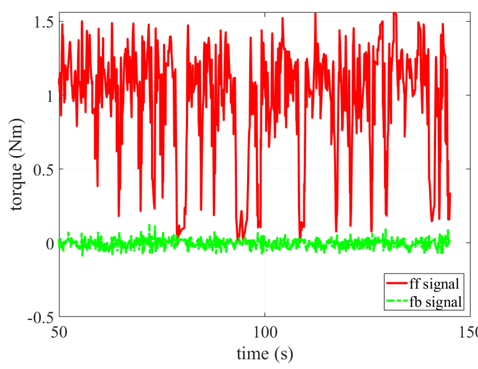
(a) The reference and measured trajectory



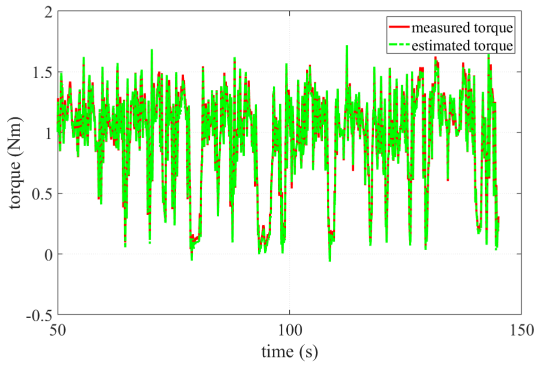
(b) The measured torque and reconstructed torque

Figure 4: Measurements from a cubic motion profile with new set points every 1.5 seconds

The second experiment was performed for several reasons. The first reason is to have a different motion profile to check if the good fit in the first experiment extends to different trajectories. This is important because the parameters are only estimated once. These results can be seen in figure 5b and show that the fit is still good, but there is a visible error. The visible error can have several causes. The first cause could be the different trajectory. The second potential cause is that the second experiment is performed at a different date. In figure 5a it can be seen that the ff indeed does most of the work. The fb signal is worse then similar experiments performed in [25]. This could be due to the difference in the setup or used reference. The result still seems good enough to use for the proposed control scheme.



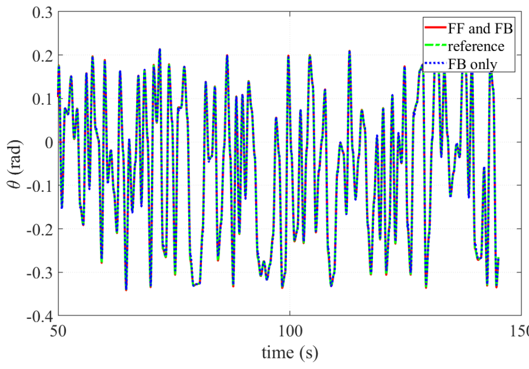
(a) comparison between the ff and fb signal of the experiment combining both



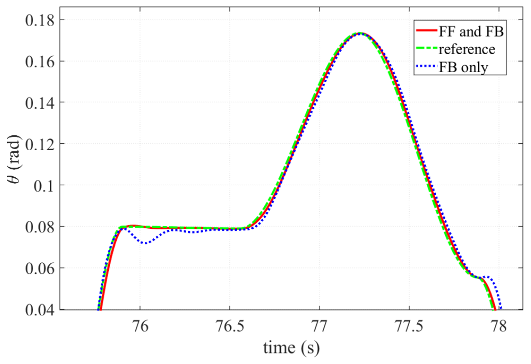
(b) The measured torque using only the fb controller and reconstructed torque

Figure 5: Measurements using a cubic motion profile with new set points every 0.66 seconds

The higher accelerations should also be more difficult to track, hopefully highlighting any issues with the combined ff and fb design. From the previous two figures no significant issues where found and the same is true when looking at the reference and measured angles found in figures 6. What does become clear is that for the more aggressive trajectory, the ff fb combination does perform visibly better when looking at figures 6b .



(a) The reference and the measured trajectories using the fb and combined ff, fb controllers



(b) A zoomed in version of figure 6a to show the difference in performance

Figure 6: Measurements using a cubic motion profile with new set points every 0.66 seconds

3.2 Controller Design

The choice is made to use MPC to control the nominal system. The system dynamics will be simplified in the system model the MPC will use. It will be used to generate a trajectory for the nominal plant, the control signal to follow this reference will be generated by a hybrid ff and feedback linearisation controller. A feedback controller will be used to minimize the error between the reference and actual plant state, thus rejecting disturbances. A block diagram for the proposed control scheme can be seen in figure 7. For safety the constraints have been tightened by the expected deviation from the nominal trajectory with the proposed control scheme. The reason that this control method is chosen is that section 3.1 shows that is possible to perform most of the required control on the Tflex using feedforward, leading to good tracking of the nominal trajectory. A feedback controller is also already available, and is shown to be able to track relatively aggressively trajectories, even without the presence of a feedforward controller. It is also able to compensate for disturbances and model errors when combined with ff, without producing excessive control signals. This meant that the proposed control scheme seemed likely to perform well in practice. In the remaining parts of this section, each part of the whole control scheme will be described in more detail. Exact values for the parameters will not be provided yet, instead this will be done in the experimental section 4, as they differ between experiments.

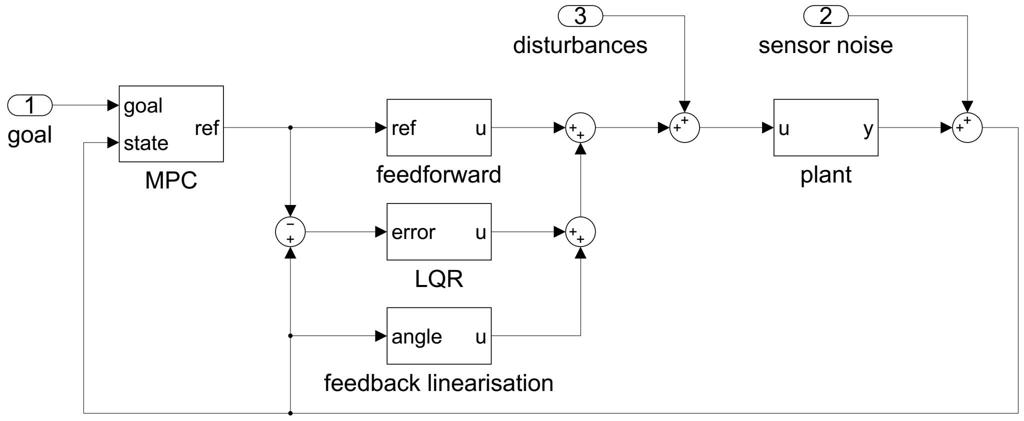


Figure 7: The MPC control setup

3.2.1 Feedforward and Computed Torque Control

The hybrid feedforward and feedback linearisation is explained first, because it impacts the design of both the MPC and fb controllers. Equation (15) shows the how the ff torque is calculated. The $\tau_{\text{ff}}(\ddot{\theta},\dot{\theta},\theta)$ can be split into its separate components $\tau_{\text{ff}_{\theta}}(\theta) = \bar{\phi}_{\theta}\bar{p}_{\theta}(\theta)$, $\tau_{\text{ff}_{\dot{\theta}}}(\dot{\theta}) = \bar{\phi}_{\dot{\theta}}\bar{p}_{\dot{\theta}}(\dot{\theta})$ and $\tau_{\text{ff}_{\ddot{\theta}}}(\ddot{\theta}) = \bar{\phi}_{\ddot{\theta}}\bar{p}_{\ddot{\theta}}(\ddot{\theta})$. The T-flex dynamics can be described using $\ddot{\theta}I = \tau((\dot{\theta},\theta))$ with $\tau = \tau_{\text{motor}} + \tau_{\dot{\theta}}(\dot{\theta}) + \tau_{\theta}(\theta)$. The idea of the ff is that if $\tau_{\text{motor}} = uI - \tau_{\dot{\theta}} - \tau_{\theta}$ that the angle and velocity contributions are canceled out, leaving only the motor term in the dynamics, which has already been multiplied with the inertia such that the controller controls the acceleration instead of the torque. Thus, assuming a perfect ff the resulting system as controlled by the controller is $\ddot{\theta} = u$.

Two options to calculate τ_{ff} are considered. The first option is to create a feasible reference of the desired trajectory, and use this reference to compensate the system dynamics. The second option is to use the measured state of the actual system instead of the trajectory state. The difference between these two options is that the first is a feedforward technique, while the second one is a feedback linearisation technique called computed torque control. Computed torque control being a feedback technique means it suffers from the same potential stability issues as other feedback methods.

Under the assumption of a truly ideal system, meaning perfect knowledge and no delay, both methods control the plant perfectly, it is only when non ideal behavior is considered that the difference between them becomes noticeable. Both methods use a model of the system to calculate the desired torque, so any errors in the model will have the same effect on both methods. The ff method uses the reference to calculate the required torque, meaning it will become less accurate if the actual system state deviates from the reference. While the computed torque control method uses the measured state, meaning it is sensitive to measurement errors, noise and delays. For MPC this difference does not matter as it will control the nominal plant, which both methods control perfectly as the non-ideal behavior is not included in the nominal plant. However, for the fb controller the difference does matter. To explain the difference consider the case where the non angle dependent torques J, d are assumed to be perfectly compensated using ff, leaving only the position dependent torques shown in figure 8.

