Quasi-Linear Model Predictive Control:
Stability, Modelling and Implementation

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von
Pablo Sebastian Gonzalez Cisneros

aus
Monterrey, Mexiko

2021

Prüfungsvorsitzende:

Prof. Dr.-Ing. Gerhard Bauch

Gutachter:

Prof. Dr. Herbert Werner

Dr.-Ing. Hossam Abbas

Tag der mündlichen Prüfung:

22. April 2021

DOI: https://doi.org/10.15480/882.3574


https://orcid.org/0000-0001-8559-9945

Summary

This thesis proposes a Nonlinear Model Predictive Control (NMPC) framework which leverages the quasi-Linear Parameter Varying (quasi-LPV) modelling paradigm. The motivation for this is two-fold: firstly, it enables the online nonlinear optimization entailed by NMPC to be carried out in an efficient manner; and secondly, methods and results from the LPV literature can be used in the context of MPC to derive tractable stability conditions which are both easier to establish and comparatively less conservative than those derived by other methods.

Predictive control has established itself as an attractive methodology, particularly for constrained systems. In practice, all systems exhibit at least input constraints and being able to consider them explicitly allows MPC to exploit, rather than avoid, said constraints, driving the system to its physical limits in a reliable way. Unfortunately, particularly in the nonlinear case, computational complexity can prove to be prohibitive for systems with fast dynamics. The presented approach uses quasi-LPV representations of the plant's dynamics and constraints, resulting in an optimization problem with complexity comparable to that of linear MPC, thereby enabling this control scheme to be applied to fast nonlinear systems with sampling times in the (sub-) millisecond range.

The use of quasi-LPV modelling allows to express nonlinear systems as linear ones, as well as to extend well-known and powerful design and synthesis techniques from linear systems to nonlinear systems. In this thesis, a similar approach is followed to extend stability conditions often encountered in MPC for Linear Time-Invariant (LTI) systems to nonlinear systems. Stability can thus be established by solving a convex optimization problem with Linear Matrix Inequality (LMI) constraints offline and imposing so-called stabilizing terminal constraints on the online optimization problem. A further benefit of quasi-LPV modelling is that input-output (IO) models can be treated as well and the stability analysis follows similar steps to the state space case, making it a viable alternative with the added benefit of relinquishing the need for state estimators.

A velocity algorithm for NMPC is proposed which results from using velocity-based linearization on the nonlinear model. The resulting quasi-linear model can readily be expressed as quasi-LPV, thereby making it suitable for the proposed framework. The velocity algorithm is simpler in design and implementation when compared to standard MPC and the use of velocity linearization makes finding a suitable quasi-LPV parameterization effortless.

Contents

Acronyms and Abbreviations

BMI	Bilinear Matrix Inequality
DMC	Dynamic Matrix Control
DOF	Degrees of Freedom
GPC	Generalized Predictive Control
IQC	Integral Quadratic Constraints
IO	Input-Output
KKT	Karush-Kuhn-Tucker
LFR	Linear Fractional Representation
LFT	Linear Fractional Transformation
LMI	Linear Matrix Inequality
LPV	Linear Parameter-Varying
LTI	Linear Time-Invariant
LTV	Linear Time-Varying
PDQC	Parameter-Dependent Quadratic Constraints
qLMPC	quasi-Linear Model Predictive Control
qLPV	quasi-Linear Parameter-Varying
QCQP	Quadratically Constrained Quadratic Program
QP	Quadratic Programming / Quadratic Program
MIMO	Multiple Input-Multiple Output
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
SISO	Single Input-Single Output
SQP	Sequential Quadratic Programming
SS	State Space

Notation

$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{\geq 0}$	Set of nonnegative real numbers
$\mathbb{R}^n$	Set of real-valued vectors of size n
$\mathbb{R}^{n\times m}$	Set of real-valued matrices of size $n \times m$
$\mathbb{S}^n$	Set of real-valued symmetric matrices of size $n \times n$
$\mathbb{Z}$	Set of integer numbers
$\mathbb{Z}_{\geq 0}$	Set of nonnegative integers numbers
ℓ_2	Space of square summable sequences
ℓ_2	Space of square summable sequences of dimension n
ℓ_{2e}	Extended space of square summable (truncated) sequences
ℓ_{2e}^n	Extended space of square summable (truncated) sequences of dimension n
I_n	Identity matrix of size $n \times n$
$\mathbf{1}_n$	Vector of *ones* of size n
$A^\top$	Transpose of matrix A
A^{-1}	Inverse of matrix A
$A^\dagger$	Pseudo-inverse (Moore-Penrose inverse) of matrix A
$\mathrm{diag}(A, B, C)$	(Block-) diagonal matrix with entries (matrices) A, B, C along the diagonal
$\mathrm{diag}_N(A)$	(Block-) diagonal matrix with entries (matrices) A repeated N times along the diagonal
$A \otimes B$	Kronecker product
$A \succ 0$ / $A \succeq 0$	A is a symmetric positive definite / positive semi-definite matrix
$A \prec 0$ / $A \preceq 0$	A is a symmetric negative definite / negative semi-definite matrix
$\mathcal{F}_u(A, \Delta)$	Upper Linear Fractional Transformation
$\dot{x}(t)$	Time-derivative of the signal $x(t)$, i.e. $\frac{d}{dt}x(t)$
x_k	Discrete-time signal x at time instant k, i.e. $x(k)$
q^{-1}	Backward time-shift operator
Δ	Difference operator, i.e. $(1 - q^{-1})$
$\|x\|_Q^2$	Weighted vector 2-norm, i.e. $x^\top Q x$
$\nabla f(x)$	Gradient of scalar function $f(x)$ / Jacobian of vector-valued function $f(x)$
$\nabla_x f(x, y)$	Gradient of scalar function $f(x, y)$ / Jacobian of vector-valued function $f(x, y)$ with respect to x
$Hf(x)$	Hessian of $f(x)$
$f[A]$	Image of set $A \subseteq X$ under map $f : X \to Y$, i.e. $f[A] = \{f(x) : x \in A\}$
$f^{-1}[B]$	Preimage of set $B \subseteq Y$ under map $f : X \to Y$, i.e. $f^{-1}[B] = \{x \in X : f(x) \in B\}$
$\mathrm{Proj}_B(A)$	Projection of set A onto B
$A_\perp$	Orthogonal complement of set A

Chapter 1

Introduction

Model Predictive Control (MPC) has established itself as one of the most widely used advanced control strategies in industrial applications. The main reason for this is its ability to explicitly consider limits on variables of the system in the form of hard constraints, whose inclusion in the control law is possible due to the premise of the control strategy; namely to plan ahead of execution. The use of predictions to plan a desired response from a system should feel natural, as it to some extent reflects how humans interact with their environment. Likely the most widely used example to illustrate the philosophy of predictive control is driving a vehicle. While driving, the road ahead is observed and the decision to make control changes to the vehicle is based on the information acquired from this observation. When a turn is sighted ahead, for instance, one decides to lower the velocity before the turn occurs so as to gracefully navigate this curve using an internal criterion of what an *optimal* turn is. Likewise, an internal *model* of the car is used to know how much to brake and how much to steer; naturally, the resulting performance is highly dependent on how well the driver knows the car they are driving. i.e. how accurate this model is. The idea of Model Predictive Control (MPC) can be summarized as follows:

MPC relies on a mathematical *model* to make a *prediction* of the system's behavior and choose the correct *control* moves that yield an *optimal* system response, while considering the system's limitations. This optimality is given with respect to a given performance index.

1.1 Motivation

To compute the control moves which make the system's response optimal, complex computations[1] need to be carried out at execution time and within a sampling period. Depending on the system and the sampling period, this might prove prohibitive. In particular nonlinear systems with fast dynamics (i.e. with short time constants) represent a challenging problem since the nonlinear nature makes said computations substantially more complex, and the fact that dynamics are fast precludes using long sampling times that could allow computations to take longer. For this reason, considerable research effort has been spent on the development of algorithms which can carry out the computations entailed by MPC for nonlinear systems in an efficient manner, in order to apply them to systems with fast dynamics.

[1] At least compared to other control strategies.

Beside the computational issues mentioned above, from a system theoretic perspective the use of MPC poses another challenging task. Indeed, the family of finite-horizon optimal controllers to which MPC (and its variants GPC [40], DMC [42], etc.) belongs is not guaranteed to stabilize the system to be controlled. For this reason, this family of control strategies did not receive much attention from academics for over a decade, since its popularization in an industrial context in 1978 [100] until the earliest publications addressing the (at that point) open problem of stability-guaranteeing finite-horizon controllers in late 1980's [66][83].

These two issues are to be addressed in this dissertation. At the core of the developments to follow is the use of quasi-Linear Parameter Varying (quasi-LPV) modelling (Section 2.4). This modelling paradigm was proposed in [109] in the context of gain-scheduled control to enable powerful synthesis and analysis tools tailored to Linear Time-Invariant (LTI) systems, to be applied to time-varying and nonlinear systems. Similarly, in this thesis results and methods from the LTI MPC literature are extended to nonlinear systems, leveraging quasi-LPV modelling; furthermore, several results from the LPV literature are used to improve upon current practices of nonlinear MPC in terms of efficiency and conservatism.

Although admittedly LPV modelling can be considered niche in the context of predictive control, there have been numerous studies which discuss the topic. These can be broadly categorized into MPC for LPV systems and MPC for quasi-LPV systems, given the fundamental limitation that uncertain parameters impose on the former.

1.1.1 LPV Model Predictive Control

Whereas for gain-scheduled LPV control, i.e. non-linear $\mathcal{H}_\infty$ control, it makes no difference at synthesis time whether the LPV system is scheduled by exogenous or endogenous parameters, in predictive control a clear distinction is to be made. In the former case the parameter trajectory cannot be predicted (being external), whereas in the quasi-LPV case parameter trajectories are functions of state and/or input trajectories, thus enabling their prediction.

MPC for LPV systems with exogenous parameters

The bulk of the LPV-MPC literature focuses on this kind of model as, admittedly with conservatism, it can also be used for quasi-LPV systems if one decides not to exploit the knowledge of the future parameter trajectory. With this in mind, most results in LPV-MPC are extensions of Robust MPC [15], with the distinction that the parameter is assumed to be a time-varying but *measurable* bounded uncertainty. An early result, which considers Linear Time-Varying models given a time-varying, unmeasurable uncertain parameter which evolves within a polytopic set is given in [71]. This approach relies on the online solution to an LMI problem to guarantee that the worst case uncertain system is stabilized by using min-max optimization. Although it does not consider LPV systems, it laid the foundation for several results in LPV-MPC for years to come. [79] extended the min-max optimization to include the information acquired from measuring the parameter online, i.e. it extended the approach to the scheduled case. This "quasi-min-max" scheduled algorithm allowed the first stage cost to be computed exactly without uncertainty (given the knowledge of the parameter in the current time instant), and it

therefore reduced the conservatism when compared to previous results. This result was further extended in [81] to consider bounded parameter rates, assuming the bounds on the parameter rates are known. As the parameter dependency is assumed to be polytopic, the uncertain plant descriptions for future time instants evolve in a polytope, and given bounded parameter rate the polytope is a subset of the maximum allowable one, given by the worst-case parameter values. A variant of this result is investigated in [26] where it was observed that the use of closed-loop predictions, using a nominal scheduled state feedback gain, and the possibility to consider longer control horizons lead to an increase in performance. The quasi-min-max algorithm was then extended to the output-feedback problem in [69], where a robust observer is designed offline and the quasi-min-max algorithm is applied online with the estimated state. [120] addresses the conservatism of using a common Lyapunov function by considering parameter-dependent Lyapunov functions and extends the result in [71] to LPV systems, without recurring to min-max optimization. In [131] an MPC law with horizon $N = 1$ is implemented with a dual-mode parameter-dependent state feedback controller; compared to previous results, parameter dependency of the state feedback law gives additional degrees of freedom and reduced conservatism, at the cost of computational load.

An approach akin to tube-based MPC for LPV systems is presented in [115]: closed-loop predictions are carried out and the center system (the system in the center of the future parameter ranges) is used as the center of uncertainty ellipsoids which are built online, parameter range information is built-in to scale the ellipsoids accordingly. In [49] a Linear-Time Varying (LTV) model is used to construct a tube-based MPC, the contribution of this work is that, under the assumption that the LTV model is fully known (i.e. its time-dependence is known a priori), the tubes are adapted online according to the current system dynamics. This procedure has similarities to the gain-scheduled case, in which adaptation is used according to the current operating conditions, although assuming full knowledge of the LTV system enables less conservative results when compared with uncertain parameter variations. An output feedback controller combining an observer for state estimation and a tube-based MPC is presented in [114], where the interaction between observer and controller is considered by bounding the estimation error and absorbing this into the uncertain description of the LPV model, a combination of tube MPC (for disturbances) and the quasi-min-max algorithm (for parametric uncertainty) is then used to robustly stabilize the system.

Most of the results summarized above rely on online solutions to LMI problems in order to guarantee stability, this is of course considerably expensive to compute and as such, the methods are only applicable to relatively slow processes. To address this issue, several results in the literature aim to replace this LMI problem with a linear or quadratic program. In [94], interpolations of mode-2 controllers (i.e. the fixed state feedback law of the dual-mode control) found offline is done by solving a Quadratic Program (QP) online for control of LPV systems. This is possible thanks to the definition of robustly invariant polyhedral sets, which also offer the possibility to consider asymmetric constraints and are generally less conservative than ellipsoids. In [27] an ellipsoidal MPC scheme for LPV systems is presented, wherein a family of indexed ellipsoids is computed offline, which have the feature that an ellipsoid indexed i can be steered in one-step to an ellipsoid indexed $i - 1$ and to a terminal set in at most i steps. The online optimization problem reduces then to a Quadratically-Constrained Quadratic Program (QCQP) or even to a QP for a particular choice of stage cost. [52] introduces the concept of a scheduling-

tube which bounds the future admissible parameter values and can lead to reduced conservatism when compared to min-max approaches, particularly when parameters rates are known, similar to [115]. An advantage of this approach is that it solves a single linear program each time-step and it is shown that complexity scales well with horizon, but rather poorly with system's order as the cross-sections of the tube are considerably complex, thus negatively impacting computational complexity even for low-order systems. This result was then extended in [50] to consider finite-step (or periodic) contractive terminal sets rather than $\rho-$contractive sets, which represents a relaxation of the previous result.