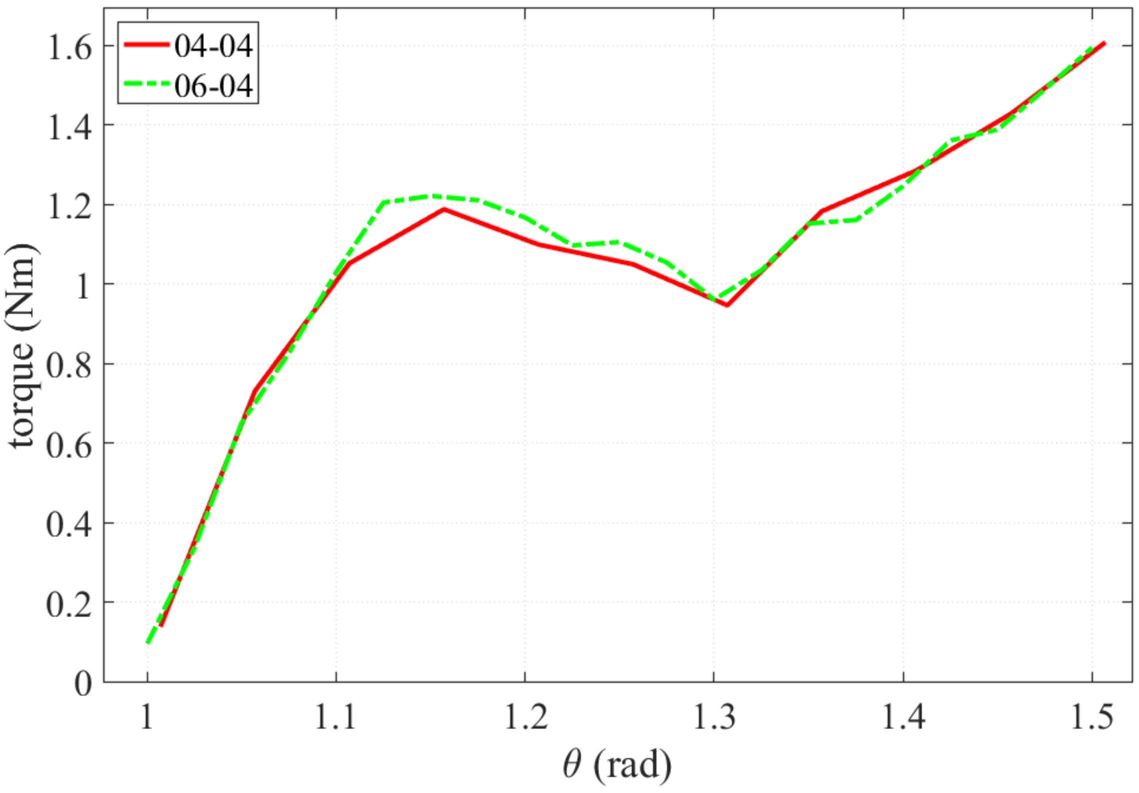


Figure 8: Angle vs torque measurements taken at the fourth and sixth of April

If ff is then also used for the angle, and the reference angle is 1.4 rad but the actual angle is lower. The ff will apply more torque then is required to keep the angle, steering the system towards the desired angle. The opposite is true between 1.15 and 1.3 rad. generally for the angle it can be stated that if ff is used, the system will appear to be a mass spring damper system, with a spring constant equal to the derivative of the angle dependent torque. Because this torque is not linear with respect to the angle, the "spring constant" changes with the angle state of the system. The same is not true if computed torque control is used. Because then the torque is calculated using the measured system state. Of course this measurement is not perfect, but considering the accurate sensors and 4000 Hz sample rate the Tflex uses, it seems likely computed torque control will result in better performance. For the J, d states the relation between the state and torque is linear, thus the derivative can be used as the apparent system dynamics when designing the fb controller. This is the reason computed torque control is only used for the angle dynamics, while the $\ddot{\theta}$ and $\dot{\theta}$ dynamics use ff. The added risk of computed torque control does not seem like it is worth it for these linear states with worse state estimation.

3.2.2 Feedback

A feedback controller is required to manage disturbances. A PID controller, that is shown to work for reference tracking, is already available. Still the choice was made to design a state feedback controller using LQR. There are two reasons for this choice. First, with regard to the inclusion of the integrated state in the PID controller, its potential drawbacks have already been explained in section 2.5. Secondly, in the authors opinion it seems more natural to design the fb using a method that results in a controller that is designed for the same objective as the MPC controller, this made LQR appear like an fitting and easy to use choice. However, there do not seem to be any problems if other methods such as H_2 or H_∞ were to be used.

The benefit of using computed torque control to cancel the angle dynamics is that the fb design becomes simpler, as only the linear velocity and acceleration dynamics of the system remain. There will still be some angle dependent dynamics, but these are the result of modeling errors so assumed to be unknowable.

The resulting ode of the system the fb controller controls is,

$$\begin{aligned} \dot{\theta}_{\text{error}} &= \dot{\theta}_{\text{error}} \\ \ddot{\theta}_{\text{error}} &= -\frac{d}{I}\dot{\theta}_{\text{error}} + \frac{u}{I} \end{aligned} \tag{16}$$

with θ_{error} and it derivatives being $\theta - \theta_{\text{ref}}$ and it derivatives.

All of the above results in the following system for which the LQR controller is designed using matlabs LQR() function [10]. Which takes the A and B matrices from a linear state space model, together with the state and action weights R and K, to return the state feedback gains K.

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ 0 & \frac{-d}{I} \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ \frac{1}{I} \end{bmatrix} \\ Q &= \begin{bmatrix} q_\theta & 0 \\ 0 & q_{\dot{\theta}} \end{bmatrix} \\ R &= q_u \\ K &= R^{-1}(B^T S + N^T) \end{aligned} \tag{17}$$

The variable S is the solution to the algebraic Riccati equation $A^T S + SA - (SB + N)R^{-1}(B^T S + N^T) + Q = 0$, $N = \mathbf{O}$ [10]. The q values are weighting factors, as these are experiment specific, they which will be chosen in the experimental section. The feedback is implemented using the equation,

$$\text{fb}(\theta_{\text{error}}, \dot{\theta}_{\text{error}}) = -K_\theta \theta_{\text{error}} - K_{\dot{\theta}} \dot{\theta}_{\text{error}} \tag{18}$$

with K_θ being the angle feedback gain and $K_{\dot{\theta}}$ the angular velocity feedback gain. The states are not filtered and the velocity is obtained by taking the discrete derivative of the angle measurements.

3.2.3 Nominal NMPC

The choice was made to let MPC control the nominal plant. Because ff and computed torque control is used the MPC can generate a reference for a double integrator system with the states θ and $\dot{\theta}$.

Ideally the way MPC generates references includes all reference that can be tracked by the system, without including ones that can not be tracked or excluding ones that can be tracked. This is why the reference acceleration was chosen to be the controlled variable, while the reference velocity and position are obtained by integrating the reference acceleration.

The justification for this is that this variable can be freely changed between timesteps, while still allowing most physically feasible solutions. With the limiting factor being the sample rate of the MPC controller limiting the temporal resolution of the reference. This choice of reference generation adds two additional states to the MPC controller, namely the reference velocity ($\dot{\theta}_{\text{ref}}$) and position ($\theta_{\text{ref}}$).

The fifth state is the integrated error between the goal and the actual position (e_θ). The reason for its inclusion is discussed in section 2.5. Although the author expects that using a disturbance observer is a better solution, there was not time to implement one leading to the possibly worse but quicker to implement solution. It should be noted that $\theta - \theta_{\text{goal}}$ is not a good observer for constant disturbances. because it measures the error between the current state and the goal, meaning it accumulates an error when moving between setpoints. Sadly MPC can not use $\theta - \theta_{\text{ref}}$ because this state is not controllable for the MPC controller.

The sixth and final state is that of the thermal state of the motor. Because the experiments described in section 3.1 did not increase the temperature of the motors by more then $1°C$, the thermal behavior of the system could not, and was not, estimated or verified using the experiments run on the real setup. Instead, like the motor current torque constant, the values provided in the data sheet [1] will be used. The relevant values are the thermal time constant $T_{\text{tc}} = 29$s and maximum steady state current $I_{\text{lim}} = 3.92$A. The idea behind these parameters is that, instead of fully modeling the thermal behavior of the motors, a first order system with the thermal time constant is used to track the thermal state of the motor. This state may not exceed the I^2_{lim}, as this state represents the maximum safe temperature of the motor. If T is used for the thermal state the inequality constraint becomes.

$$I^2_{\text{lim}} \geq T \tag{19}$$

If the controller is aware of this state it can chose to let I momentarily exceed the I_{lim}, as long as it will not result in a violation of the state limit. As the MPC consists of a set of odes, the ode describing the evolution of T is,

$$\dot{T} = \frac{(\tau/K_{\text{i}})^2}{T_{\text{tc}}} - \frac{T}{T_{\text{tc}}} \tag{20}$$

with τ being the total torque applied to the system and K_{i} being the current-torque constant of the motor. The total torque applied to the system is a combination of the torque required to follow the reference generated by the MPC and the fb controller. The torque required by the fb controller is not modeled, instead the torque required for the fb is determined and allocated offline, and the MPC torque allocation has been reduced by this amount (b_{fb}). The reason for this choice is due to the different sample times of the MPC and fb controllers. the MPC will only update each 0.05 seconds, while the fb controller has a time interval of only 0.00025 seconds. If a large disturbance is introduced between MPC updates, the τ_{fb} could become large enough to exceed the torque limit before the MPC controller received an update.

This just leaves the ff torque which is not yet included in the MPC model, because the system dynamics where simplified. To include this the following equation

$$\tau = \text{ff}(x_{\text{MPC}}) \tag{21}$$

is used. The subscript MPC is used for the variable x to indicate it is the state the MPC predicts the system will have. The function $\text{ff}(x_{\text{MPC}})$ is the hybrid feedforward and feedback linearisation function and has already been explained in subsection 3.2.1.

To summarise the complete set of odes the MPC model of the system are given by this

$$\begin{aligned}
\theta &= \dot{\theta} \\
\dot{\theta} &= u + \text{fb}(\theta_{\text{error}}, \dot{\theta}_{\text{error}}) \\
\theta_{\text{ref}} &= \dot{\theta}_{\text{ref}} \\
\dot{\theta}_{\text{ref}} &= u \\
e_\theta &= \theta - \theta_{\text{goal}} \\
\dot{T} &= \frac{(\tau/K_{\text{i}})^2}{T_{\text{tc}}} - \frac{T}{T_{\text{tc}}}
\end{aligned} \tag{22}$$

set of odes. The MPC subscript is not included as it should be clear that all states are predicted.

This just leaves the description of the constraints. First are the constraints setting the first state equal to the observed states.

$$x(1) = x_{\text{measured}} \tag{23}$$

To be clear the vector x used here does not include the decision variable u. Secondly are the continuity constraints because multiple shooting is used. A fixed interfall 4th order Runge–Kutta method is used, as it is generally a good default option to numerically discretize the continuous time system equation. This set of equations is then given by,

$$\begin{aligned}
&k_{p1} = P(x(k), u(k)) \\
&k_{p2} = P(x(k) + \Delta t\ k_{p1}/2,\ u(k)) \\
&k_{p3} = P(x(k) + \Delta t\ k_{p2}/2,\ u(,k)) \\
&k_{p4} = P(x(k) + \Delta t\ k_{p3},\ u(k)) \\
&x(k+1) = x(k) + \Delta t/6\ (k_{p1} + 2\ k_{p2} + 2\ k_{p3} + k_{p4}) \\
&k_{L1} = L(x(k), u(k)) \\
&k_{L2} = L(x(k) + \Delta t\ k_{L1}/2,\ u(k)) \\
&k_{L3} = L(x(k) + \Delta t\ k_{L2}/2,\ u(k)) \\
&k_{L4} = L(x(k) + \Delta t\ k_{L3}\ u(k)) \\
&\text{loss} = \text{loss} + \Delta t/6\ (k_{L1} + 2\ k_{L2} + 2\ k_{L3} + k_{L4})
\end{aligned} \tag{24}$$

where $P(x, u)$ is the substitution of the state and action in ode (22), $L(x, u)$ is the loss function and Δt is the time between time steps. The specific equality constraint is $x(k+1) = x(k) + \Delta t/6\ (k_{p1} + 2\ k_{p2} + 2\ k_{p3} + k_{p4})$.

The MPC controller will have a sample time of 0.05 seconds with an horizon of 40 time steps. The sample time is chosen because it seems a nice balance between update rate and the difficulty of completing the optimisation within the available time. The amount of time steps is chosen for convenience and could likely be reduced, considering the arm can reach the entire works space within 0.66 seconds.

This just leaves the safety and stability constraints. The first safety constraint is,

$$\theta_{\text{lower lim}} + \sigma_\theta \leq \theta_{\text{ref}} \leq \theta_{\text{upper lim}} - \sigma_\theta \tag{25}$$

which keeps the reference away from the angle limits by the expected uncertainty σ_θ. The second constraint

$$I^2_{\text{lim}} \geq T - b_{\text{fb}} \tag{26}$$

makes sure the current budget is not exceeded. The variable b_{fb} is the thermal budget reserved for the fb controller.

The final constraint are the stability constraints

$$\begin{aligned}
\theta(N) &= \theta_{\text{goal}} \\
\dot\theta(N) &= \dot\theta_{\text{goal}}
\end{aligned} \tag{27}$$

which ensures both the velocity and position at the terminal stat of the prediction horizon match the goal. This also ensures that if a solution is found the system will be and remain safe. Because the solver provides a warning if it did not find a feasible solution, a fallback policy can be used to bring the system to rest if no solution is found.

The final part is the cost function used by the MPC controller.

$$\begin{aligned}
&l = \sum_{i=1}^{5} l_i \\
&l_1 = q_\theta (\theta - \theta_{\text{goal}})^2 \\
&l_2 = q_{\dot\theta} (\dot\theta - \dot\theta_{\text{goal}})^2 \\
&l_3 = q_{\text{int}} (e_\theta)^2 \\
&l_4 = q_T\ T^2 \\
&l_5 = q_u\ u^2
\end{aligned} \tag{28}$$

To close this section, warm starting will not be used. Although preferable in practice, it should not help significantly at time steps where the setpoint changes. Because the assumption that the problem is similar to the last time step does not hold in those instances. As creating a MPC implementation that can work in real time is one of the goals, and the simulation is such that the time that is required for MPC is printed in a window after each time step. This makes it easier to roughly see what the worst case runtime is.

4 Simulation

In this section the design of the experiments and their goals will be explained. The experiments are performed using a simulation of two inverted pendulum systems. The first system will be a digital recreation of the Tflex, while the second system will be a system using different and simplified parameters. The second system is included to better demonstrate situations where MPC can be useful or demonstrates behavior that the author wants to highlight. This chapter contains two parts. the first part describes the systems and their parameter in detail and describes how the controllers described in chapter 3.2 are implemented. The second part describes the different simulation experiments that will be performed, the reason they are performed and show and discuss the results.

The objective is to move to different setpoints within the operating range as quickly as possible. How this can be done is limited by two factors. First are the constraints, but for the T-flex there is also a frequency above which higher order modes of the system are excited which should be avoided. Thus the objective is to create controllers which move between arbitrary, but steady state feasible, setpoints, while not violating any constraints or exciting the parasitic modes. Note that the setpoints are steady state feasible on the setup, if the control method introduces additional restrictions, this may make the setpoint infeasible.

4.1 The systems

In this section a more detailed explanation of the systems will be given, beginning with the Tflex simulation.

4.1.1 Tflex Model

As explained in the feedforward section 3.1, a model of the Tflex used to for the feedforward is already available. As could be seen in figure 4b, the predicted torques are close to those of the real system. Therefore this model can be used to replace the external Tflex system in the Simulink model used to control the Tflex. This should result in a model which has dynamics comparable to the real system. This model clearly does not include noise or disturbances yet. However, as already explained in section 3.1, the sensor noise is not significant. Instead most of the "noise" present in the unfiltered position measurement, is the result of noise in the motor driver. This means that the "noise" is not actually measurement noise, but a unmodeled disturbance instead. It is assumed this disturbance is uncoloured. The disturbance should be able to replicated by adding a white noise signal in the mili-ampere range to the motor current. A a previous experiment shown in figure 24 of[20] shows the angle to be disturbed to around 5μrad. Trough trial and error, a motor current disturbance with a "band" going to around 0.3mA produces a similar position disturbance as in the experiment.

In the end the simulation is not as good a match to the real system as hoped. Likely because of the choice to not include higher frequent behavior, the simulation is stable with a PID controller that was likely unstable in reality. The word likely is used because on the real setup, the motor drives would sometimes make a noise that would slowly increase over time. But this didn't happen every time, or produced any visible oscillations. Regardless, the PID parameters were changed to values where this would not happen. The original PID values where also tried in the simulation but did not result in an increasing oscillation. This leads to the conclusion that the simulation can likely not be used to verify the stability of control schemes.

Because the simulation can not be used to show the stability of controllers, the LQR controller is designed to have similar characteristics as the PID controller. This required q_u for the LQR to be changed to 10^{-3}, a 150 hz lowpass filter was also added to limit the high frequency gain. The remaining q values, q_θ and $q_{\dot{\theta}}$, are the same as in equation (31) shown below. This results in the LQR controller parameters

$$\begin{aligned} K_\theta &= 31.62 \\ K_{\dot{\theta}} &= 2.756 \end{aligned} \tag{29}$$

and PID parameters

$$\begin{aligned} P &= 0.0848 \\ I &= 5.6549 \\ D &= 1.4137 \end{aligned} \tag{30}$$

The frequency response of both the continuous LQR and PID controllers are shown in figure 9, discretised versions is used in the simulation.