MPC for quasi-LPV systems

As mentioned above, a clear distinction can be made between predictive schemes tailored for quasi-LPV systems and those that consider general LPV systems. This is due to the fact that quasi-LPV systems come from embeddings of nonlinear systems into linear models and are such that the parameters are functions of the state and/or inputs, thus enabling the prediction of the parameter trajectory so that it no longer needs to be treated as uncertain. This is however, not straightforward and there is usually a trade-off between certainty of the future parameter trajectory and the complexity of the solution to the optimization problem, the extreme cases being fully uncertain reverting to LPV systems as mentioned in the previous section and fully nonlinear requiring nonlinear optimization to be performed online. In [10], nonlinear embeddings into polytopic LTV uncertain models are used, considering both input and state constraints; the latter become important because polytopic embeddings of nonlinear systems in this context require future states to be bounded. Complexity of the online optimization is still relatively high because it is based on solution of LMIs. An application of the quasi-min-max algorithm in [79] to nonlinear systems is presented in [80], this follows from the fact that, as before, measuring the parameter (in this case the state) at time instant k enables the prediction of x_{k+1} to be certain via the linearization of the nonlinear model, while the rest of the prediction horizon is handled as before in a uncertain way minimizing the worst-case cost given a polytopic description of the linearization-based quasi-LPV model. Polytopic embeddings are also used by [29], where the issue of computational complexity is also addressed by solving a single QP online. However, the performance index for this optimization problem is the deviation of the receding horizon control law from an auxiliary controller computed offline (along with its corresponding polytopic invariant sets) and hence does not optimize the state trajectory and can lead to suboptimal closed-loop performance. In close relationship with quasi-LPV models, LTV models obtained from successive linearization have also been used to address the issue of computational complexity. In [72], an incremental linear model, obtained from linearizing along a seed trajectory (obtained from the input trajectory in the previous time step), is used to efficiently solve the optimization problem, linearization errors are handled by means of polytopic tubes given bounds on the linearization errors. Although the optimization problem reduces to a linear problem, online computation of the uncertainty sets remains expensive. This was some time later addressed in [25] where the tubes were regarded as ellipsoidal, enabling the use of a quadratic cost in lieu of an l_1 cost. Similarly, [51] considers the use of LPV embeddings and tubes both in state and in the so-called scheduling sequences, where as opposed to previous work in tube-based LPV MPC, the authors exploit the knowledge of the functional dependency

of the parameter on the state. In [12], the Takagi-Sugeno Fuzzy modelling framework is used to express the nonlinear system as a polytopic linear embedding (i.e. a polytopic quasi-LPV model) and an iterative algorithm is proposed to use state and input trajectories of the previous iteration to compute the affine parameter sequence to be used on the next iteration.

The work presented in this thesis best fits into this category, and most of it has been reported in a collection of papers: similar to works mentioned in this section, in [33] the use of exact quasi-LPV models with general parameter dependence (i.e. not restricted to affine) is exploited to solve the nonlinear optimization problem as a sequence of QP problems: predicted state trajectories from previous time steps/iterations are used to schedule the model matrices in order to solve the optimization problem as an LTV problem. Stability analysis is carried out by means of terminal ingredients computed offline. This approach was extended to consider parameter rate information and parameter dependent terminal ingredients in [37], computational tractability of the offline optimization problem was addressed later in the journal paper [48]; these results are presented in Chapter 3. An extension of this approach, with a velocity implementation, was presented in [35]. The use of velocity algorithms makes establishing stability, particularly in the case of unreachable set point tracking, simpler and more meaningful. In a different approach, stability of the algorithm developed in [33] is established a priori and offline using dissipativity arguments and parameter dependent quadratic constraints in [34], this discussion can be found in Chapter 7.

1.1.2 Input-Output LPV systems

A further advantage of using LPV as the modelling paradigm of choice is that there is also a relatively rich body of literature discussing the input-output (IO) framework. Input-Output LPV models arise mainly when LPV system identification is performed using the parameter-dependent extension of the prediction error method [14]. However, first principle models can also be readily obtained by appropriate discretization of the continuous-time nonlinear dynamic equations. The main motivations to use IO in lieu of state space (SS) models is twofold: firstly, the use of potentially complex (especially in the nonlinear case) state observers is forgone; secondly, system identification methods extending the prediction error method to LPV systems are comparatively simpler than subspace identification methods, and for the most part similarly accurate [105].

Most LPV control synthesis techniques are given in a state-space (SS) framework, and as pointed out by [118], converting an I/O model to SS brings forth the issue of dynamic dependence (dependence on time-shifted parameter values) on one hand, or results in static-parameter-dependent non-minimum SS realizations on the other hand. To circumvent these issues, which might be prohibitive in some cases, controller design can be carried out directly in the I/O framework. A gain-scheduled $\mathcal{H}_\infty$ single-input single-output (SISO) controller synthesis result is presented in [47] where a linear matrix inequality (LMI) formulation given a somewhat heuristic choice of a central polynomial is derived. Based on this result, a systematic bilinear matrix inequality (BMI) formulation is presented in [9] yielding a fixed-structure polytopic LPV controller. This approach was then extended to multiple-input multiple-output (MIMO) controller synthesis in [8]. When closing the loop around a scheduled controller, dynamic dependence should also be considered, this issue had been neglected in the literature mentioned

above. An approach which considers, and indeed circumvents this issue by using dynamic constraints and Finsler's Lemma, is presented in [126], this result yields a fixed-structure gain-scheduled controller and is again given by the solution of a BMI problem. In [127], the use of so-called 'image representation' LPV I/O models enables I/O LPV controller synthesis to be cast as a state feedback problem and the use of convexifying change of variables turns the BMI condition into an LMI.

Input-Output LPV Model Predictive Control

This area of research remains relatively unexplored with only a few rather recent results in the literature. In [2], similar to several results above, the future scheduling trajectory is considered uncertain and worst-case minimization is carried out online by means of an LMI problem. Stability is established by terminal ingredients obtained offline by the solution of a non-convex Bilinear Matrix Inequality (BMI) approach. This approach is extended to handle MIMO IO models in [4], this extension also recasts the offline optimization to be solvable as an LMI problem and in addition considers a terminal constraint set for each reference, rather than the restrictive condition of requiring all set points to be contained in the unique terminal set, as before. [53] makes use of non-minimal state space realizations formed with previous input and output data to establish stability in the SS setting, that is, with ellipsoidal terminal state regions and a dual-mode controller. The somewhat unrealistic assumption that the future scheduling trajectory is exactly known is made in order to perform convex online optimization. As elaborated in Chapter 5, in [38] the image-representation analysis from [127] is burrowed to establish stability of an IO-quasi-LPV MPC scheme which builds upon the result presented in [33]. This enables the problem of finding terminal ingredients to be solved as an LMI problem, while online optimization remains efficiently solvable as a sequence of QP.

1.1.3 Efficient Nonlinear MPC algorithms

As mentioned above, in particular MPC for fast nonlinear systems represents a challenging problem, since complex computation must be carried out online within a sampling period. The interest in the clear benefits of MPC with regards to constraint handling and optimizing nature, which can and has been leveraged with economics in mind [99], has sparked an interest in solving said complex computations efficiently. Beside the methods mentioned above which make use of quasi-LPV modelling or linearization along reference trajectories to make the nonlinear dynamic equations linear, methods based on established numerical optimization algorithms but adapted to a real-time context are commonly used. Newton type algorithms and Sequential Quadratic Programming (SQP) are used for example in the real-time iterations algorithm from [44] which implements a direct multiple-shooting approach [17]. This algorithm, which has in the mean time gained a lot of traction given its user-friendly implementation in the toolkit ACADO [59], performs a single Newton step at each sampling instant and is therefore quite efficient. Although the real-time iteration scheme is likely the most popular in the research community, it was not the first of its kind; indeed SQP resulting from Newton type optimization in the context of predictive control was also proposed almost two decades prior in [74], although no stability proof was given for the nonlinear case. The Generalized Minimal Residual method

(GMRES) is used in the Continuation/GMRES method from [90], in contrast to the previously mentioned approaches, it used an Interior Point handling of inequality constraints. The projected gradient method is used in the software GRAMPC [63](GRAdient-based MPC) which is tailored for real-time implementation of the control law, this is achieved by prematurely stopping the gradient iterations. The interior-point method (popularized by [65]) is also frequently used for NMPC problems; one of the most widely used toolboxes is the open-source IPOPT [119] which efficiently implements the interior point method for general nonlinear optimization problems.

This short literature review is by no means comprehensive, it merely points out relevant results which have been proven to be quite efficient and real-time capable. At the core of all these approaches is the leveraging of the MPC problem structure to simplify its recursive solution. Indeed, in MPC at each subsequent time step an optimization problem is solved which is almost identical to the one solved at the previous time step (see Section 2.2.2), this allows to use the resulting system trajectories to bootstrap the optimization at the next time step.

1.2 Contributions

The contributions of this thesis can be seen from two perspectives, the numerical complexity and the system theoretic one (i.e. stability analysis). Both of these leverage the quasi-LPV modelling framework to extend LTI results to the nonlinear realm.

1.2.1 Numerical complexity

A numerical algorithm deemed quasi-Linear Model Predictive Control (qLMPC) arising from the use of quasi-LPV models of nonlinear systems to solve the optimization problem efficiently is proposed. As most other efficient NMPC algorithms, the idea at the core is to use the predicted trajectories of the previous time step to bootstrap the optimization problem; particularly, the input sequence from the previous solution is used to predict a state sequence, these two trajectories are used to compute a scheduling sequence, which in the quasi-LPV case is given as a function of state and input. A fixed scheduling trajectory makes the system linear (but time-varying), and leads to an optimization problem which can be solved efficiently (by solving a QP). Iterations, similar to those encountered in SQP can be performed within a time step to speed up the convergence (which would otherwise stretch over several time steps); this is particularly relevant when so-called stabilizing terminal constraints are used so that the predicted trajectory matches the real system's trajectory as closely as possible.

1.2.2 Stability Analysis

Stability analysis based on the well-know *Dual-Mode Control* [85] concept is carried out. Compared to current practices, where the second-mode fixed state feedback controller is designed based on a linearization of the nonlinear system around the desired (set point) equilibrium, in this thesis the use of quasi-LPV modelling gives an exact representation of the nonlinear dynamics and tractable stability conditions in the form of Linear Matrix Inequalities (LMIs) are proposed.

The quasi-LPV representation need not be restricted to exhibit affine parameter dependence as in similar approaches, thereby greatly reducing conservatism by avoiding severe over bounding of the admissible parameter set, an effect found whenever affine parameter dependence is artificially imposed by defining additional, functionally dependent parameters (which happens in most real world applications). Furthermore, a novel approach to reduce conservatism is proposed by considering parameter dependent *terminal ingredients*, that is, a scheduled state feedback second mode controller, terminal cost function and terminal constraint set. The additional degrees of freedom can be used to increase the size of the terminal region, improving feasibility and enlarging the region of attraction.

A novel nonlinear *velocity form* MPC is proposed, which uses velocity-based linearization to obtain a quasi-LPV model of the nonlinear plant. The advantage of using an incremental (velocity) form MPC is clear as all equillibria are mapped to the origin of the velocity space. This enables tracking for nonlinear systems without complex steady state parameterization (i.e. without the need to parameterize steady state input and state values as a function of a desired output reference); furthermore integral action is built-in on this framework. Compared to the standard stabilizing MPC, design is considerably simpler, as no terminal ingredients need to be computed.

An input-output version of stabilizing qLMPC is proposed, for which stability is enforced (as in the standard state space formulation) by using stabilizing terminal ingredients. These are found by offline solution to an LMI problem equivalent to a certain state feedback problem.

An approach for stability analysis of qLMPC by means of a dissipation inequality is proposed, this is made possible by a characterization of the nonlinearity arising from a QP as a Parameter Dependent Quadratic Constraint (PDQC). If the system to be controller fulfills certain conditions (specifically, if the equilibrium to be stabilized is open-loop stable and $u = 0$ is always feasible) then stability can be established a priori using this result. Terminal ingredients, and even stabilizing terminal constraints are therefore not necessary and both feasibility and stability are guaranteed without affecting performance or computational complexity, both of which could potentially be negatively impacted by imposing such constraints.

1.2.3 Practicality Oriented Approach

Whereas qLMPC in its standard form is already an approach which tailors to practical implementation, given its simplicity and computational efficiency, a relatively accurate model is needed. A model might not be available so that in practice, the usual approach is to perform system identification; in this regard, the IO version of qLMPC might be better suited as IO-LPV identification is still comparatively simpler than its SS counterpart. Alternatively, data-driven control techniques could be employed. Indeed, a data-driven control scheme is proposed which uses the Koopman Operator framework coupled with velocity-based qLMPC. The result is a predictive control scheme which *discovers* the dynamics of the systems in real time given a set of basis functions while stabilizing the system.

It is worth mentioning that most of the results proposed in this dissertation have been experimentally validated, and indeed exhibit exceptional performance. The application examples chosen

to validate the controllers are a PenduBot (or Arm-Driven Inverted Pendulum), a 2-DOF robotic manipulator (with and without nonlinear constraints) and a Control Moment Gyroscope.

1.3 Organization of the Thesis

Before discussing the main topic of the dissertation, several important mathematical and control theoretic concepts are introduced in Chapter 2. The first part of the chapter is dedicated to the definition of mathematical and system and set theoretic concepts to be used throughout this work. The latter part discusses preliminaries of the two central topics of the dissertation, namely Model Predictive Control and Linear Parameter-Varying modelling; it is written assuming little if any familiarity with these topics in order to make the rest of the thesis easily digestible.

Chapter 3 presents the foundation of the qLMPC framework in the state space setting. An iterative algorithm is presented, which can be used to efficiently solve the nonlinear MPC optimization problem; stability conditions in the form of LMI problems are derived, whose offline solution yield the terminal ingredients (terminal cost and terminal constraint set) to be used in the online MPC law. The problem of offset-free tracking and nonlinear constraints is discussed as well. The control law is tested experimentally on an Arm-Driven Inverted Pendulum (PenduBot).

A velocity-form nonlinear MPC is presented in Chapter 4. The stability result in this case makes use of terminal equality constraints in the velocity space, making its implementation comparatively simple (as no terminal ingredients need to be computed). The use of velocity-based linearization and state augmentation allows to straightforwardly consider nonlinear output equations. The velocity algorithm is tested on a 2-DOF robotic manipulator considering both nonlinear constraints and nonlinear output.

The qLMPC framework is extended to consider input-output quasi-LPV models in Chapter 5. The stability analysis of the state space framework is appropriately modified for this kind of models and the tracking problem is discussed in more detail as the stability conditions tailor better to tracking in an IO setting. The IO-qLMPC law is validated on a 2-DOF robotic manipulator.

A data-driven predictive control law based on Koopman operators and qLMPC is presented in Chapter 6. A short overview of the Koopman framework is given and an algorithm to compute the Koopman operator online is derived. The Koopman-based lifted linear model is used in conjunction with the qLMPC framework resulting in a data-driven control strategy. This approach is tested experimentally on a 4-DOF Control Moment Gyroscope.

A stability analysis tool for LPV MPC is proposed in Chapter 7. Under certain assumptions about the nonlinear system, the nonlinearity arising from a QP can be characterized by a so-called Parameter Dependent Quadratic Constraint. This characterization is used to derive a dissipation inequality and establish stability of the closed-loop a priori without artificially imposed stabilizing constraints.

Chapter 2

Preliminaries

This chapter introduces the reader to many of the mathematical and system and control theory concepts which are relevant in the present context. The presentation and developments here are brief but encompass much of the prerequisites to easily follow the rest of the thesis.

2.1 Mathematical Preliminaries

Before introducing the two central topics of this thesis, namely Model Predictive Control (MPC) and quasi-Linear Parameter-Varying (quasi-LPV) modelling, several useful mathematical tools, definitions and results to be used throughout this work are summarized in this section. The section is intended to serve both as the definition of these concepts and as a quick reference for the reader.

2.1.1 Optimization Problems

The following definitions deal with relevant types of optimization problems, the interested reader is referred to [20] and [21] for a more thorough discussion.

Definition 2.1 (Quadratic Program)**.** A Quadratic Program (QP) is an optimization problem of the form

$$\begin{aligned} \min_{x} \quad & x^\top H x + g^\top x + r \\ \text{subject to} \quad & Ax \leq b \\ & A_{eq} x = b_{eq} \end{aligned}$$

This family of optimization problems represents the most widely used structure in the context of Model Predictive Control. It arises naturally when using quadratic cost functions and the system's dynamics are linear; however, even for Nonlinear MPC (NMPC) it is frequently used when using second-order approximations of the nonlinear optimization problem (e.g. Newton's method).

Definition 2.2 (Linear Matrix Inequality)**.** A Linear Matrix Inequality (LMI) is a relationship expression of the form

$$F(x) = F_0 + \sum_{i=1}^{m} F_i x_i \succ 0$$

where $x \in \mathbb{R}^m$ is the variable and $F_i = F_i^\top \succ 0$ are constant matrices.

The inequality symbol ($\succ$) is used in this context to denote that the matrix is positive definite, i.e. $M \succ 0 \Leftrightarrow x^\top M x > 0 \; \forall x \neq 0$. A generalization of LMIs which, although often found, are not as attractive as LMIs (for reasons to be discussed in what follows) are Bilinear Matrix Inequalities.

Definition 2.3 (Bilinear Matrix Inequality)**.** A Bilinear Matrix Inequality (BMI) is a relationship expression of the form

$$F(x) = F_0 + \sum_{i=1}^{m} F_i x_i + \sum_{j=1}^{n} G_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} H_{ij} x_i y_j \succ 0$$

where $G = G^\top \succ 0$, $H = H^\top \succ 0$.

LMIs, and to a somewhat lesser extent BMIs, are routinely used to express stability and performance metrics for controller synthesis. Indeed the variables in the LMI (BMI) are related to the controller and finding a solution yields a controller that meets the required specifications. A solution for the LMI case is found by solving a semidefinite program.

Definition 2.4 (Semidefinite Program)**.** A Semidefinite Program (SDP) is an optimization problem of the form

$$\begin{aligned} \min_x \quad & g^\top x \\ \text{subject to} \quad & F(x) = F_0 + \sum_{i=1}^{m} F_i x_i \succ 0 \end{aligned}$$

An important characteristic of SDPs is that they are convex, which guarantees uniqueness of a solution, if one exists (i.e. if the problem is feasible). Examining the structure of the problem, it is clear that BMIs cannot be included in an SDP and for that case other (often heuristic) methods have to be used to solve the optimization problem. Given the central role played by LMIs in some of the derivations of this thesis, and the fact that SDP encompass a much richer class of optimization problems (indeed QPs are a special case) SDPs with LMI constraints are henceforth referred to simply as *LMI problems*, correspondingly optimization problems subject to BMI constraints are referred to as *BMI problems*.

When faced with a BMI problem with a certain structure, there are several tools that can be used to convexify the problem i.e. to turn the BMI into an LMI. The simplest is a so-called *linearizing change of variable*, which, by means of defining a new variable $z_k = x_i y_j$ can turn the problem linear. Another very useful tool is presented in the following Lemma.

Lemma 2.1 (Schur Complement)**.** The matrix inequalities

$$S(x) \succeq 0,\ Q(x) - R(x)^\top S(x)^{-1} R(x) \succeq 0$$

where $Q(x), R(x), S(x)$ are affine functions of x, are equivalent to the LMI

$$\begin{bmatrix} Q(x) & R(x)^\top \\ R(x) & S(x) \end{bmatrix} \succeq 0.$$

2.1.2 Stability

This section lists several definitions and theorems useful for establishing stability of equilibria of nonlinear systems. These are standard in nonlinear control literature e.g. [67]. Consider the dynamic system

$$\dot{x} = f(x).$$

In what follows it is assumed that the equilibrium of interest (i.e. the one for which stability is to be analyzed) has been appropriately shifted to the origin so that $\bar{x} = 0$.

Definition 2.5 (Lyapunov Stability)**.** The equilibrium point $\bar{x} = 0$ is said to be stable (in the sense of Lyapunov) if $\forall \epsilon > 0$, $\exists\ \delta(\epsilon) > 0$ such that

$$\|x(0)\| < \delta \implies \|x(t)\| < \epsilon, \quad \forall t \geq 0.$$

The previous definition is a somewhat weak statement of stability as it implies that the state remains within a neighborhood of the equilibrium, but not necessarily that it converges to it. A stronger statement would be to say that, not only should the trajectory remain within a neighborhood of the equilibrium point, but that it ultimately convergences to it.

Definition 2.6 (Asymptotic Stability)**.** The equilibrium point $\bar{x} = 0$ is said to be asymptotically stable if it is stable according to Definition 2.5 and

$$\exists \delta > 0\ :\ \|x(0)\| < \delta \implies \lim_{t \to \infty} x(t) = 0$$

Lyapunov stability is the most widely used method for stability analysis for nonlinear systems, as well as for time varying and uncertain linear systems. The reason for this is that it can be readily characterized by existence of an energy-like function that fulfills certain conditions.

Definition 2.7 (Lyapunov function)**.** A function $V(x) : \mathbb{R}^n \to \mathbb{R}$ is called a Lyapunov function if $\exists\ r > 0$:

- $V(0) = 0, \quad V(x) > 0, \quad 0 < \|x\| < r$
- $\dot{V}(x) = \nabla V \frac{dx}{dt} = \nabla V f(x) \leq 0\,, \quad 0 < \|x\| < r$

Note that both conditions are local in nature as they must hold only within a *ball* of radius r. The connection between existence of a Lyapunov function and stability as defined above is given by the following theorem.

Theorem 2.1 (Lyapunov Stability Theorem)**.** The equilibrium $\bar{x} = 0$ is stable if there exists a Lyapunov function for the system. If, in addition, $\dot{V}(x) < 0,\ 0 < \|x\| < r$ then the equilibrium is (locally) asymptotically stable.

Remark 2.1. *The definitions and results in this section are not the most general; stability in this context is characterized as what is often referred to as* uniform *stability. The difference is that the general definitions can depend explicitly on time, while their uniform counterparts do not and are therefore more restrictive. Likewise, the Lyapunov function can depend explicitly on time; this case is not discussed in this context.*

All the definitions above assume a continuous-time system. It turns out that all the definitions apply to a discrete time system of the form $x_{k+1} = \tilde{f}(x_k)$, however, the stability conditions on the Lyapunov function need to be redefined.

Definition 2.8 (Lyapunov function (discrete-time))**.** A function $V(x) : \mathbb{R}^n \to \mathbb{R}$ is called a Lyapunov function if $\exists\ r > 0$:

- $V(0) = 0, \quad V(x) > 0, \quad 0 < \|x\| < r$
- $\Delta V(x) = V(\tilde{f}(x)) - V(x) \leq 0\,, \quad 0 < \|x\| < r$

The Lyapunov stability theorem holds for the discrete-time case as well, with the only modification being that for asymptotic stability, the *Lyapunov difference* ΔV (as opposed to its time derivative) needs to be strictly negative.

2.1.3 Set Theory

Set theoretic results are frequently used in the context of MPC. For this reason, some important definitions are listed below ([21]).

Definition 2.9. A set S is said to be convex if

$$\forall x_1, x_2 \in S, \lambda x_1 + (1-\lambda)x_2 \in S, 0 \leq \lambda \leq 1$$

Definition 2.10 (Ellipsoid)**.** An ellipsoidal set centered at $\bar{x}$ is defined by the inequality

$$\mathcal{E} = \left\{x \ : \ (x-\bar{x})^\top W(x-\bar{x}) \leq 1\right\}, \quad W = W^\top \succ 0$$

An important characteristic of ellipsoidal sets is that their volume is proportional to $\det(W^{-1/2})$, this fact is used to find maximum volume ellipsoids which fulfill certain constraints.

Definition 2.11 (Sublevel set)**.** A sublevel set of a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ is the set given by

$$L_\alpha = \{x \ : \ f(x) \leq \alpha\}.$$

All sublevel sets of convex function are convex; in particular, a sublevel set of a quadratic form $x^\top Px$ is an ellipsoid.

Definition 2.12 (Convex Hull)**.** The convex hull of a collection of vector x_i is the set defined as

$$\mathrm{Co}(x_1, x_2 ..., x_n) = \left\{ \sum_{i=1}^{n} \lambda_i x_i \ : \ \sum_{i=1}^{n} \lambda_i = 1, \ \lambda_i \geq 0 \ \forall i \right\}.$$

Definition 2.13 (Invariant set)**.** A subset $\mathbb{X}$ of the state space is said to be positively invariant with respect to the dynamic system $x(k+1) = f(x(k))$ if $f(x(k)) \in \mathbb{X}, \forall x(k) \in \mathbb{X}$.

Intuitively, once a trajectory of the system $x(k+1) = f(x(k))$ enters an invariant set, it will never leave it.

2.2 Model Predictive Control

This section is intended to provide an overview of Model Predictive Control (MPC) fundamentals and can be skipped without affecting readability of subsequent sections.

Recall the general definition of MPC given in the first paragraph of Chapter 1:

> MPC relies on a mathematical *model* to make a *prediction* of the system's behavior and choose the correct *control* moves that yield an *optimal* system response, while considering the system's limitations. This optimality is given with respect to a given performance index.

With this general definition in mind, this section briefly introduces each of the highlighted concepts in an abstract way, these will then be defined more precisely in the rest of the chapter.

2.2.1 Mathematical Model and Prediction

A model is a mathematical tool that is used to characterize the dynamic behavior of a system. Most systems of interest can be described by a set of differential equations

$$\begin{aligned} \dot{x}(t) &= \tilde{f}(x(t), w(t), t) \\ y(t) &= \tilde{h}(x(t), w(t), t) \end{aligned} \tag{2.1}$$

where x is the state of the system and w is an external input. Often, particularly in the context of MPC, a discrete-time model is preferred, which describes the evolution of the system states only *at* a countable set of instants at which sampling takes place, these are called the *sampling instants*:

$$\begin{aligned} x(kT+1) &= f(x(kT), w(kT), kT) \\ y(kT) &= h(x(kT), w(kT), kT) \end{aligned} \tag{2.2}$$

where T is the sampling interval (or sampling time) and $k \in \mathbb{Z}_{\geq 0}$. In what remains of this work the shorthand notation k is used in lieu of kT and the compact notation $x_k = x(k)$ is used throughout. Given a suitable model, the nominal future response of the system[1] is fully characterized by its initial condition and the future input trajectory, a typical prediction scenario for a Single Input-Single Output (SISO) system is shown in Figure 2.1. Here the *prediction horizon* (i.e. how far into the future the output is predicted) is given by $N_y = N_2 - N_1$ whereas the so-called *control horizon* (how far into the future the input is predicted) is denoted by N_u, where $N_2 \geq N_u - 1$; a non-zero N_1 represents a model time-delay (i.e. system's dead time). For cases in which $N_2 \neq N_u - 1$ it is often assumed that the input remains constant on the interval $[N_u - 1 \quad N_2]$.

It is apparent from Figure 2.1 that the output *anticipates* the reference, i.e. the system reacts to a reference change before it occurs. This is often observed in MPC when the reference is known a priori and is one advantage of predictive control that is somewhat overlooked, but that might prove beneficial (recall the vehicle example).

[1] Nominal in this context refers to the case in which there are no disturbances or uncertainties. In reality, a model that fully describes a system is virtually impossible to find, as uncertainties, external disturbances and unmodelled dynamics are always present.

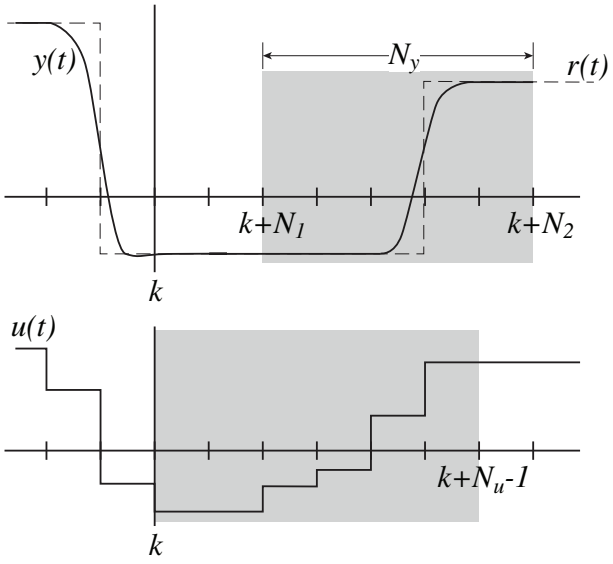


Figure 2.1: Prediction and control horizons. The shaded regions correspond to the data that is being forecast by the prediction.