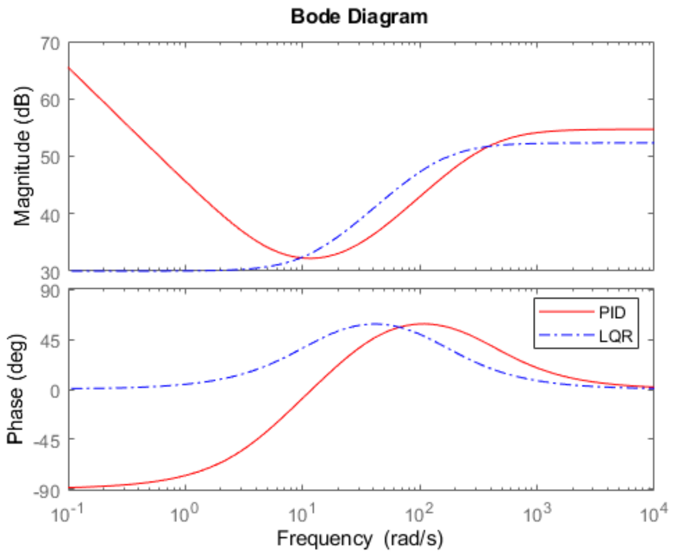


Figure 9: Bode plot of both the LQR and PID controllers

Based on figure 5a the fb will get 15% of the total torque allocated to it. This is based on the assumption the torque required scales linearly with the torque of the ff. The angle uncertainty σ_θ is estimated to be 0.02. This seems more then sufficient based on the error seen in the fb only reference tracking error observed in figure 6b.

The final part is the cost function used by the MPC controller. For simplicity the same choice as the fb design will be used, meaning a cutoff frequency of 20rad/s. Because a cost function is used and the target velocity is always set to 0, this cutoff frequency can be achieved by choosing the weight for the velocity to be $\frac{1}{20^2}$ the weight of the position error. The reference can not be controlled separately from the nominal plant, so will not have any cost associated with it directly. The cost for the integral error will be 0. Because this experiment is designed such that the setup is spending most of the time moving towards the set points. Both the temperature state and controller actions do not seem to have a natural choice for their costs. To prefer solutions that do not needlessly use control actions or increase the temperature, without improving the solution, both weights will be set to $1 \cdot 10^{-6}$. This makes the total cost function.

$$\begin{aligned} q_\theta &= 1 \\ q_{\dot\theta} &= \frac{1}{20^2} \\ q_{\text{int}} &= 0 \\ q_T &= 1 \cdot 10^{-6} \\ q_u &= 1 \cdot 10^{-6} \end{aligned} \tag{31}$$

4.1.2 Simplified Model

The idea behind the second model is to retain most of the interesting behavior of the Tflex, but make the model easier to work with. As well as to change the parameters such that the thermal limits actually become relevant. To do this an inverted pendulum with a motion range of $0 \leq \theta \leq 90°$ was chosen with the dynamics,

$$\begin{aligned} f_g(\theta) &= -\cos(\theta) m\, g \\ f_k(\theta) &= (\frac{45}{180}\pi - \theta)k \\ \text{ff}(\theta, u) &= -f_g(\theta) - 0.92 f_k(\theta) + 0.97 u\, I \end{aligned} \tag{32}$$

f_g is the torque created by gravity and f_k is the spring torque, which is a spring with a constant (k) centered at 45°. The system has no damping. The actual combined ff feedback linearisation has some estimation errors, which is why the 0.92 and 0.97 factors are present.

The parameters that differ from, or are not present in, the Tflex simulation are.

$$\begin{aligned} m &= 0.1 \\ I &= 1 \\ g &= 10 \\ k &= 3 \\ d &= 0 \\ K_i &= 1.25 \\ K_\theta &= 100 \\ K_{\dot{\theta}} &= 14.4914 \end{aligned} \tag{33}$$

The simulations are also initialised with the $T = (3.92 \cdot 0.8)^2$ to demonstrate a situation where the thermal limits are relevant. Finally the MPC and LQR models are the same as described in the implementation section 3. The chosen cost weights for both the LQR and MPC are.

$$\begin{aligned} q_\theta &= 1 \\ q_{\dot{\theta}} &= 1 \cdot 10^{-3} \\ q_{\text{int}} &= 0 \\ q_T &= 0 \\ q_u &= 1 \cdot 10^{-4} \end{aligned} \tag{34}$$

4.2 Reference Generation

Because neither the Tflex or simplified model can follow instantaneous setpoint changes in the angle. The setpoint changes need to be translated into a reference that the ff fb controlled systems can follow. This is not required for the MPC cases, as MPC should generate the optimal reference. A third order motion profile was chosen because it produces smooth enough references while being easy to implement and work with.

To limit the references to trajectories that are feasible, the constraints of the system need to be considered. Although it is theoretically possible to formulate a reference generation that is optimal, this reference generation would be the same as the solution found when solving the infinite horizon MPC problem offline. Because doing this is tedious, if analytically possible at all, references are generated using a method the author believes is both practical but does not use overly pessimistic simplifications.

The resource allocation between fb and ff will be the same as MPC, meaning the trajectory ff combination only has access to 85% of the constraint resources. The max acceleration of a third order motion profile is $t_{\text{trajectory}} = \sqrt{\frac{8\Delta\theta_{\text{setpoint}}}{\ddot{\theta}_{max}}}$, with $\ddot{\theta}_{max}$. The maximum allowed accelaration is limited by the torque limit $\ddot{\theta}_{max} = \ddot{\theta}_{lim}$, but what is $\ddot{\theta}_{lim}$?

First it is limited by the steady state torque, which is $(3.92 \cdot 0.85)^2$. This number is furter reduced as the torque required to keep a position is not yet included. This torque is $\tau_\theta = \max_\theta(\text{ff}(\theta, 0))$ also needs to be subtracted from the torque available for the trajectory. This means that when the setpoint is changed, a cubic motion profile that takes

$$t_{\text{trajectory}} = \sqrt{\frac{8\Delta\theta_{\text{setpoint}}}{0.85\ddot{\theta}_{\text{lim}} - \tau_\theta}} \tag{35}$$

seconds. This trajectory does not make maximal use of the available thermal budget. Because only the maximum acceleration uses the full budget, and the budget is decreased by the worst case τ_θ. However, finding methods that relax these pessimistic assumptions are not straight forward, partially due to the nonlinearity.

4.3 Experiments

The main objective is to track a changing setpoint. However, there is one other set of experiments that is designed to demonstrate or highlight certain aspects of the MPC control design. This section is used to explain what these two experiments are and why they where performed.

4.3.1 Setpoint Tracking

The main experiment of this thesis is the setpoint tracking experiment. The goal of this experiment is to compare a controller that is equivalent to the already used controller with the newly developed NMPC based control setup. To compare their performance two measurements will be used. First is the cumulative cost of the episode, secondly are the number of timesteps a constraint is violated. The used cost function will be the squared error between the position and setpoint $(x_{\text{goal}} - x)^2$. The reason this cost function is used and not the costfunction MPC uses is because the original goal is to minimize the position error between given setpoints and the actual position of the system. The costfunctions used for MPC and LQR are chosen such that the optimal solutions to the problem are a balance between the original objective and realworld limitations. Because these additional objectives are somewhat arbitrary, and can depend on the controller specifics. Including these additional costs in the scoring will unfairly favour the method whose objective function has been chosen as scoring metric. The requirement to not excite parasitic dynamics is assumed to be satisfied during the design phase. Both because this requirement has been explicitly taken into account and difficulties in objectively measuring how well this is achieved. Any constraint violations will be considered a fail, with the number simply being used to see how badly the controller failed. To be clear, violations of the system constraints are measured, not constraints that might be imposed on the controllers.

For the Tflex model four trajectories are generated. The setpoints are changed every 0.5, 1, 2 and 3 seconds. The first setpoint of each experiment is discarded, to allow the simulation to get past the initialisation behavior.

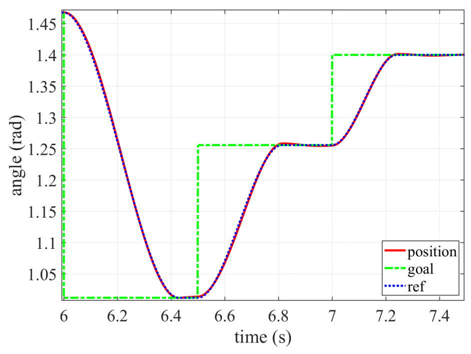
	fb ff	MPC
0.5s	$1.30 \cdot 10^3$	701
1s	535	289
2s	426	226
3s	208	113

Table 1: Table showing the cost of the fb ff and MPC control methods for the Tflex, the time between setpoints is shown in the left column.

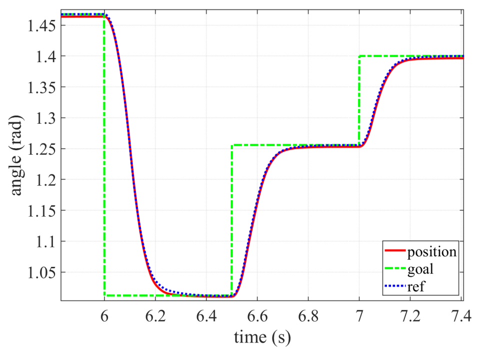
No constraints were violate in any of the experiments. The non-MPC controller performs worse, Likely because the used reference generation method creates references that take longer to reach the goal compared to the MPC references as can be seen in figure 10.