In order for predictions to be meaningful, physical limitations of the system should be taken into account. The most common ones, and indeed ubiquitous for any real system, are saturation of the control inputs, these are a consequence of physical limitations of the actuators e.g. a valve cannot further increase the flow when it is already fully open. Moreover, many systems exhibit limitations in their states as well, for instance robotic manipulators often have a limited workspace, i.e. by construction, joint angles cannot exceed certain values. These examples are illustrated in Figure 2.2. Clearly, prediction errors would arise if such limitations are not considered. The way they are included in the prediction is by means of *constraints*, thereby providing MPC with one of its main advantages over non-predictive schemes (in addition to anticipative behavior), namely the possibility to explicitly consider constraints. Whereas non-predictive schemes are usually tuned as to avoid saturation altogether, predictive controllers can exploit these limitations to achieve optimal performance. Practitioners often find it convenient to include constraints which do not arise from physical limitations but are rather imposed artificially, for instance, a self-driving vehicle can physically drive out of the highway, but this is by no means desired, so a constraint can be imposed to prevent this from happening.

Constraints are often represented by means of inclusions of the form $u \in \mathcal{U}, x \in \mathcal{X}$, that is, input and/or states belong to a set of admissible values. The way these sets are characterized strongly influences the efficiency of the solution of the MPC problem. Indeed if the sets are convex, and particularly polyhedral, the solution would be found much more efficiently (and accurately) than if the sets are non-convex. For example, characterizing the non-convex set $\mathcal{W}$ in Figure 2.2 in Cartesian coordinates would be quite complex, however, an equivalent representation of the constraint can be expressed as a box constraint in the joint coordinates as $-45° \leq \theta_1 \leq 0°$, $-100° \leq \theta_2 \leq 100°$. The topic of constraints will be examined further in Section 3.4.

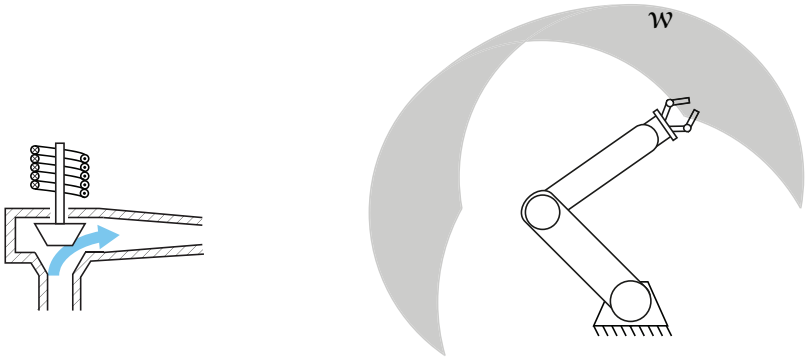


Figure 2.2: Example of an input constraint (left): a valve cannot further increase the flow when it is already fully open. Example of a state constraint (right): the workspace of a robot is limited by its construction, joint angles cannot exceed certain values.

2.2.2 Optimization problem

Having defined the prediction model, the next step is to compute the control inputs that make the response optimal. However, the notion of optimality needs to be defined first, and special care must be taken by considering the limitations of the system.

The notion of optimality, defined by a suitable performance criterion, is often encoded into a cost function that is to be minimized. The variables of this optimization problem are the future control inputs and the future state trajectory. Nevertheless, as mentioned earlier, the state sequence is fully defined by the inputs, given an initial state, therefore the *degrees of freedom* of the optimization are the future control inputs alone, and the problem is subject to the constraint that state trajectories must be admissible trajectories of the system. The cost function used for MPC is often chosen as (assuming for simplicity of exposition that $N_1 = 0$)

$$J_k = \sum_{i=0}^{N_u-1} \ell(x_{k+i}, u_{k+i}) + \sum_{i=N_u}^{N_2} \ell(x_{k+i}, u_{k+N_u}) + \Psi(x_{k+N_2}) \tag{2.3}$$

where $\ell(x, u)$ is called the *stage cost* and $\Psi(x)$ the *terminal cost*. The stage cost is often chosen as a quadratic function penalizing the weighted 2-norm of the tracking error and of the input. There is, however, a branch of MPC in which an economic cost is directly penalized, lending it the name Economic MPC (EMPC) [99], the cost in EMPC is in general not quadratic and it need not be convex, making analysis more difficult. In this work the focus is only on the former, *tracking* MPC, where the stage cost is usually quadratic and of the form

$$\ell(x, u) = ||\tilde{x}||_Q^2 + ||\tilde{u}||_R^2 \tag{2.4}$$

where $\tilde{x} = x - x_s$, $\tilde{u} = u - u_s$ represent deviation of the desired set point x_s, and steady-state input u_s, respectively. The matrices $Q \geq 0$ and $R > 0$ are weighting matrices used to prioritize certain states or input channels in the optimization. The terminal cost is usually chosen quadratic as well and of the form $||x||_P^2$, although as presented in Section 2.3.3 it is strongly linked to how stability is enforced in MPC, and hence P is often not used as a tuning parameter in the same way Q and R are.

The optimization problem in MPC is therefore given by

$$\begin{aligned} &\min_{x,u} J_k \\ &\text{subject to} \\ &\quad x_{k+i+1} = f(x_{k+i}, u_{k+i}) \quad i \in [0 \quad N_2 - 1] \\ &\quad x \in \mathcal{X} \\ &\quad u \in \mathcal{U} \end{aligned} \tag{2.5}$$

Note that this problem is equivalent to the discrete LQ-problem if $f(\cdot, \cdot)$ is linear, $N_u, N_2 \to \infty$ and $\mathcal{X} = \mathbb{R}^n$, $\mathcal{U} = \mathbb{R}^m$, where n, m are the dimensions of the state and input, respectively. However, as soon as constraints, other than the system dynamics, are included in the optimization, the infinite-horizon problem becomes intractable, thereby requiring the solution to a finite horizon problem instead. What might not be obvious is when this problem is to be solved: the fact that it is only optimizing the behavior of the system on a finite horizon suggest that it should be solved online, but one might be tempted to think that, given that the next N_u control inputs are optimized, solution of the problem is only required at intervals $N_u T$, this is however not the case and to better understand this, the discussion below introduces the concept of *receding horizon*.

Receding Horizon

Coming back to the example of driving a vehicle, control decisions made by the driver are based on observations of the road ahead up to a certain distance, the *horizon*. As the vehicle moves forward, so does this horizon, approximately at the same pace, i.e. this horizon is *receding* away at the same speed of the vehicle. As new obstacles are sighted within the horizon, updated decisions are made considering this new information. In much the same way, MPC optimizes the response of the system over the horizon, and at each sampling instant, updated decisions are made based on the new information made available to the controller. This implies that, even though the controller has a *plan* for the next N_u inputs to the system, only the first element of the input trajectory is applied to the system and the plan is updated at each sampling instant (i.e. the optimization problem is solved again) to account for new information obtained.

The main motivation for implementation of the controller in a receding horizon fashion is to provide robustness against unforeseen disturbances caused either by external effects or by model mismatch. In other words, the receding horizon concept is what *closes the loop* around the plant and controller at every sampling instant.

2.2.3 Summary

The MPC control law is given by the solution of the following optimization problem at each time instant k

$$\min_{x,u} \sum_{i=0}^{N_u-1} ||\tilde{x}_{k+i}||_Q^2 + ||\tilde{u}_{k+i}||_R^2 + \sum_{i=N_u}^{N_2} ||\tilde{x}_{k+i}||_Q^2 + ||\tilde{u}_{k+N_u}||_R^2 + \Psi(x_{k+N_2})$$

subject to

$$x_{k+i+1} = f(x_{k+i}, u_{k+i}) \quad i \in [0 \quad N_2 - 1]$$
$$x \in \mathcal{X}$$
$$u \in \mathcal{U}$$

The first element of the input trajectory is applied to the plant, and on the next time instant the process is repeated.

2.3 MPC for LTI Systems

Although the focus of this work is on nonlinear systems, reviewing the Linear Time Invariant case is relevant, in particular because it serves as the foundation for the quasi-LPV tools to be presented in Chapter 3.

2.3.1 State Space Models

Consider the discrete-time Linear Time Invariant (LTI) state space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k, \end{aligned} \tag{2.6}$$

where $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the input and $y_k \in \mathbb{R}^l$ is the output of the system. For simplicity it is assumed that there is no feedthrough from input to output, i.e. $D = 0$, this is true in most applications.

It makes sense to restrict the discussion to linear constraints in order to exploit the problem structure that arises, and be able to solve it efficiently, so for now the discussion focuses on this kind of constraints. Polyhedral constraints can be expressed via the inequalities

$$\tilde{G}_x x_k \leq \tilde{h}_x \qquad \tilde{G}_u u_k \leq \tilde{h}_u,$$

and by defining the future state and input trajectories in vectors

$$X_k = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N_2} \end{bmatrix} \qquad U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N_u-1} \end{bmatrix}$$

the same constraints can be written in vector form for all future instants within the horizon as

$$G_x X_k \leq h_x \qquad G_u U_k \leq h_u$$

where $G_x = \text{diag}_{N_2}(\tilde{G}_x)$, $G_u = \text{diag}_{N_u}(\tilde{G}_u)$, $h_* = \begin{bmatrix} \tilde{h}_*^\top & \tilde{h}_*^\top & ... \end{bmatrix}^\top$. Note that this kind of constraint encompasses the often encountered box constraints with the choices $G_u = [I \quad -I]^\top$, $h_u = [(\mathbf{1} \otimes u_{max})^\top \quad -(\mathbf{1} \otimes u_{min})^\top]^\top$ and similarly for the state, but also more general constraints given by linear combinations of the states or inputs.

Solving the state equation (2.6) recursively yields an expression for the predicted trajectory in terms of the current state x_k and the future input trajectory U_k

$$X_k = \Phi x_k + \Gamma U_k \tag{2.7}$$

where

$$\Phi = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^i \\ \vdots \\ A^{N_2} \end{bmatrix}, \qquad \Gamma = \begin{bmatrix} B & 0 & 0 & ... & 0 \\ AB & B & 0 & ... & 0 \\ A^2B & AB & B & ... & 0 \\ \vdots & & & \ddots & \vdots \\ A^{i-1}B & A^{i-2}B & A^{i-3}B & ... & \sum_{j=0}^{i-N_u} A^j B \\ \vdots & & & \ddots & \vdots \\ A^{N_2-1}B & A^{N_2-2}B & A^{N_2-3}B & ... & \sum_{j=0}^{N_2-N_u} A^j B \end{bmatrix}.$$

Note that, if prediction and control horizon are equal, i.e. $N_2 = N_u$, matrix Γ reverts to the familiar lower block-triangular matrix with B repeated along the diagonal.

Using the aforementioned definition of the future trajectories, the cost function (2.3) can be written in vector form as

$$\tilde{J}_k = (X_k - X_s)^\top \hat{Q} (X_k - X_s) + (U_k - U_s)^\top \hat{R} (U_k - U_s) \tag{2.8}$$

where $\hat{Q} = \text{diag}(Q, Q, ..., P)$, $\hat{R} = \text{diag}_{N_u}(R)$, and $X_s = \mathbf{1} \otimes x_s$, $U_s = \mathbf{1} \otimes u_s$. Note that $\tilde{J}_k = J_k - x_k^\top Q x_k$ so the value of the cost is not equal to the one defined on (2.3), however as $x_k^\top Q x_k$ is constant at time step k the minimizer of both cost functions is identical, but (2.8) is a more convenient representation of the cost function, especially when solving the optimization problem as a Quadratic Program (QP).

Sparse vs. dense formulation of the MPC optimization

The optimization problem derived so far leads to the so-called *sparse* formulation of the MPC optimization problem, by rearranging the variables of the optimization in a vector $Z_k = \begin{bmatrix} u_k^\top & x_{k+1}^\top & u_{k+1}^\top & x_{k+2}^\top & \dots & x_{k+N_2}^\top \end{bmatrix}^\top$, it can be written as

$$\begin{aligned} &\min_{Z_k} \frac{1}{2} Z_k^\top H_{sp} Z_k \\ &\text{subject to} \\ &\quad DZ_k = d \\ &\quad G_x X_k \leq h_x \\ &\quad G_u U_k \leq h_u \end{aligned}$$

where the Hessian, H_{sp} is sparse, and indeed block diagonal and the equality constraints are highly structured. Solving the equality constrained problem with this formulation and exploiting the sparsity and structure of the linear algebra can lead to highly efficient implementations. However, inequality constraints could partially break that structure and can lead to complications; furthermore, most off-the-shelf solvers do not exploit such structure. For these reasons, an alternative *dense* formulation, reviewed below, is preferred here. The condensing approach used for LTI systems is straightforward: substitute the prediction equation (2.7) (i.e. the equality constraint) into the cost function (2.8) and on the state (inequality) constraints and rearrange to obtain

$$\begin{aligned} &\min_{U_k} \frac{1}{2} U_k^\top H U_k + g^\top U_k \\ &\text{subject to} \\ &\quad \begin{bmatrix} G_u \\ G_x \Gamma \end{bmatrix} U_k \leq \begin{bmatrix} h_u \\ h_x - G_x \Phi x_k \end{bmatrix}, \end{aligned} \tag{2.9}$$

where

$$\begin{aligned} H &= 2(R + \Gamma^\top \hat{Q} \Gamma) \\ g^\top &= 2\Gamma^\top \hat{Q} \Phi x_k. \end{aligned} \tag{2.10}$$

This formulation has far fewer variables (Nm rather than $N(n+m)$) and fewer constraints, at the price of destroying the structure of the problem. It is however (particularly for relatively short horizons) preferred and leads to an efficient implementation of the QP, compatible with all off-the-shelf solvers.

Integral action and offset-free control

Adding integral action to the controller is desired, as it provides what in the MPC literature is often referred to as *offset-free* control, that is, tracking of a desired reference with no steady-state error in the presence of inevitable disturbances or model uncertainty. There are two alternative ways to achieve this goal in the context of MPC: state augmentation with an integrating disturbance,

requiring a disturbance model and an estimator, and using incremental/velocity form MPC. Although recently it has been established that the latter is a special case of the former for a particular choice of disturbance model and estimator gain [92], here they are still treated as alternatives, given the special properties of the velocity algorithms when applied to nonlinear systems (Section 4.2) and because of its simple implementation.