The trajectories both methods take is also different, with the non-MPC trajectory being more symmetrical. This is likely the result of the used objective function for MPC. It was designed with a cost function that should naturally limit the "aggressiveness" of the generated trajectories. This logic does not seem to hold when for moving towards this trajectory, producing high accelerations the moment the setpoint is changed. Increasing q_u should prevent this behavior.

Figure 10 also shows that, due to the lack of integrator, the MPC controller is not able to achieve 0 steady state error. This can best be seen by comparing both methods during the last setpoint in the figure. This will be better highlighted in the comparison between MPC implementation is the next section 4.3.2.



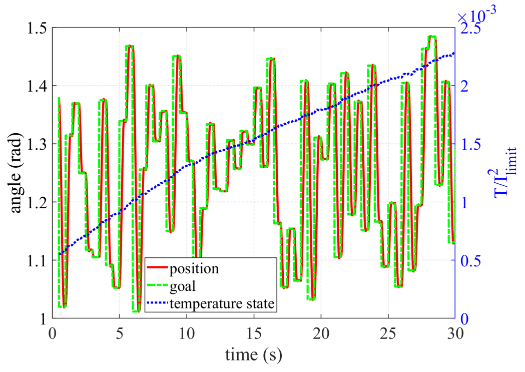
(a) State, reference trajectories and goal for the Tflex model using ff fb



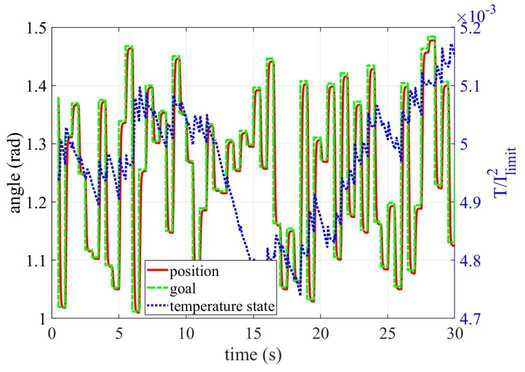
(b) State, reference trajectories and goal for the Tflex model using MPC on the left y axis

Figure 10: zoomed in comparison of the state, reference trajectories and goal of ff fb and MPC respectively. The setpoint is changed every 0.5 seconds.

The score for both methods could actually be improved by allowing for higher accelerations for the reference or the $q_{\dot{\theta}}$ weight is reduced for the MPC. Making the biggest factor in the difference in performance the chosen restrictions. It is also interesting to compare the thermal states in the 0.5s experiments.



(a) Goal and angle for the Tflex model using ff fb on the left y axis, while the right axis shows the temperature state as a fraction of its limit.



(b) Goal and angle for the Tflex model using MPC on the left y axis, while the right axis shows the temperature state as a fraction of its limit.

Figure 11: comparison of the trajectories of ff fb and MPC respectively. The setpoint is changed every 0.5 seconds.

Looking at the thermal states, it appears the MPC controller generates more heat. But neither controller seems likely to run into thermal constraints. Comparing the ff controller with the MPC controller is somewhat complicated because the MPC controller starts at $4.95 \cdot 10^{-3}$ and the non-MPC controller starts at $0.5 \cdot 10^{-3}$. This difference is caused by the startup of the simulation.

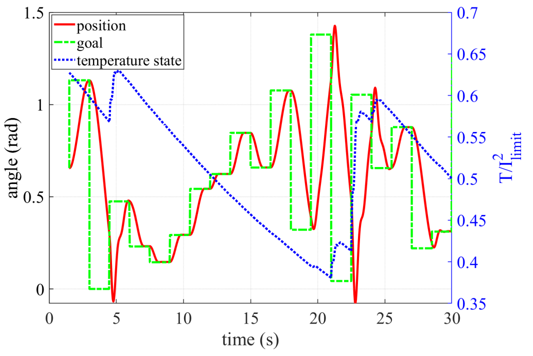
For the simplified model the setpoints are changed every 1.5, 2, 2.5 and 3 seconds. The first setpoint was discarded again.

The first two goal trajectories are infeasible for the non-MPC controller, because the time between setpoint changes is less then the time the reference requires to move between setpoints, while staying in the thermally constraint reference acceleration limit. The second setpoint is infeasible for the MPC controller in all goal trajectories. This is because the second setpoint lies between the tightened reference safety constraint and the angle constraint of the modeled system. This is discussed in more detail after figure 13.

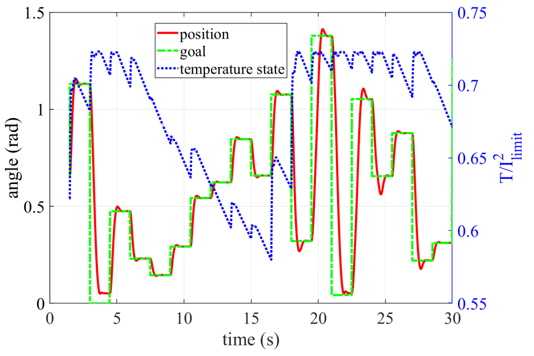
	fb ff	MPC
1.5s	$2.18 \cdot 10^4$	$6.78 \cdot 10^3$
2s	$1.75 \cdot 10^4$	$4.71 \cdot 10^3$
2.5s	$0.592 \cdot 10^4$	$1.40 \cdot 10^3$
3s	$0.552 \cdot 10^4$	$1.26 \cdot 10^3$

Table 2: The cost of the fb ff and MPC control methods for the simplified model, the time between setpoints is shown in the left column.

The non-MPC controller failed the 1.5s experiment. Table 2 shows that the MPC controller is significantly better. It is difficult to determine which part of the performance difference is explained by the MPC controller generating trajectories that have a lower cost with the used cost function, as was seen with the T-flex simulation, and which part is because it performs better in thermally constraint situations. The ability of the MPC controller to adapt to the current thermal state of the system can clearly be seen when comparing the time it takes to reach the setpoints between 17.5 and 25 seconds in figure 12. With it taking noticeably longer once the limit of $(3.92 \cdot 0.85)^2/3.92^2 = 0.7225$ has been reached.

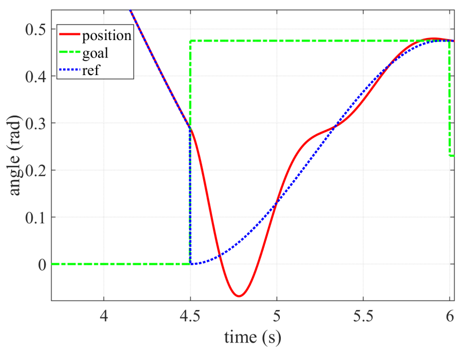


(a) Reference and angle for the simplified model using ff fb on the left y axis, while the right axis shows the temperature state as a fraction of its limit.

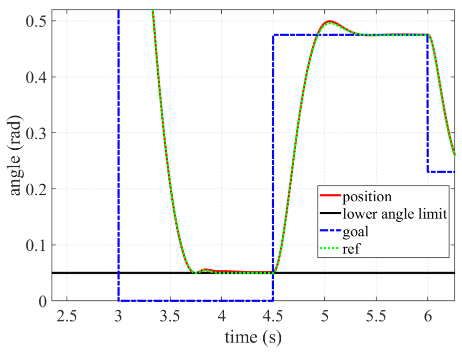


(b) Reference and angle for the simplified model using MPC on the left y axis, while the right axis shows the temperature state as a fraction of its limit.

Figure 12: Comparison of the trajectories of ff fb and MPC respectively. The setpoint is changed every 1.5 second.



(a) State, reference trajectories and goal for the simplified model using ff fb



(b) State, reference trajectories and goal for the simplified model using MPC on the left y axis

Figure 13: zoomed in comparison of the state, reference trajectories and goal of ff fb and MPC respectively. The setpoint is changed every 1.5 seconds.

In figure 13a it can be seen that the reference fails to reach the setpoint before a new setpoint is provided. The reference is generated in such a way that this results in a reference jump, if this jump is close to the state constraints, the overshoot can lead to a constraint violation, as happened in this case. For the MPC controller a different problem can be observed in 13b. The controller fails the reach the feasible objective because the reference is restricted to a region σ_θ away from the constraint boundary.

4.3.2 Disturbances

A key aspect of controllers is how well they can deal with disturbances. Showing how certain controller setups deal with disturbances also shows the issues that some setups have. The same criteria as the setpoint tracking experiment 4.3.1 will be used, but the reference will be constant and set to 1. Additionally a more subjective comparison between the results will also be made to clearly justify some of the claims made in earlier sections. The disturbance itself is an outside torque that is applied to the system $\tau_{\text{disturbance}} = 30\text{Nm}\ \forall\ (5\text{s} \leq t \leq 5.25\text{s})$ and $\tau_{\text{disturbance}} = 27\text{Nm}\ \forall(12\text{s} \leq t \leq 20\text{s})$. Because this experiment is used to compare the controllers it has only been performed on the non-Tflex simulation. The thermal limits are ignored and the new cost function

$$\begin{aligned} l &= \sum_{i=1}^{4} l_i \\ l_1 &= (\theta - \theta_{goal})^2 \\ l_2 &= 1e-3\ \dot{\theta}^2 \\ l_3 &= 10 e_\theta^2 \\ l_4 &= 1e-3\ u^2 \end{aligned} \tag{36}$$

is used for both the MPC and LQR controller. The change in cost function is to more clearly show the influence of the different ways of implementing the integrator. Six different combinations have been tested. The options are: either MPC with integrator (MPCI) or without integrator (MPC), LQR with integrator (LQRI) or without (LQR) and finally, the LQRI cases where the MPC(I) is aware of the integrator of the LQRI (LQRII). This results in the following list of cases:

- LQRII MPCI
- LQRII MPC
- LQRI MPCI
- LQRI MPC

- LQR MPCI
- LQR MPC

For MPC and LQR, the l_4 from (36) has been set to 0. For ease of coding the LQR gain was actually set to $1 \cdot 10^{-8}$ because Matlabs LQR function requires a positive definite matrix. After the gains where obtained the integral gain was manually set to zero. This only influenced the gain values after the third digit for the other gains, so should not significantly influence the results.

To make the model predictive controller aware of the integrator, a new state

$$\dot{e}_{\text{LQR}} = \theta - \theta_{\text{ref}} \tag{37}$$

is added, with e_{LQR} being the modeled integrator state of the LQRI. The second state ode is also changed to reflect the new LQRI equation,

$$\dot{\theta} = u + K_{\text{int}}(e_{\text{LQR}}) + K_{\text{pos}}(\theta - \theta_{\text{ref}}) + K_{\text{vel}}(\dot{\theta} - \dot{\theta}_{\text{ref}}) \tag{38}$$

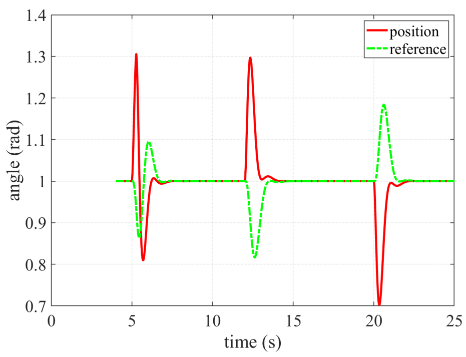
where K_{int} is the LQRI gain for the integrated state.

The total costs for the experiments are.

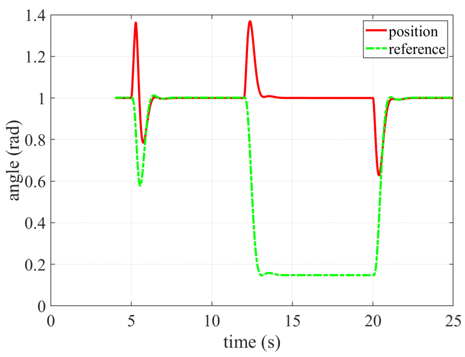
	LQR	LRQI	LQRII
MPC	$7.20 \cdot 10^3$	494	$3.23 \cdot 10^3$
MCPI	590	394	362

Table 3: Table showing the cost of different MPC and LQR controller combinations

As can be seen from the costs, the adding the integrator to the MPC leads to significantly worse performance then adding it to LQR. Although expected if the system spends a lot of time moving towards the setpoint, the experiment was designed such that this was not the case. It was expected that the performance would be roughly equal between the MPCI LQR and MPC LQRII cases, but that is clearly not the case. An explanation might be found when looking at figure 14.



(a) reference and angle for the disturbed system using LQRII and MPCI

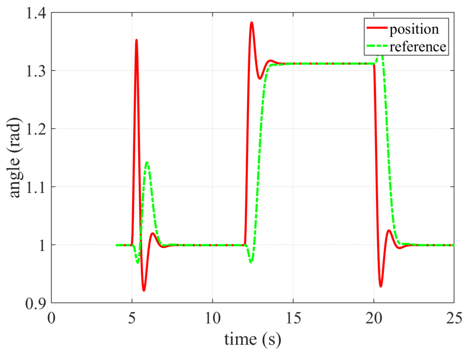


(b) reference and angle for the disturbed system using LQR and MPCI

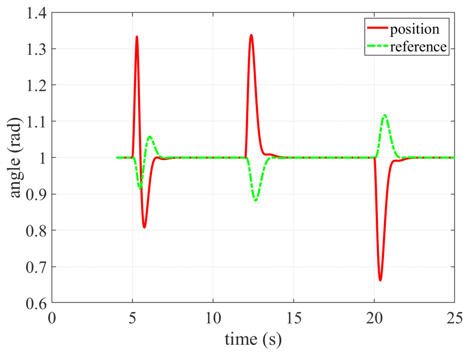
Figure 14: Comparison between the trajectories of MPCI in combination with LQRII and LQR respectively

What is most notable is the difference between the peak heights at the time the disturbance starts or disappears. With the LQR they are higher, likely because the MPC controller alone results in a slower response.

What was predicted in section 2.5 was the bad performance of the LQRII MPC combination.



(a) Reference and angle for the disturbed system using LQRII and MPC



(b) Reference and angle for the disturbed system using LQRI and MPC

Figure 15: Comparison of the trajectories of MPC in combination with LQRII and LQRI respectively

It clearly shows that, despite the presence of the integrator in the fb controller, there is still a steady state error. This is not the case if the MPC model does not include the integrator.

4.4 Computational Performance

To close this section a note about performance. A maximum of 300 ms was observed for the MPC calculation of a single time step for the T-flex, while the simple model took 100 ms. Observed maximum because it relies on reading the status window while the simulation is running.

Because Simulink uses a different compiler optimisations for simulation and target compilation, the code is run on a PC instead of an embedded controller, and there is a significant overhead from the Casadi runtime. It is not possible to make any concrete claims about if the MPC controller meets the realtime requirement of 20 Hz. Additionally, no effort made to reduce the required computation time by, for example, reducing the time horizon, or lowering the desired accuracy.

These measurements are generated using an Intel i7-4700MQ, which is a quad core laptop cpu from 2014. Making it likely that if more modern hardware is used without the Casadi runtime, which imposes a 5-10x slowdown [2], the 20 Hz realtime requirements can be met.

5 Conclusion

In this thesis a MPC controller was designed for the T-flex. To support and explain the chosen design, relevant sources are found and the details most relevant for this thesis are highlighted.

The chosen design attempts to negate the low sample rate of NMPC implementations by using it as a reference generator for the nominal system. A combination of a feedforward and computed torque controller use this reference to generate control signals that would make the nominal system follow the reference trajectory. The real system is then stabilised to this nominal reference trajectory using a LQR feedback controller to reject disturbances and small model inaccuracies. To ensure that the constraints are satisfied the constraints of the nominal system are tightened by the control resources and state uncertainty to ensure safety.

To demonstrate the performance of the new NMPC controller it is compared to a controller using a reference generation, ff and PID. This comparison is done using a simulation using a parametric model of a single T-flex arm, with parameters obtained trough system identification. This simulation showed that the performance difference in unconstrained situations is small, which was expected considering both controllers where designed using the same specifications. The simulation also supported the observation that it is unlikely the thermal limits will become relevant in practice.

A simulation of a system similar to the T-flex arm, but with simplified dynamics and different parameters was created. This system was used to simulate situations in which the thermal constraints become relevant. Using this system the advantages of the proposed NMPC controller are demonstrated. Showing that the proposed controller is better able to use all available resources, improving its performance compared to the more traditional reference generation.

A final set of experiments is performed to explore different choices that can be made when combining the fb controller and MPC. These experiments demonstrate disturbance rejection performance is better if the fb controller is included in the plant model of the MPC controller. It is also demonstrated that if the fb controller is included in the MPC model, the objective of both controllers should be the same, otherwise the MPC controller will chose references that make the complete system follow the objective the MPC has been given, negating the effects the fb controller was designed to have on the system.

5.1 Future research

For the T-flex specifically, there are several questions that are still open. First is the question of how the MPC controller will perform on the real setup. Related to the first question the option to look closer at the mechanical constraints and higher frequent behavior of the setup, which were mostly ignored throughout this thesis. This could be investigated further to see if performance can be improved. Last is expanding the control of the end effector and multiple arms. This can be especially computationally difficult because the end effector position depends on the state of all six arms. The model of each arm separately would also become more complex.

A direction more relevant for the wider MPC field could be to look at updating the MPC model online. Some publications working on this already exist with a web page of berkeley containing a collection of their contribution [3].

References

[1] Mar. 2023. URL: https://www.tecnotion.com/products/qtr-torque-133-series/.

[2] Aug. 2023. URL: https://web.casadi.org/blog/mpc-simulink2/.

[3] Aug. 2023. URL: https://sites.google.com/berkeley.edu/lmpc/home.

[4] Aug. 2023. URL: https://docs.acados.org/installation/index.html.