Disturbance models. The state vector of the plant model is augmented with an integrating disturbance driven by a zero-mean, stationary, white noise process, i.e. $d_{k+1} = d_k + w_k$, resulting in the augmented model

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ d_{k+1} \end{bmatrix} &= \begin{bmatrix} A & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} 0 \\ I \end{bmatrix} w_k \\ y_k &= \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} + v_k, \end{aligned} \tag{2.11}$$

where B_d, C_d dictate how the disturbance enters the model. These can be chosen based on physical insight, e.g. if one expects input disturbances one can choose $B_d = B$, or can be used as tuning parameters, however this choice is not free and detectability conditions have to be met; for a more detailed discussion the reader is referred to [98]. An observer is used to estimate the disturbance (and possibly the state, in the output feedback case) using the augmented model, providing integral action by integrating the model error (through the estimation of d); this way, both disturbances and model uncertainty can be compensated.

Target selector. Along with a disturbance estimator, a target selector is often used in tracking problems. The goal of a target selector is to calculate admissible steady state values for the input and the state vector, taking into account constraints and the effect of disturbances. For the nominal, unconstrained case, steady state values of the state and inputs can be computed by solving

$$\begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}$$

if it has a solution, where r is a reference set point. However, in order to account for the effect of disturbances and constraints, in general an optimization problem needs to be solved instead:

$$\begin{aligned} &\min_{x_s, u_s} \frac{1}{2} ||Cx_s + C_d\hat{d} - r||^2_{Q_s} + ||u_s||^2_{R_s} \\ &\text{subject to} \\ &\begin{bmatrix} I - A & -B \\ C & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B_d\hat{d} \\ r - C_d\hat{d} \end{bmatrix} \\ &\tilde{G}_x x_s \leq \tilde{h}_x \\ &\tilde{G}_u u_s \leq \tilde{h}_u, \end{aligned} \tag{2.12}$$

where $Q_s \in \mathbb{R}^{l \times l}$, $R_s \in \mathbb{R}^{m \times m}$ are tuning parameters and $\hat{d}$ is the estimate of the disturbance, coming from the observer. The solution to this optimization problem yields steady state target

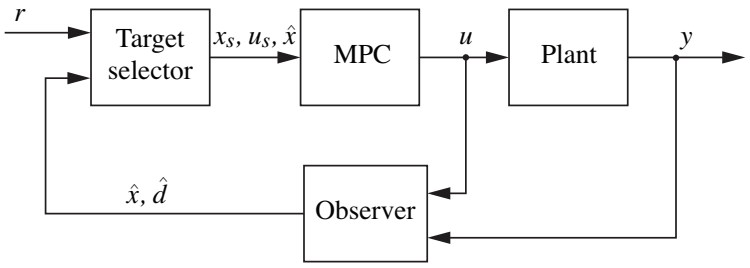


Figure 2.3: Typical MPC loop for offset-free control [98]

values x_s, u_s which fulfill constraints and take into account the effect of disturbances. A typical MPC loop under these circumstances is shown in Figure 2.3. Note that, as the target selector problem uses the current disturbance estimate, this problem is to be solved at every time step and not only when a change in reference takes place.

Velocity form Assuming again a disturbance driven by a zero-mean, white noise process, one can cancel the effect of the additive disturbance on the state equation by using an incremental model[2], i.e. $\Delta x_{k+1} = A\Delta x_k + B\Delta u_k$, where Δ is the backward difference operator. In order to have a meaningful tracking problem, the (now incremental) state is then augmented with the output to yield

$$\begin{bmatrix} y_{k+1} \\ \Delta x_{k+1} \end{bmatrix} = \begin{bmatrix} I & CA \\ 0 & A \end{bmatrix} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix} + \begin{bmatrix} CB \\ B \end{bmatrix} \Delta u_k \tag{2.13}$$

$$y_k = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix}. \tag{2.14}$$

In this case, no disturbance estimation is needed and if full state information is available, neither is an observer. However this is in general not the case and an observer is often needed, both for estimation and for filtering as using incremental models can have an adverse effect with noisy measurements. A target selector, on the other hand is not needed, as all equilibria (steady-states) are mapped to the origin of the incremental model. Further discussion about velocity form MPC is presented in Chapter 4.

2.3.2 Input-Output Models

Input-output (IO) LTI discrete-time models are often given as a transfer function in the complex variable z, nevertheless, to be consistent throughout this work where input-output models of time- and parameter-varying models are treated, a matrix polynomial notation on the backward time-shift operator q^{-1} is adopted. Consider the IO model

$$y_k = \sum_{i=1}^{n_{dy}} A_i q^{-i} y_k + \sum_{j=1}^{n_{du}} B_j q^{-j} u_k \tag{2.15}$$

[2]This is a consequence of the fact that the best prediction for a disturbance driven by the described noise process is $\hat{d}_{k+1} = d_k$

where $y \in \mathbb{R}^l$ is the output and $u \in \mathbb{R}^m$ is the input of the system. The assumption that the input does not have an immediate effect on the output (i.e. no feedthrough) is made in this case as well. Note that this is a straightforward MIMO extension of the SISO case, in which polynomials $a(q)$, $b(q)$ are used, in this case, however, these are matrix polynomials $A(q) = A_1 q^{-1} + A_2 q^{-2} + ... + A_{n_{dy}} q^{n_{dy}}$, and similarly for $B(q)$.

In order to write the prediction equation in an IO framework, the definition of vectors of both future and past values of the input and output is required:

$$Y_k = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+N_2} \end{bmatrix}, \qquad U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N_u-1} \end{bmatrix}, \qquad Y_k^p = \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-n_{dy}} \end{bmatrix}, \qquad U_k^p = \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-n_{du}} \end{bmatrix}.$$

shifting (2.15) forward in time leads to the prediction equation

$$\mathcal{A}_f Y_k + \mathcal{A}_p Y_k^p = \mathcal{B}_p U_k^p + \mathcal{B}_f U_k \tag{2.16}$$

where

$$\mathcal{A}_f = \begin{bmatrix} I & 0 & 0 & \dots \\ A_1 & I & 0 & \dots \\ A_2 & A_1 & I & \dots \\ \vdots & & \ddots & \dots \\ A_{n_{dy}} & A_{n_{dy}-1} & \dots & \\ 0 & A_{n_{dy}} & \dots & \\ \vdots & & \ddots & \end{bmatrix} \in \mathbb{R}^{N_2 l \times N_2 l} \qquad \mathcal{A}_p = \begin{bmatrix} A_1 & A_2 & \dots & A_{n_{dy}} \\ A_2 & A_3 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ A_{n_{dy}} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \end{bmatrix} \in \mathbb{R}^{N_2 l \times n_{dy} l}$$

$$\mathcal{B}_f = \begin{bmatrix} B_1 & 0 & 0 & \dots \\ B_2 & B_1 & 0 & \dots \\ B_3 & B_2 & B_1 & \dots \\ \vdots & \ddots & \dots & \\ B_{n_{du}} & B_{n_{du}-1} & \dots & \\ 0 & B_{n_{du}} & \dots & \\ \vdots & & \ddots & \end{bmatrix} \in \mathbb{R}^{N_2 l \times N_u m} \qquad \mathcal{B}_p = \begin{bmatrix} B_2 & B_3 & \dots & B_{n_{du}} \\ B_3 & B_4 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ B_{n_{du}} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \end{bmatrix} \in \mathbb{R}^{N_2 l \times n_{du} m}$$

Input/Output optimization problem

As there is no state vector, the optimization problem is slightly modified to penalize the output error directly, and state constraints are replaced by output constraints, leading to the optimization problem

$$\min_{Y_k, U_k} ||Y_k - \hat{r}_k||^2_{\hat{Q}} + ||U_k - U_s||^2_{\hat{R}}$$

subject to

$$\begin{aligned} &\mathcal{A}_f Y_k + \mathcal{A}_p Y_k^p = \mathcal{B}_p U_k^p + \mathcal{B}_f U_k \\ &G_u U_k \leq h_u \\ &G_y Y_k \leq h_y \end{aligned}$$

where $\hat{r}_k$ is the vector containing future reference values. Similar to the state space case, the optimization problem can be equivalently expressed in a dense form by substituting the equality constraint onto the cost function. To this end, define the matrices

$$\begin{aligned} \Phi &= -\mathcal{A}_f^{-1}\mathcal{A}_p \\ \Phi_u &= \mathcal{A}_f^{-1}\mathcal{B}_p \\ \Gamma &= \mathcal{A}_f^{-1}\mathcal{B}_f \end{aligned}$$

and write the condensed optimization problem as

$$\min_{U_k} \frac{1}{2} U_k^\top H U_k + g^\top U_k$$

subject to

$$\begin{bmatrix} G_u \\ G_y\Gamma \end{bmatrix} U_k \leq \begin{bmatrix} h_u \\ h_x - G_y(\Phi Y_k^p + \Phi_u U_k^p) \end{bmatrix}$$

where

$$\begin{aligned} H &= 2(\hat{R} + \Gamma^\top \hat{Q} \Gamma) \\ g^\top &= 2\Gamma^\top \hat{Q}(\Phi Y_k^p + \Phi_u U_k^p - \hat{r}_k) \end{aligned} \tag{2.17}$$

Integral action

The common practice to include integral action in input-output MPC is by means of incremental models, similar to the one presented in the state space framework. Indeed a disturbance can be included by integrating a zero-mean, stationary, white noise process v, i.e.

$$\sum_{i=0}^{n_{dy}} -A_i q^{-i} y_k = \sum_{j=1}^{n_{du}} B_j q^{-j} u_k + \sum_{l=1}^{n_{du}} T_l q^{-l} \frac{v_k}{\Delta}$$

where $\sum_{l=1}^{n_{du}} T_l q^{-l}$ dictates how the disturbance affects the output (disturbance model). Note that in the nominal case (i.e. if $v_k \equiv 0$) this equation is equivalent to (2.15) by setting $A_0 = -I$. The equation can then be multiplied by Δ to yield

$$\sum_{i=0}^{n_{dy}} -A_i q^{-i} \Delta y_k = \sum_{j=1}^{n_{du}} B_j q^{-j} \Delta u_k + \cancel{\sum_{l=1}^{n_{du}} T_l q^{-l} v_k}$$

where the cancellation follows, as before, given the characteristics of the noise process. Using output increments in an input-output framework is not meaningful, as one normally tracks a reference, and information on the absolute output y_k is needed. For this reason, practitioners often multiply $\Delta = 1 - q^{-1}$ by the $A(q)$ polynomial, leading to $\tilde{A}(q) = (1 - q^{-1})A(q)$ [101], and

$$y_k = \sum_{i=1}^{n_{dy}+1} \tilde{A}_i q^{-i} y_k + \sum_{j=1}^{n_{du}} B_j q^{-j} \Delta u_k. \tag{2.18}$$

The prediction model (2.18) yields unbiased predictions and hence integral action, with the further advantage that no observer is necessary as it is in most cases when using state space representations.

2.3.3 Stability

In this section, a brief overview of the most often encountered stability analysis in the context of MPC is given. This is mostly done in a state space framework and for now the discussion is limited to this case; furthermore, for ease of exposition, only the regulator case is discussed; as such the objective is to stabilize the origin of the state space.

Given that stability is an infinite horizon concept, the general idea is to emulate an infinite horizon, in a finite horizon setting. To this end, the prediction horizon is set to $N_y = \infty$ yielding the cost function

$$J_k = \sum_{i=0}^{N-1} \ell(x_{k+i}, u_{k+i}) + \sum_{i=N}^{\infty} \ell(x_{k+i}, u_{k+i}).$$

where for notational simplicity the control horizon is denoted by N. The most straightforward way to make this cost function equivalent to the finite-horizon cost function (the first term on the right-hand side of the equation) is to enforce $\sum_{i=N}^{\infty} \ell(x_{k+i}, u_{k+i}) = 0$, this is achieved by imposing a *terminal equality constraint* $x_{k+N} = 0$ [66]. Even though feasibility of this problem at $k = 0$ guarantees stability and recursive feasibility[3], the resulting trajectories are suboptimal with respect to the infinite horizon cost function. This can be concluded from the fact that the closed-loop behavior does not correspond to the open-loop prediction as seen in the following example

[3]Recursive feasibility is ensured because a feasible u_k is always given by shifting u^*_{k-1}.

Example 2.1 (Terminal equality constraints)**.** Given the following LTI state space model [4]

$$x_{k+1} = \begin{bmatrix} 4/3 & -2/3 \\ 1 & 0 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_k$$
$$y_k = \begin{bmatrix} -2/3 & 1 \end{bmatrix}$$

an MPC law is used with $Q = C^\top C$, $R = 100$, $N = 5$ and a terminal constraint $x_{k+N} = 0$. The resulting optimal trajectories for $k \in [0\ \ 2]$ are shown in Figure 2.4

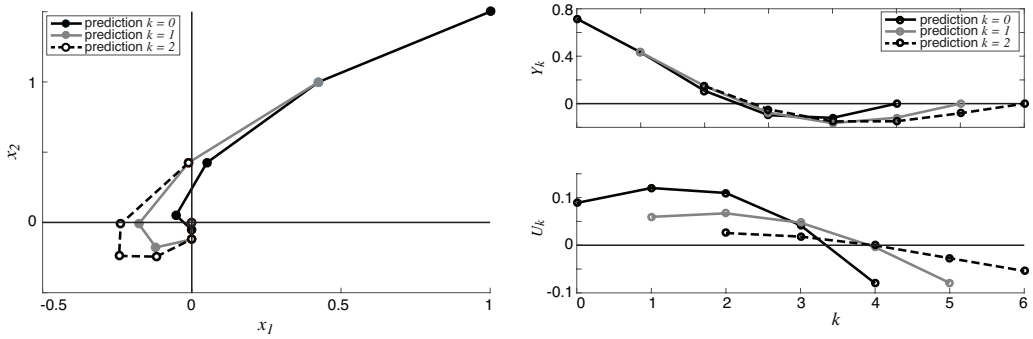


Figure 2.4: Open-loop finite-horizon predictions.

The behavior seen in example 2.4 can be explained by the fact that, as the horizon recedes away, so does the constraint $x = 0$, so the optimal choice is to use less control effort to make J_k smaller but still be able to satisfy the constraint (a heavy penalty is given on the control effort for this effect to be more noticeable). This is also the reason why the response is not optimal in the infinite horizon sense.

Dual Mode Control. Suboptimality of the open-loop prediction is not the only disadvantage of using a terminal equality constraint. Indeed a major caveat is feasibility, as the controller needs to be able to drive the system to the origin in N steps while satisfying constraints. This could potentially require a long horizon and translate in increased online computational complexity. To alleviate this problem, Michalska and Mayne [85] proposed to use a concept deemed *dual mode control*. Conceptually, the idea behind it is to let the MPC law drive the state into a region surrounding the origin within N steps, once this region is reached, control authority is switched to a fictitious state-feedback law, which then drives the state to the origin. That is,

$$u_{k+i} = Fx_{k+i} \qquad i = N, N+1, ..., \infty$$

resulting in the infinite horizon cost

$$J_k = \sum_{i=0}^{N-1} ||x_{k+i}||_Q^2 + ||u_{k+i}||_R^2 + \sum_{i=N}^{\infty} ||x_{k+i}||_Q^2 + ||Fx_{k+i}||_R^2,$$

focusing on the *tail* (the second sum on the right-hand side) and defining the closed-loop state transition matrix $\Lambda = A + BF$, note that $x_{k+N+j} = \Lambda^j x_{k+N}$, then

[4]This model is taken from [98] where it was used to highlight that using MPC with a short horizon can make an open-loop stable system, closed-loop unstable. In this case, this is a consequence of the fact that the plant has an unstable zero, which the controller tries to invert if a short-horizon is used.

$$\begin{aligned}\sum_{i=N}^{\infty} ||x_{k+i}||_Q^2 + ||Fx_{k+i}||_R^2 &= \sum_{i=N}^{\infty} x_{k+i}^\top (Q + F^\top RF) x_{k+i} \\ &= x_{k+N}^\top \left(\sum_{i=N}^{\infty} (\Lambda^{i-N})^\top (Q + F^\top RF) \Lambda^{i-N} \right) x_{k+N} = x_{k+N}^\top P x_{k+N}\end{aligned} \tag{2.19}$$

so that the infinite horizon cost function becomes

$$J_k = \sum_{i=0}^{N-1} \left(||x_{k+i}||_Q^2 + ||u_{k+i}||_R^2 \right) + ||x_{k+N}||_P^2$$

Note that this takes the form of (2.3) for $N_2 = N_u$, where the tail of the infinite horizon is captured by the terminal cost.

There are two implicit assumptions made in the derivation above. The first, rather reasonable is that the state feedback gain F is stabilizing, i.e. $|\text{eig}(\Lambda)| < 1$ so that the sum in (2.19) converges. The second, which contrary to the first is non-trivial, is that $u_{k+i} = Fx_{k+i} \in \mathcal{U}$, $i = N, ..., \infty$. For a fixed state feedback gain F this imposes a restriction on the state x_{k+i} $i = N, ..., \infty$, namely that it must lie within a set $\mathbb{X}$ in which the condition $Fx \in \mathcal{U}$ is met, this can also be stated as $x_{k+N} \in \mathbb{X}$ and $\mathbb{X}$ is a control invariant set. The following theorem formalizes the discussion above.

Theorem 2.2 (Stabilizing MPC for LTI systems [77])**.** Assume there exists state feedback law $F(x) = Fx$, a terminal constraint set $\mathbb{X}$ and terminal cost function $\Psi(x)$ such that

1. $0 \in \mathbb{X}$
2. $(A + BF)x \in \mathbb{X}, \forall x \in \mathbb{X}$
3. $\Psi(0) = 0, \Psi(x) > 0 \ \ \forall x \neq 0$
4. $\Psi((A + BF)x) - \Psi(x) \leq x^\top Qx - x^\top F^\top RFx, \forall x \in \mathbb{X}$
5. $Fx \in \mathcal{U}, \forall x \in \mathbb{X}$

hold. Then, assuming feasibility of the initial state, an MPC controller solving the optimization problem

$$\begin{aligned}&\min_{U_k} J_k \\ &\text{subject to} \\ &\quad u_{k+i} \in \mathcal{U} \quad i = 0, ..., N-1 \\ &\quad x_{k+N} \in \mathbb{X}\end{aligned}$$

guarantees asymptotic stability.

The dual-mode concept is illustrated in Figure 2.5. Note that the second mode controller (the fixed state-feedback) is only a fictitious controller that is not applied even when inside the

terminal region, establishing its existence however, is important to have guarantees of stability and recursive feasibility.

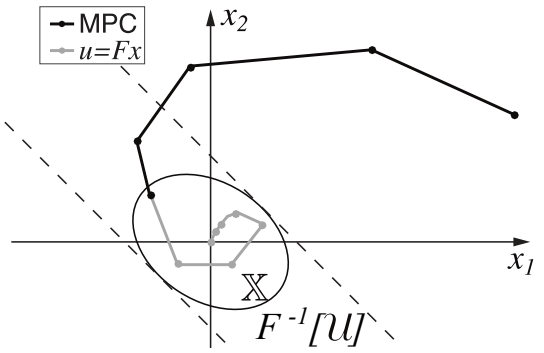


Figure 2.5: Dual mode control[5]

2.4 Linear Parameter Varying Modelling

Linear Parameter Varying systems, as the name suggest are linear systems whose matrices depend on time-varying but assumed measurable parameters. As such, the concept of LPV systems encompasses Linear Time Invariant (LTI) and Linear Time-Varying (LTV) systems as special cases. The interest in LPV systems started growing in the early 1990's in the context of gain-scheduled control ([110]) for systems with exogenous scheduling parameters, aiming to systematically provide stability and performance guarantees which were absent in classical gain-scheduling approaches. The framework quickly became popular and became a natural segue of linear $\mathcal{H}_\infty$ design techniques to nonlinear systems, thereby enabling powerful linear design techniques to be applied to complex nonlinear systems [11] by allowing the scheduling parameters to be endogenous. In what follows only the relevant aspects of LPV modelling are addressed leaving much of the stability derivations out, as they do not play a role in the context of this thesis.

Definition 2.14 (LPV system)**.** A Linear Parameter Varying (LPV) system is a dynamic system of the form

$$\begin{aligned} \dot{x} &= A(\rho(t))x + B(\rho(t))u \\ y &= C(\rho(t))x + D(\rho(t))u \end{aligned} \tag{2.20}$$

where $x \in \mathbb{R}^n$ is the state of the system, $y \in \mathbb{R}^l$ is the output, $u \in \mathbb{R}^m$ is the input and $\rho \in \mathbb{R}^{n_\rho}$ is an unknown but measurable time-varying parameter. The parameter vector ρ belongs to a compact set of admissible values, i.e.

$$\rho(t) \in \mathcal{P} \qquad \forall t \geq 0.$$

[5]Note that F needs not be invertible, $F^{-1}[\mathcal{U}]$ is the preimage of the set $\mathcal{U}$ under F, see Notation section.

The matrix-valued maps $A : \mathbb{R}^{n_\rho} \to \mathbb{R}^{n\times n}$, $B : \mathbb{R}^{n_\rho} \to \mathbb{R}^{n\times m}$, $C : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times n}$, $D : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times m}$ are continuous functions of ρ on $\mathcal{P}$.

Note that the fact that ρ is confined to a compact set, together with continuity of $(A(\cdot), B(\cdot), C(\cdot), D(\cdot))$ implies that these maps are bounded on $\mathcal{P}$. It is often assumed, in addition, that the rate of variation of the parameter also belongs to a compact set $\mathcal{V}$:

Definition 2.15 (Rate-bounded LPV system)**.** An LPV system with bounded parameter rates is a system as defined in Definition 2.14, which in addition fulfills

$$\dot{\rho}(t) \in \mathcal{V}, \quad \forall t \geq 0$$

where $\mathcal{V}$ is a compact set.

Given these definitions, the sets of admissible parameter *trajectories* are described by ([128])

$$\mathcal{F}_{\mathcal{P}} = \left\{\rho(t) \in C^1(\mathbb{R}_{\geq 0}, \mathbb{R}^{n_\rho}) : \rho(t) \in \mathcal{P}\ \forall t \geq 0\right\}$$

and

$$\mathcal{F}_{\mathcal{P}}^{\mathcal{V}} = \left\{\rho(t) \in C^1(\mathbb{R}_{\geq 0}, \mathbb{R}^{n_\rho}) : \rho(t) \in \mathcal{P}, \dot{\rho}(t) \in \mathcal{V}\ \forall t \geq 0\right\}$$

respectively, where $C^1(\mathbb{R}_{\geq 0}, \mathbb{R}^{n_\rho})$ is the set of continuously differentiable functions mapping $\mathbb{R}_{\geq 0}$ to $\mathbb{R}^{n_\rho}$.

Quasi-LPV systems

An important special class of LPV system, indeed the one that is the focus of this thesis, is the so-called quasi-LPV system. A quasi-LPV system arises when the scheduling parameter is a function of endogenous signals. To make a distinction, LPV systems with exogenous parameters are referred to as *general* or *pure* LPV systems.

Definition 2.16 (quasi-LPV system)**.** A quasi-LPV system is an LPV system as defined in Definitions 2.14 or 2.15 for which the time varying parameters are functions of endogenous signals, i.e.

$$\rho(t) = \varrho(x(t), u(t))$$

where $\varrho : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_\rho}$ is a continuous function on $\mathcal{P}$.

Remark 2.2. *In the quasi-LPV case, the fact that $\rho \in \mathcal{P}$ implicitly defines a subset of the state space X such that $x \in X \implies \rho \in \mathcal{P}$ ($X \subseteq \varrho^{-1}[\mathcal{P}]$). This entails that any stability result obtained using a quasi-LPV approach is local in nature, as no statement regarding stability can be made for $\rho \notin \mathcal{P} \implies x \notin X$.*

The attentive reader may notice that, in essence, a quasi-LPV system is nothing more than a nonlinear system expressed as an LPV system by a suitable parameterization (often called an *embedding*) implied by the function ϱ. It is therefore not entirely correct to speak of quasi-LPV

systems, since they do not arise naturally (i.e. no real system is quasi-LPV), as opposed to the underlying nonlinear system or even pure LPV systems, the quasi-LPV nature comes when modelling and the correct nomenclature would therefore be *quasi-LPV model of the nonlinear system*. To avoid this cumbersome word play, throughout this work, when referring to quasi-LPV *systems*, it is meant as the quasi-LPV model arising from suitable parameterization of a nonlinear system.

Obtaining a quasi-LPV model from a nonlinear system is a nontrivial issue that deserves attention, especially considering that quasi-LPV parameterizations are non-unique. The most prominent methods are the so-called ad-hoc and Jacobian linearization-based parameterizations [102]. The former attempts to hide nonlinearities as parameters by clever substitutions (see Example 2.2 below), needless to say this highly depends on the designer's experience and on the system itself (see Example 2.3 below for a counter-example). The latter is somewhat self-explanatory, a Jacobian linearization is performed, and the resulting state-dependent system matrices are turned into parameter-dependent maps by defining $\rho = H \begin{bmatrix} x^\top & u^\top \end{bmatrix}^\top$ where H is a selection matrix. Each method has advantages and disadvantages, the most obvious is exactness of the former while being more difficult to obtain than the latter. To illustrate this, consider the following example.

Example 2.2 (Ad-hoc qLPV parameterization)**.** Assume a nonlinear model of a single pendulum of length L without friction, and the operating region $\theta \in [-\pi \;\; \pi]$

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta).$$

An ad-hoc qLPV parameterization can be obtained by substituting $\operatorname{sinc}(\theta) = \frac{\sin(\theta)}{\theta}$ and defining $\rho = \theta$, yielding

$$\Sigma_1 \begin{cases} \ddot{\theta} = -\dfrac{g}{L}\operatorname{sinc}(\rho)\theta \quad \rho \in [-\pi \;\; \pi]. \end{cases}$$

Alternatively, in this case a simpler affine parameter-dependent model can be obtained by defining $\rho = \operatorname{sinc}(\theta)$ resulting in the model

$$\Sigma_2 \begin{cases} \ddot{\theta} = -\dfrac{g}{L}\rho\theta \quad \rho \in [-0.218 \;\; 1]. \end{cases}$$

This simple yet realistic example highlights the non-uniqueness of qLPV parameterization and the benefits of ad-hoc parameterization as both representations are simple and exact, so that Jacobian linearization-based parameterization is not worth discussing. In the following example however, the downside of ad-hoc parameterization is clearly seen.

Example 2.3 (Ad-hoc vs. Jacobian linearization qLPV parameterization)**.** Consider a kinematic model of an actuated disk with mass $m = 1$, inertia $I = 1$, inputs v, ω and operating region $x, y \in \mathbb{R}$, $\theta \in [-\pi \;\; \pi]$, $v \in [-1 \;\; 1]$, $\omega \in [-1 \;\; 1]$

$$\dot{x} = \cos(\theta)v \quad \dot{y} = \sin(\theta)v \quad \dot{\theta} = \omega$$

a naive ad-hoc parameterization can be carried out by defining $\rho = \theta$

$$\Sigma_1 \begin{cases} \dot{x} = \cos(\rho)v \\ \dot{y} = \sin(\rho)v \qquad \rho \in [-\pi \;\; \pi] \\ \dot{\theta} = \omega \end{cases}$$

however, the resulting qLPV model is not useful for control, since the model lost all coupling information between the states x, y and θ; furthermore this model is not stabilizable for the whole operating region[a]. A Jacobian linearization, on the other hand would result in $\rho_1 = \theta$, $\rho_2 = v$

$$\Sigma_2 \begin{cases} \dot{\tilde{x}} \approx \cos(\rho_1)\tilde{v} - \sin(\rho_1)\rho_2\tilde{\theta} \qquad \rho_1 \in [-\pi \;\; \pi] \\ \dot{\tilde{y}} \approx \sin(\rho_1)\tilde{v} + \cos(\rho_1)\rho_2\tilde{\theta} \qquad \rho_2 \in [-1 \;\; 1] \\ \dot{\tilde{\theta}} \approx \tilde{\omega} \end{cases}$$

which, at the price of being an approximate model and having more parameters, it is far more meaningful in the context of control. Note that this model also becomes uncontrollable for $(\rho_1, \rho_2) = (0, 0)$, $(\rho_1, \rho_2) = (\pi, 0)$ however this comes from the underlying system, and not because of artifacts introduced by parameterization, as in the previous case.