[5] Joel A E Andersson et al. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[6] Richard Bellman. *Dynamic Programming*. 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[7] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. Vol. 1. Athena scientific, 2012.

[8] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. DOI: 10.1017/9781139061759.

[9] Lars Grüne et al. *Nonlinear model predictive control*. Springer, 2017.

[10] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: https://nl.mathworks.com/help/releases/R2022b/control/ref/lti.lqr.html.

[11] Jason Kong et al. "Kinematic and dynamic vehicle models for autonomous driving control design". In: *2015 IEEE intelligent vehicles symposium (IV)*. IEEE. 2015, pp. 1094–1099.

[12] Daniel Limon, Teodoro Alamo, and Eduardo F Camacho. "Enlarging the domain of attraction of MPC controllers". In: *Automatica* 41.4 (2005), pp. 629–635.

[13] Björn Lindqvist et al. "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles". In: *IEEE robotics and automation letters* 5.4 (2020), pp. 6001–6008.

[14] Brett T Lopez, Jean-Jacques E Slotine, and Jonathan P How. "Dynamic tube MPC for nonlinear systems". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 1655–1662.

[15] Dailiang Ma et al. "Active disturbance rejection and predictive control strategy for a quadrotor helicopter". In: *IET Control Theory & Applications* 10.17 (2016), pp. 2213–2222.

[16] David Q Mayne. "Model predictive control: Recent developments and future promise". In: *Automatica* 50.12 (2014), pp. 2967–2986.

[17] David Q Mayne et al. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6 (2000), pp. 789–814.

[18] DQ Mayne et al. "Tube-based robust nonlinear model predictive control". In: *Int. J. Robust. Nonlinear Control* 21 (2011), pp. 1341–1353.

[19] Mark Naves et al. "T-Flex: A fully flexure-based large range of motion precision hexapod". In: *Precision engineering* 72 (2021), pp. 912–928.

[20] Mark Naves et al. "T-Flex: A fully flexure-based large range of motion precision hexapod". In: *Precision engineering* 72 (2021), pp. 912–928.

[21] S Joe Qin and Thomas A Badgwell. "A survey of industrial model predictive control technology". In: *Control engineering practice* 11.7 (2003), pp. 733–764.

[22] Guilherme V Raffo, Manuel G Ortega, and Francisco R Rubio. "An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter". In: *Automatica* 46.1 (2010), pp. 29–39.

[23] Sasa V Rakovic and William S Levine. *Handbook of model predictive control*. Springer, 2018.

[24] Anil V Rao. "A survey of numerical methods for optimal control". In: *Advances in the Astronautical Sciences* 135.1 (2009), pp. 497–528.

[25] Bram Seinhorst and Wouter Hakvoort. "Efficient formulation of hexapod kinematics enabling real time adaptive feedforward control". In: *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2021, pp. 186–191.

[26] Yang Wang and Stephen Boyd. “Fast model predictive control using online optimization”. In: *IEEE Transactions on control systems technology* 18.2 (2009), pp. 267–278.

[27] Stephen Wright, Jorge Nocedal, et al. “Numerical optimization”. In: *Springer Science* 35.67-68 (1999), p. 7.

Appendices

B Safety Constraints

For the T-flex there are two safety constraints. First the kinetic safety constraint which should limit the system to the angles it can reach without running into mechanical limits. Second is the thermal constraint which should prevent the motors from being damaged due to excessive heat. These constraints where not used during the experiments, because the used terminal equality constraint already guarantees safety. This section still shows how a horizon independent safety constraint can be obtained, and show a potential issue if this method is chosen.

B.1 Kinetic

The first constraint is given by the kinetic energy and the distance to the limits of the operating range. This limit exists because the acceleration is limited by the maximum torque the motors may or can apply, whichever is lower. This can be described with the equation,

$$E_k \leq \int_{\theta}^{\theta_{upper}} T_{lim} d\theta - \Delta E_{pot} \tag{39}$$

in which E_k is the kinetic energy and ΔE_{pot} is the difference in potential energy between the current angle θ and the maximum value the angle may be (θ_{lim}). The following equation defines two of the variables used in equation 39,

$$\begin{gathered} E_k = 0.5 I \dot{\theta}^2 \\ \Delta E_{pot} = (sin(\theta_{upper}) - sin(\theta)) r\, m\, g \end{gathered} \tag{40}$$

with $\dot{\theta}$ being the Angular velocity in rad/s, r distance between the center of mass and rotation point in meters, m the mass in kg and g the gravitational constant which is taken as $9.81 m/s^2$. Describing T_{lim} is slightly more difficult. The torque itself is only limited by the motor to 21.9 N continuously, or a maximum of 35.3 N for peak loads under certain conditions [1].

The work that can be done is $F\, \Delta x$, so fully writing out equation (39) returns.

$$0.5 I \dot{\theta}^2 \leq T_{lim}(\theta_{upper} - \theta) - (sin(\theta_{upper}) - sin(\theta)) r\, m\, g \tag{41}$$

B.2 Thermal

The second safety constraint that is required, is to make sure there is enough thermal budget to bring the system to rest. The description is for a system where the actual thermal state is modeled, even with the equations being slightly more complex, the overall derivation and issue would remain the same if equation (19) was used. This equation is slightly more involved and will show why constructing safety constraints can become problematic. Starting with,

$$\dot{T} = \left(\left(\frac{\tau}{K_i} \right)^2 R - \Delta T K_t \right) / C_p; \tag{42}$$

which describes the change in temperature of the motor. In eq. (42) K_i is the motor torque constant ($\frac{Nm}{A}$), C_p the heat capacity ($\frac{W}{K}$) of the motor windings and K_t the heat transfer coefficient between the windings and the room. The $\frac{\tau}{K_i}$ comes from substituting $I = \frac{\tau}{K_i}$. Before continuing the derivations the following definitions are also required.

$$\begin{gathered} T_{budget} = (T_{lim} - T) \\ \Delta T = T_{room} - T_{motor} \end{gathered} \tag{43}$$

To simplify the equations for the remaining derivation of the safety constraint, the heat loss to the room will be ignored, which is the $\Delta T K_t$ term of equation 42. This is justified because the situations for which this constraint becomes active happen over short enough time frames that the heat loss likely won't be significant. The difference will also restrict states that are actually safe, instead of allowing unsafe states.

An expression for τ in equation 42 is also needed. Because the heating is related to the square of the torque, the optimal way to slow down is to apply the minimum constant torque possible. This means that the most deceleration is possible by using the entire remaining distance untill the angle limit. However equation 42 requires τ so a way needs to be found to find both the time and the torque required to stop when following this policy. This can be done by using the fact that the traveled distance, given a certain acceleration, is given by $x = 0.5a\,t^2$. Combining this with the knowledge that it takes $\frac{v}{a}$ seconds to come to a stop, the distance required to come to a stop can be calculated by substituting $\frac{v}{a}$ for t. Then, assuming the optimal policy uses the complete distance and has a constant deceleration, Δx can be substituted for x. This gives an expression that gives the optimal a given a distance and velocity as seen in equation 44.

$$\Delta x = 0.5a\,t^2 \xrightarrow{\text{subs: } t=\frac{v}{a}} \Delta x = 0.5\frac{v^2}{a} \tag{44}$$

Before finding the thermal safety equation two more equations are needed. First the relation between torque and acceleration which is $\tau = I\,a$. Secondly equation 42 needs to be integrated over time such that it describes the temperature change of the motor using the policy. Luckily none of the variables are variable with time, meaning the integration results in equation 45 after substituting $t = \frac{v}{a}$ and $a = 0.5\frac{v^2}{\Delta x}$.

$$T_{budget} \geq \left(\frac{0.5\,m\,v^2/\Delta x}{K_i}\right)^2 \frac{R}{C_p}\frac{v}{0.5v^2/\Delta x} \tag{45}$$

this equation can be simplified to the version shown in equation 46.

$$T_{budget} \geq \frac{0.5v^3}{\Delta x}\left(\frac{I}{K_t}\right)^2 \frac{R_e}{h_c} \tag{46}$$

Equations 41 and 46 are both used to create inequality constraints that keep the terminal state of the prediction horizon within a known set of safe states. The terminal constraint will also limit some states that are actually safe. But the size of the incorrectly disallowed states is small, and the constraint only applies to the last state in the horizon. Meaning that it will probably not have a big impact on the quality of the solutions that will be found. A bigger problem is demonstrated by the choice to neglect the thermal losses and the assumption of the optimal policy. The reason that these choices where made is because otherwise the equation is time dependent, Which could make the evaluation of this boundary condition an optimization problem itself. This can potentially significantly increase the computation required to evaluate this condition and thus also of the entire mpc problem. This shows that it can be difficult to find boundary conditions that both guarantee safety, do not restrict the states too much while still making the mpc problem solvable within the time allowed.

C T-flex QP Implementation

Originaly the goal was to demonstrate the chosen MPC implementation on the physical setup. Eventualy the choise was made to abandon the physical demonstration due to the difficulties of finding an software implementation that would work on the used hardware. The first part of this appendix chapter contains a description of the difficulties regarding the software that where encountered in this failed attempts. The second part details a Quadratic Programming (QP) implementation for the T-flex.