[a] this is an important assumption usually made in order to use Lyapunov arguments to establish stability and synthesize controllers

Beside the two presented parameterization methods, others exist which may be better suited for certain applications. In Chapter 4 a different method is presented based on velocity-based linearization [73], this approach offers the advantage that it is exact as ad-hoc parameterization but as easy to obtain and coupling-preserving as the Jacobian linearization method.

2.4.1 Discrete-time LPV systems

Even though most of the design and synthesis results in the LPV literature focus on the continuous-time setting, it is of utmost practical interest to consider a discrete-time setting as well. This is perhaps even more important in the present context as predictive control laws are considerably more easily handled in discrete-time. With this in mind, the definitions above can be appropriately modified to accommodate the effect of sampling.

Definition 2.17 (Discrete-time LPV system)**.** A discrete-time Linear Parameter Varying (LPV) system is a dynamic system of the form

$$\begin{aligned} x_{k+1} &= A(\rho_k)x_k + B(\rho_k)u_k \\ y_k &= C(\rho_k)x_k + D(\rho_k)u_k \end{aligned} \tag{2.21}$$

where $\rho \in \mathbb{R}^{n_\rho}$ is an unknown but measurable time-varying parameter. The parameter vector ρ belongs to a compact set of admissible values, i.e.

$$\rho_k \in \mathcal{P} \qquad \forall k \geq 0.$$

The maps $A : \mathbb{R}^{n_\rho} \to \mathbb{R}^{n\times n}$, $B : \mathbb{R}^{n_\rho} \to \mathbb{R}^{n\times m}$, $C : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times n}$, $D : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times m}$ are continuous functions of ρ on $\mathcal{P}$.

In addition, the parameter *difference* can also be bounded, so as to belong to a compact set $\mathcal{V}$

$$\Delta\rho_k \in \mathcal{V}, \quad \forall k \geq 0.$$

The sets of admissible trajectories $\mathcal{F}_\mathcal{P}$, $\mathcal{F}_\mathcal{P}^\mathcal{V}$ can be appropriately redefined with the difference that $\rho_k \in C^1(\mathbb{Z}_{\geq 0}, \mathbb{R}^{n_\rho})$. Finally discrete-time quasi-LPV systems can be defined as

Definition 2.18 (quasi-LPV system)**.** A quasi-LPV system is an LPV system as defined in Definition 2.17 for which the time varying parameters are functions of endogenous signals, i.e.

$$\rho_k = \varrho(x_k, u_k).$$

where $\varrho : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_\rho}$ is a continuous function on $\mathcal{P}$.

Input-Output LPV models

As in the LTI case, an equivalent representation of an LPV system in the form (2.21) can be obtained which only depends on time-shifted inputs and outputs. Although such a representation arises more naturally when performing system identification using the prediction error method [14], first principle models can also be derived directly e.g. by appropriate discretization of the nonlinear differential equations.

Definition 2.19 (Input-Output LPV model)**.** A Input-Output Linear Parameter Varying (LPV) model is a model of the form

$$y(k) = \sum_{i=1}^{n_{d_y}} A_i(\rho(k))q^{-i}y(k) + \sum_{j=1}^{n_{d_u}} B_j(\rho(k))q^{-j}u(k) \tag{2.22}$$

where $y \in \mathbb{R}^l$ is the output, $u \in \mathbb{R}^m$ is the input and $\rho \in \mathbb{R}^{n_\rho}$ is an unknown but measurable time-varying parameter. The parameter vector ρ belongs to a compact set of admissible values, i.e.

$$\rho_k \in \mathcal{P} \qquad \forall k \geq 0.$$

The maps $A_i : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times l}$ $i = 1, ..., n_{d_y}$, $B_j : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times m}$ $j = 1, ..., n_{d_u}$ are continuous functions of ρ on $\mathcal{P}$.

In addition, the parameter *difference* can also be bounded, so as to belong to a compact set $\mathcal{V}$

$$\Delta\rho_k \in \mathcal{V}, \quad \forall k \geq 0.$$

Correspondingly, in the quasi-LPV case, the parameters are now functions of inputs and outputs.

Definition 2.20 (quasi-LPV IO model)**.** An IO quasi-LPV model is an LPV model as defined in Definition 2.19 for which the time varying parameters are functions of endogenous signals, i.e.

$$\rho_k = \varrho(y_k, u_k).$$

where $\varrho : \mathbb{R}^l \times \mathbb{R}^m \to \mathbb{R}^{n_\rho}$ is a continuous function on $\mathcal{P}$.

A more detailed discussion of IO qLPV representations is given in Chapter 5 in the context of predictive control for IO quasi-LPV systems.

Dynamic Parameter Dependence

An issue with discrete-time LPV models which is excluded from the definitions above, is the possibility to exhibit dependence on time-shifted parameter values. This is an undesired effect often stemming from conversions between IO and state space (SS) models.

Definition 2.21 (Dynamic Parameter Dependence)**.** An LPV model is said to exhibit dynamic parameter dependence if the model matrices are function of current, as well as time-shifted, parameter values, i.e. if $(A(\mathsf{P}_k), B(\mathsf{P}_k), C(\mathsf{P}_k), D(\mathsf{P}_k))$ in the SS case and $A_i(\mathsf{P}_k), B_j(\mathsf{P}_k), i = 1, ..., n_{d_y}, j = 1, ..., n_{d_u}$ in the IO case where

$$\mathsf{P}(k) = \left\{\rho(k) \;\; \rho(k + n_1^+) \;\; ... \;\; \rho(k - n_1^-) \;\; ...\right\}$$

where $n_i^+, n_i^- \in \mathbb{Z}_{\geq 0}$, if in the other hand, the matrices are functions of only current parameter values ρ_k, the model is said to have static parameter dependence.

The definition for the LPV models above only consider the static parameter dependence case, this although less general, represent the models that are of interest in the context of this thesis, unless otherwise stated.

Chapter 3

MPC for quasi-LPV Systems: State Space Framework

This chapter presents the framework on which the rest of this work is based. The basic premise is to use quasi-LPV modelling as a tool to embed nonlinear system dynamics in a linear model; this allows the optimization problem which arises in MPC to be efficiently solved. Furthermore, this mature modelling framework enables the use of several available results that prove useful when establishing stability conditions; as LPV systems naturally extend many non-predictive LTI stability results to nonlinear systems, so can similar procedures be applied in the context of MPC. Whereas for gain-scheduling control it is irrelevant whether the scheduling parameters are exogenous or endogenous, in the context of MPC the former case is such that evolution of the scheduling variables is unknown and can thus not be predicted; for this reason, MPC for LPV systems faces the problem that the prediction model is uncertain. In the case of quasi-LPV systems, the scheduling variable is determined by the state and/or input and can therefore be predicted, although the resulting prediction equations are nonlinear. To carry out this prediction efficiently, an iterative scheme is proposed which is obtained by freezing the scheduling trajectory to the one obtained in the previous time-step, making the prediction linear time-varying, thus enabling efficient optimization.

Stability analysis is based on the concept of *dual mode control* [85], the infinite horizon prediction is made assuming that a region in state space containing the origin is reached within a finite horizon, control authority is then switched to fixed state feedback control law which, within this region, generates feasible control trajectories to drive the system to the origin. The novelty of the proposed approach lies in extending analysis to parameter dependent terminal region and controller, thus making them less conservative and comparatively easier to compute than those obtained by other methods.

This chapter is organized as follows: Section 3.1 gives the problem setup and presents the parameter dependent predictions along with an iterative algorithm which can be used to solve the resulting online optimization problem efficiently. The stability result is given in Section 3.2 where stability conditions in the form of LMIs, to be solved offline, are derived. Section 3.3 discusses the extension to the tracking problem and how offset-free tracking can be achieved, while Section 3.4 presents how to consider nonlinear constraints in this framework. Finally, Section 3.5 validates the control scheme experimentally on an Arm-Driven Inverted Pendulum and Section 3.6 summarizes the chapter.

3.1 Parameter Dependent Predictive Control

Throughout this chapter, a quasi-LPV discrete-time state space model is considered

$$x_{k+1} = A(\rho_k)\, x_k + B(\rho_k)\, u_k \tag{3.1}$$

according to Definition 2.18, where $\rho_k = \varrho(x_k, u_k)$. An additional assumption is made, namely that $(A(\rho), B(\rho))$ is stabilizable $\forall \rho \in \mathcal{P}$, this is a necessary condition for the existence of a solution to the LMI problems presented below. The predictive control law is computed by minimizing a finite-horizon objective function[1]

$$J_k = \sum_{i=0}^{N-1} \ell(x_{k+i}, u_{k+i}) + \Psi(x_{k+N})$$

in a receding horizon fashion, where the terminal cost function $\Psi(x) > 0 \; \forall \; x \neq 0, \; \Psi(0) = 0$ and the stage cost is defined as

$$\ell(x, u) = x^\top Q x + u^\top R u. \tag{3.2}$$

Correspondingly, the terminal cost function is usually also chosen quadratic, as $\Psi(x) = x^\top P x$, with $P > 0$.

The optimization problem to be solved is thus

$$\begin{aligned} &\min_{\substack{u_k,\ldots,u_{k+N-1} \\ x_{k+1},\ldots,x_{k+N}}} J_k \\ &\text{s.t.} \\ &\quad x_{k+i+1} = A(\rho_{k+i})x_{k+i} + B(\rho_{k+i})u_{k+i} \\ &\quad x_{k+i} \in \mathcal{X} \\ &\quad u_{k+i} \in \mathcal{U} \qquad \forall i \in [0 \quad N-1] \\ &\quad x_{k+N} \in \mathbb{X} \end{aligned} \tag{P.1}$$

where $\mathbb{X} \subset \mathbb{R}^n$ is a so-called terminal constraint set to be defined. For the time being it is assumed that $\mathcal{X} = \mathbb{R}^n$, in other words, the admissible state set is the whole state space. How to handle nonlinear state constraints in this framework is presented in Section 3.4, for now it is assumed that the optimization problem only includes input constraints and a terminal state constraint. The equality constraint in problem (P.1) assumes that the future parameter trajectory is available, this would make the problem not solvable for a general LPV system as the scheduling parameter is not assumed to be known a priori but only measurable online. In contrast, for a *quasi-LPV system*, where the scheduling parameters ρ_k are determined by x_k and u_k, the state trajectory can be predicted.

[1] In light of the observation made in Section 2.3.3, the control horizon is henceforth denoted by N and it is assumed that the prediction horizon $N_y \to \infty$ with the terminal cost bounding the tail of the cost.

Lemma 3.1. For the quasi-LPV system (3.1), the predicted state vector $x(k+l)$ can be calculated for each $l > 0$ from $x(k)$ and $u(k+i),\ i = 0, 1, \ldots, l-1$.

Proof. Let $\mathcal{A} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n\times n}$ and $\mathcal{B} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n\times m}$ denote the concatenation of maps $A \circ \varrho$ and $B \circ \varrho$, respectively, such that $A(\rho_k) = \mathcal{A}(x_k, u_k)$ and $B(\rho_k) = \mathcal{B}(x_k, u_k)$. A map can be defined

$$\Gamma^l : \mathbb{R}^n \times \mathbb{R}^m \times \ldots \times \mathbb{R}^m \ \to \ \mathbb{R}^n$$

such that given $x(k)$ and $u(k+i),\ i = 0, 1, \ldots, l-1$, the evolution of states governed by (3.1) can be expressed as

$$x_{k+l} = \Gamma^l(x_k, u_k, \ldots, u_{k+l-1}).$$

Using the shorthand notation $\Gamma^l_k = \Gamma^l(x_k, u_k, \ldots, u_{k+l-1})$, this map is defined recursively by

$$\Gamma^0_k = x_k,$$

$$\Gamma^1_k = \mathcal{A}(\Gamma^0_k, u_k)\Gamma^0_k + \mathcal{B}(\Gamma^0_k, u_k)u_k,$$

and

$$\Gamma^{l+1}_k = \mathcal{A}(\Gamma^l_k, u_{k+l})\Gamma^l_k + \mathcal{B}(\Gamma^l_k, u_{k+l})u_{k+l}.$$

It is straightforward to check that

$$x_{k+l} \ = \ \Gamma^l_k \ = \ \Phi^{l-1}_{k.0}x_k \ + \ \sum_{j=1}^{l} \Phi^{l-1}_{k.j}\mathcal{B}(\Gamma^{j-1}_k, u_{k+j-1})\, u_{k+j-1}, \tag{3.3}$$

where

$$\Phi^p_{k.q} = \begin{cases} \prod_{i=q}^{p} \mathcal{A}(\Gamma^i_k, u_{k+i}), & p \geq q \\ I_n, & p < q \end{cases}$$

and the right hand side of (3.3) depends only on $x(k)$ and control inputs up to $k+l-1$. □

Lemma 3.1 is easily illustrated with a simple example.

Example 3.1. Assume that a single input, first order system with a single parameter $\rho_k = x_k$ is given by

$$x_{k+1} = \underbrace{(a_0 + a_1\rho_k)}_{a(\rho_k)} x_k + bu_k,$$

substituting $\rho_k = x_k$ and $\rho_{k+1} = x_{k+1}$, an expression for $a(\rho_{k+1}) = a(x_{k+1})$ can be easily obtained

$$a(x_{k+1}) = a_0 + a_1x_{k+1} = a_0 + a_1a_0x_k + a_1^2x_k^2 + a_1bu_k.$$

The value of x_{k+2} is then

$$x_{k+2} = (a_0 + a_1 a_0 x_k + a_1^2 x_k^2 + a_1 b u_k)(a_0 x_k + a_1 x_k^2 + b u_k) + b u_{k+1}$$

A similar expression can be obtained for all x_{k+i}, $i = 1, 2, ..., N$. Note that the dependence of the predicted states on the control inputs is in this case polynomial, and in the general case nonlinear.

Analogous to the procedure presented in Section 2.3 for LTI systems, problem (P.1) can be expressed in a dense form. To do this, the following definitions are required

$$X_k = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N} \end{bmatrix} \qquad U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ x_{k+N-1} \end{bmatrix} \qquad \mathsf{P}_k = \begin{bmatrix} \rho_k \\ \rho_{k+1} \\ \vdots \\ \rho_{k+N-1} \end{bmatrix} \tag{3.4}$$

slightly abusing notation and for compactness $\varrho(X_k, U_k)$ is also used to denote $\mathsf{P}_k = \varrho([x_k^\top \;\; X_k^\top]^\top, U_k)$. Note that the map ϱ is not directly applied to X_k but to $[x_k^\top \;\; X_k^\top]^\top$, as the vector P_k contains the parameters in the interval $[k \quad k+N-1]$ whereas the state prediction is done for the interval $[k+1 \quad k+N]$.