C.1 MPC Harware Implementation

Finally a word about actually implementing the MPC algorithm and the mathematical optimizer that takes an optimisation problem and finds a solution. This has turned out to be more difficult then expected. Due to limited time and experience, only software that would work with Matlab was considered. As the entire process of creating and building the code has already been done using Simulink. Two software packages that seemed to fit this requirement and looked most promising are Casadi and ACADOS. It quickly turned out Casadi is not an option to run on the actual hardware as it requires the Casadi runtime environment for non-linear problems [2]. It looks like the SQP solver used for linear problems is supported, and the SQP solver possibly even supports nonlinear problems. Sadly it looks like the SQP solver will not work either as one of the functions used in the S-function block is not supported for target compilation. This ended attempts to use Casadi to control the physical setup.

Acados has two other problems. The first problem is that the author was unable to get the solver to find a solution if the heat was included in the problem. Admittedly this could simply be due to a lack of experience or not spending enough time on finding a workaround. However, ACADOS was discarded as an option after attempts to install ACADOS using either the MinGW install instructions, or the MSVC instructions [4] where unsuccessful.

A Phd candidate in the group has been successful in implementing MPC using matlab's buildin functions and compile it for the hardware. This sadly also turned out to not be viable in the remaining time, as matlab2018a doesn't support target compilation of the used function.

Although there is not much in actual results that resulted from this work, the reading and trial and error work was still a significant time investment.

C.2 QP Formulation

Due to the issues mentioned in the previous section an QP implementation was attempted as a proof of concept. The purpose was to demonstrate that a MPC implementation can work on a real system, not only in simulation. This subsection will go into detail how a linear MPC problem can be written as a QP problem, which can then be solved by a QP solver. This implementation was chosen as it was believed a QP solver was easily availible, this belief turned out to be wrong.

First the structure of a QP problem. A common structure, and the one the Matlab function quadprog expects is,

$$
\begin{aligned}
&\min_x \left(\frac{1}{2}x^T H x + f^T x\right) \\
\text{s.t.}\quad &Ax \le b \\
&A_{\text{eq}}x = b_{\text{eq}} \\
&x_{\text{lb}} \le x \le x_{\text{ub}}
\end{aligned}
\tag{47}
$$

with x being the state vector, Q the quadratic cost matrix, R the linear cost term, A and A_{eq} the inequality and equality matrices respectively, with b, b_{eq} being their respective inequality and equality constraints and x_{lb} x_{ub} the lower and upper state constraints. To translate MPC into this formulation first a method must be chosen to integrate the dynamics in the cost term as mentioned in section 2.2.1, which is the multiple shooting method. The multiple shooting method produces the following problem shape. For the T-flex $f = 0$, meaning the value of a single time step is given by,

$$
V_k = \frac{1}{2}x_k^T H_k x_k \tag{48}
$$

with the subscript k denoting the variables refer to the kth timestep. The variables are given by

$$
\begin{aligned}
x_k &= \begin{bmatrix} \bar{u}_{k-1} \\ \bar{x_k} \end{bmatrix} \\
H_k &= \begin{bmatrix} R & . \\ . & Q \end{bmatrix}
\end{aligned}
\tag{49}
$$

where $\bar{u_{k-1}}$ and $\bar{x_k}$ are vectors of $\mathbf{R}^{n_u}$ and $\mathbf{R}^{n_x}$ respectively. These vectors contain the actions and states of their respective time steps. While R and Q are the action and state cost matrices. The full problem is then described by the vector x being a vector of the vector x_k, while H will be a block diagonal matrix, containing the matrices H_k along the diagonal. For the T-flex all H_k are the same so their exact index will be ignored from now on, the subscript k will still be used to differentiate between H and H_k.

$$
\begin{aligned}
x &= \begin{bmatrix} x_2 \\ x_3 \\ ... \\ x_N \end{bmatrix} \\
H &= \begin{bmatrix} H_k & 0 & ... & . \\ 0 & H_k & ... & . \\ ... & ... & ... & . \\ . & . & . & H_N \end{bmatrix}
\end{aligned}
\tag{50}
$$

To quickly recap, the equations above describe how each time step contributes to the cost for the whole horizon, but there is nothing connecting the time steps together yet. How each time step depends on the preceding one is handled trough equality constraints with multiple shooting. The dynamics are encoded in A_{eq} and b_{eq}, the vector x in the constraints is the same as the one in the optimisation function.

Because QP solves a linear problem, a state space representation of the dynamics can be modified and used as the equality constraints. The usual discrete state space formulation is $x_{k+1} = Ax_k + Bu_k$. The first change is to account for the structure of x in the QP formulation. Because u_k and x_k are now part of the same vector, the same can be done to the A and B matrices, this creates the structure

$$
x_{k+1} = \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}
\tag{51}
$$

with A and B containing the state and input dependent dynamics respectively. How these can be found for linear problems from a continuous time description will be explained later, for now just assume these are known. The form of equation (51) is not yet in the shape of the equality constraint in equation (47). For that two timesteps need to be considered. Take the slice $[x_k \, x_{k+1}]$, its associated slice from b_{eq} and a sub matrix from A_{eq}. This returns vectors of lengths $n_x + n_u + n_x$ and a square matrix with sides of the same lengths. If equation (51) is rewritten as

$\begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} - x_{k+1} = 0$

this can be translated to the QP shape as

$$
\begin{bmatrix} -I & 0 & 0 \\ 0 & 0 & 0 \\ A & B & -I \end{bmatrix} \begin{bmatrix} x_k \\ u_k \\ x_{k+1} \end{bmatrix} = \mathbf{0}
\tag{52}
$$

with

$$
\begin{bmatrix} -I & 0 & 0 \\ 0 & 0 & 0 \\ A & B & -I \end{bmatrix} = A_{\text{eq}}
\tag{53}
$$

representing the part of the A_{eq} matrix,

$$
\begin{bmatrix} x_k \\ u_k \\ x_{k+1} \end{bmatrix} = x
\tag{54}
$$

the slice of x and $\mathbf{0}$ the slice of b_{eq}. Note that only the bottom row of the shown part of A_{eq} is the result of the dynamics of $[x_k\, x_{k+1}]$, the first $-I$ is belongs to the same problem but for the combined $[x_{k-1}\, x_k]$ states. For clearity, the matrices A, B and I are square of size n_x, n_u and n_x respectively and I is the identity matrix.

The initial conditions also need to be encoded into the equality constraints. The equation is $x_1 = [0\, x_{\text{measurement}}]^T$, the 0 is the result of the first element of x_1 being $\bar{u}_1$. As x_1 is not part of the vector x the initial conditions can be set by forcing x_2 to be the value expected if u_1 is applied to the system state $x_{\text{measurement}}$. This leads to the following equality constraints for $[u_1\, x_2]$

$$\begin{bmatrix} B & 0 \\ 0 & -I \end{bmatrix} x_1 = - \begin{bmatrix} 0 \\ x_{\text{measurement}} \end{bmatrix} \tag{55}$$

Finally, the A and B matrices need to be derived. As stated in section 2.3.2 it is assumed that the dynamics are known in continuous time, while a discrete time problem is solved by MPC. For nonlinear systems integration methods like forward Euler are used, while the QP formulation requires an A and B matrix. The reason for this is that linear problems can be solved analytically, by writing the dynamics in the state space form and using a discretisation method. Because the control input will remain the same for the entire timestep the zero order hold discretisation will result in exact discretised A and B matrices. These discrete state space matrices can be used as the A and B matrices mentioned in this section.

C.3 Linearizing the T-flex

After using the previous chapter to explain how QP can be used to solve a linear MPC optimisation problem, this chapter will be used to describe how this could be done for the nonlinear T-flex. Hopefully making clear what some of the issues with this approach are.

First the system dynamics, the velocity, acceleration and fb controller dynamics are all linear so pose no problem. Sadly the heat relies on an I^2 term. A linearisation around the current guess of I seems unlikely to work, because it results in the incorrect assumption that a reduction in I will actively cool the system down. It is expected that this caused an attempted ACADOS implementation to fail to find solutions. The author does not know if there are ways around this problem.

If the heating term is ignored the other nonlinear part of the system is the angle dependent torque. Because feedback linearisation is used, the angle dynamics appear linear for the MPC controller. However, the nonlinear relation between the angel and torque does still show up if there are torque constraints. Because the torque constraints apply on the actual torque, and not the MPC control action which is the acceleration, the acceleration needs to be translated to a torque. This can be done by using equation (15). Sadly, as already mentioned the angle-torque relation is nonlinear. One way this could be solved is by using a pessimistic linear approximation of the angle-torque relation. This means a linear relation has to be found where the upper limit can never be below the actual angle-torque function, while the lower limit can never be above it, within the expected operating range.

D Cogging parameters

The parameters $a1$ trough 9 are

$$[0.1172, -0.2366, 0.0746, -0.0354, 0.0043, -0.0091, -0.0094, 0.0013, 0.0005] \tag{56}$$

and $b1$ trough 9 are

$$[-0.1447, -0.0494, 0.0189, 0.0015, -0.0210, -0.0079, 0.0027, -0.0228, 0.0031] \tag{57}$$