The prediction equation can be found by recursively solving the state equation (3.1) for $k = 1, 2, ..., N$ yielding

$$X_k = \Phi(\mathsf{P}_k) x_k + \Gamma(\mathsf{P}_k) U_k \tag{3.5}$$

where

$$\Phi(\mathsf{P}_k) = \begin{bmatrix} A(\rho_k) \\ A(\rho_{k+1})A(\rho_k) \\ \vdots \\ A(\rho_{k+N-1})A(\rho_{k+N-2})...A(\rho_k) \end{bmatrix}$$

$$\Gamma(\mathsf{P}_k) = \begin{bmatrix} B(\rho_k) & 0 & ... & 0 \\ A(\rho_{k+1})B(\rho_k) & B(\rho_{k+1}) & ... & 0 \\ A(\rho_{k+2})A(\rho_{k+1})B(\rho_k) & A(\rho_{k+2})B(\rho_{k+1}) & ... & 0 \\ \vdots & & \ddots & \vdots \\ A(\rho_{k+N-1})...A(\rho_{k+1})B(\rho_k) & A(\rho_{k+N-1})...A(\rho_{k+2})B(\rho_{k+1}) & ... & B(\rho_{k+N-1}) \end{bmatrix}.$$

Substituting the prediction equation (3.5) onto the cost function yields a parameter dependent dense formulation of the optimization problem

$$\begin{aligned} &\min_{U_k} \frac{1}{2} U_k^\top H(\mathsf{P}_k) U_k + g(\mathsf{P}_k)^\top U_k \\ &\text{subject to} \\ &\quad u_{k+i} \in \mathcal{U} \qquad \forall i \in [0 \quad N-1] \\ &\quad [0 \quad ... \quad 0 \quad I_n]\Phi(\mathsf{P}_k) x_k + [0 \quad ... \quad 0 \quad I_n]\Gamma(\mathsf{P}_k) U_k \in \mathbb{X} \end{aligned} \tag{3.6}$$

where

$$\begin{aligned} H(\mathsf{P}_k) &= 2(\hat{R} + \Gamma(\mathsf{P}_k)^\top \hat{Q}\Gamma(\mathsf{P}_k)) \\ g(\mathsf{P}_k) &= 2\Gamma(\mathsf{P}_k)^\top \hat{Q}\Phi(\mathsf{P}_k)x_k. \end{aligned} \tag{3.7}$$

and $\hat{Q} = \mathrm{diag}(I_{N-1} \otimes Q, P)$, $\hat{R} = I_N \otimes R$.

Dependence on the current scheduling sequence makes it necessary to compute $H(\cdot)$ and $g(\cdot)$ online, as opposed to the LTI case where it can be hard-coded, albeit with a relatively expensive memory cost. Note that the terminal state constraint has been translated into an input constraint, as the state sequence is no longer a variable in this formulation.

3.1.1 Iterative Predictions

As shown in Lemma 3.1, the quasi-LPV MPC problem can be solved given that the parameter trajectory depends on the state and can thus be predicted as a function of previous states and inputs. However, the nonlinear nature of the problem would make its solution on a real-time environment difficult to implement. To overcome the complexity arising from the nonlinear optimization, an iterative algorithm is proposed (henceforth referred to as qLMPC, quasi-Linear Model Predictive Control) which requires only the solution of a sequence of Quadratic Programs (QP) to find the solution to the underlying nonlinear optimization problem.

Note that for fixed scheduling trajectories P_k, the predicted states X_k in (3.5) are linear in the control inputs U_k, and one can - just as in the case of LTI systems - find a solution to problem (P.1) by solving a QP. This motivates the following iterative approach:

- Initially problem (P.1) is solved with the quasi-LPV model (3.1) replaced by the LTI model that is obtained when the state-dependent scheduling sequence P_k is replaced by $\mathsf{P}_k^0 = 1 \otimes \varrho(x(0), u(0))$.
- A scheduling sequence P_k^l is then iteratively driven towards a possibly sub-optimal sequence $\mathsf{P}_k^* = \varrho(X_k^*, U_k^*)$, where X_k^* and U_k^* denote the state and input trajectories corresponding to the (sub-) optimal solution to (P.1).
- This is achieved by solving at iteration l the optimization problem (P.1) with P_k replaced by P_k^l, and by generating a new scheduling sequence from the resulting optimal state sequence X_k^l as $\mathsf{P}_k^{l+1} = \varrho(X_k^l, U_k^l)$.
- After the last iteration, X_k^* and U_k^* are used in the next time step to warm-start P_{k+1}^0, i.e. $\mathsf{P}_{k+1}^0 = \varrho(X_k^*, U_k^*)$ by appropriately shifting X_k^* and U_k^*.

Thus the idea is to solve a sequence of optimization problems where the quasi-LPV model (3.1) is replaced by an LTV model that is generated from (3.1) by imposing a fixed scheduling sequence, which is then updated at each iteration step using the optimized state sequence: the initial scheduling sequence P_k^0 yields an LTV system; this system is referred to as Σ^0. The optimization problem yields U_k^0 as an estimate of the control input, where the superscript is

used to indicate that the sequence corresponds to the system Σ^0. The computation of the state sequence X_k^0 is done using (3.5), which is then used to calculate a parameter trajectory for a subsequent iteration, i.e., $\mathsf{P}_k^1 = \varrho(X_k^0, U_k^0)$, this LTV system is now called Σ^1. A new input trajectory U_k^1 can then be found by solving (P.1) again. Note that input and state trajectories are calculated iteratively.

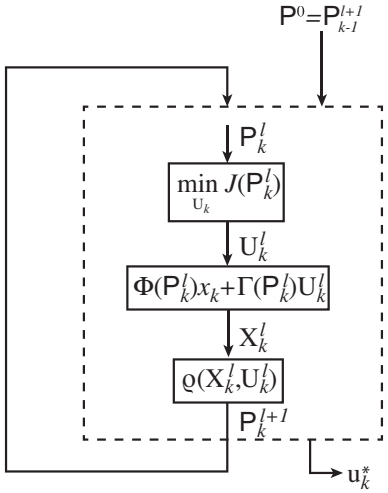
When $X_k^l \approx X_k^{l-1}$, the input sequence U_k^l gives an approximation of the optimal solution U_k^* to (P.1). The first element of the sequence is then applied to the plant and the procedure is repeated for all subsequent time steps. Implementation of the proposed approach is summarized in the algorithm in Figure 3.1a and a graphical depiction is shown in Figure 3.1b.

Stop Criterion

A crucial factor of any iterative procedure is the stop criterion. Whereas Sequential Quadratic Programming (SQP) and SQP-like methods (which qLMPC could be considered to be) for nonlinear optimization often use a stop criterion of the form $||X_k^l - X_k^{l-1}|| \leq \epsilon$ where ϵ is a predefined tolerance, in a real-time context such a stop criterion could cause the computation time to exceed the sampling time with potentially catastrophic effects. In the opposite end of the spectrum one could carry out only 1 iterative step per sampling time and leave convergence to the warm-start kind of *iteration*; this is a common practice in MPC as seen in e.g. [44]. In the case of qLMPC, experience has shown that 1-2 iterations give satisfactory results (see Figure 3.3 below). However, one should consider that by not driving the optimization to convergence at each time step, any stability guarantee would be lost in the first few time steps before convergence is achieved.

Initialization: plant model, $\hat{Q}$, $\hat{R}$, N
1: $k \leftarrow 0$
2: Define $\mathsf{P}^0 = \mathbf{1}_N \otimes \varrho(x(0), u(0))$
3: **repeat**
4: $\quad l \leftarrow 0$
5: $\quad$ **repeat**
6: $\quad\quad$ Solve (P.1) using P_k^l to obtain U_k^l
7: $\quad\quad$ Predict state $X_k^l = \Phi(\mathsf{P}_k^l) + \Gamma(\mathsf{P}_k^l)U_k^l$
8: $\quad\quad$ Define $\mathsf{P}_k^{l+1} = \varrho(X_k^l, U_k^l)$
9: $\quad\quad$ $l \leftarrow l + 1$
10: $\quad$ **until** stop criterion
11: $\quad$ Apply u_k to the system
12: $\quad$ Define $\mathsf{P}_{k+1}^0 = \varrho(X_k^l, U_k^l)$
13: $\quad k \leftarrow k + 1$
14: **until** end

(a)



(b)

Figure 3.1: qLMPC Algorithm

Example 3.2 (qLMPC Iterative Algorithm)**.** Consider the nonlinear system

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 4/3 + 0.2x_1(k) & -2/3 + 0.1x_2(k) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} u_k$$
$$y(k) = \begin{bmatrix} -2/3 & 1 \end{bmatrix}$$

with the constraint $u \in [-1 \;\; 1]$. The system is similar to the one presented in Example 2.1, albeit perturbed with nonlinear terms and with a scaled input. A linearization around the origin reveals that the origin is a stable equilibrium, however contrary to Example 2.1, this system is not globally stable nor can it be globally stabilized, given the input constraint. A quasi-LPV model can be readily obtained by defining $\rho_1(k) = x_1(k)$, $\rho_2(k) = x_2(k)$. Applying the algorithm in Figure 3.1a with $Q = C^\top C$, $R = 0.1$, $P = 10Q^a$ and $N = 10$ yields the closed-loop response shown below.

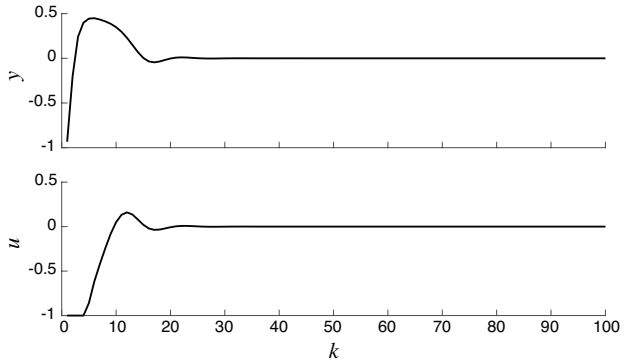
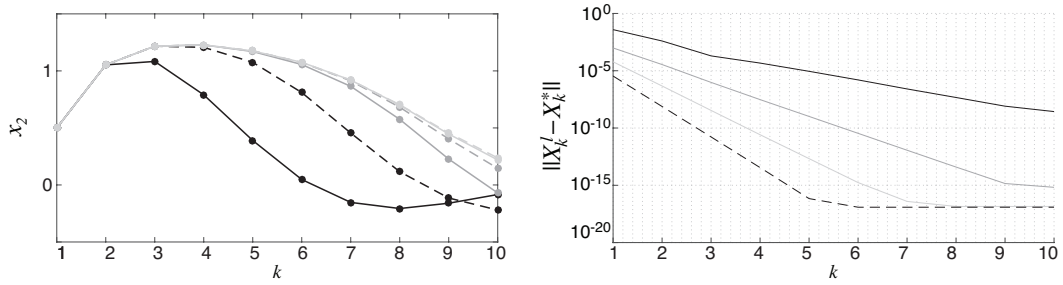


Figure 3.2: Closed-loop response of Example 3.2

In order to investigate the convergence properties of the iterative algorithm, 10 iterations were carried out. Prediction of the state x_2 at $k = 0$ after each iteration is shown in Figure 3.3a which highlights that predicted trajectories are indeed convergent. The norm of the prediction error at each iteration w.r.t the nonlinear prediction given U_k^{10} (i.e. the prediction resulting from nonlinear simulation given the predicted input sequence after 10 iterations) at several time steps is shown in Figure 3.3b, where it is clear that not only are the iterations converging, indeed so are the initial guesses (i.e. the real-time *iterations*) thanks to the warm-start. This leads to the conclusion that in this example even not using iterations would lead to a satisfactory response. Indeed, relying only on warm-start and not using intra-time-step iterations leads to a qualitatively identical closed-loop response.



(a) (b)

Figure 3.3: (a) Predicted x_2 trajectories at $k = 0$ after 1 (——), 2 (- - - - -), 3 (——), 4 (- - - - -), 5 (——) and 6 (- - - - -) iterations, iterations 7-10 are not shown for clarity.
(b) Prediction error given U_k^{10} w.r.t nonlinear simulation at $k = 0$ (——), $k = 5$ (——), $k = 10$ (——), and $k = 15$ (- - - - -).

[a] The terminal cost function is not positive definite in this case, this does not represent a problem as no stability analysis is to be made.

Convergence of the iterative scheme is addressed in Appendix A. The discussion is based on drawing an analogy to Newton-type Sequential Quadratic Programming. Indeed under some conditions, qLMPC can be understood as a version of Newton-SQP, for this reason an overview of SQP is given and conditions for local convergence are stated. The velocity-based qLMPC (Chapter 4) is seen to have a similar structure to SQP, enabling the same convergence analysis to be carried out.

3.2 Stability of quasi-LPV MPC

This section extends the stability result presented in Section 2.3.3 to LPV systems. Even though the focus of this thesis is on quasi-LPV systems, the stability analysis could be used for LPV systems as well. However, an approach akin to robust MPC would need to be applied in addition, since as established before, for LPV systems the prediction is uncertain. An implicit assumption made in the following result is that the iterative approach presented in the previous section has converged at least to a sub-optimal solution (see Remark 3.3).

Additionally, for ease of notation it is assumed that each individual parameter is a function of a single state or input, i.e. $\varrho_i : \mathbb{R} \to \mathbb{R}$

$$\rho = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_i \\ \vdots \end{bmatrix} = \begin{bmatrix} \varrho_1(x_j) \\ \vdots \\ \varrho_i(u_l) \\ \vdots \end{bmatrix}$$

and that each ϱ_i is bijective. It is therefore possible to map admissible parameter values into the maximum admissible state values by defining

$$\bar{x}_i = \max_{\mathcal{P}} |\varrho_j^{-1}(\rho_j)|. \tag{3.8}$$

This imposition on the selection of parameters might seem restrictive, it is however the natural and common choice when constructing a quasi-LPV model, exception being when LFT or polytopic synthesis techniques are used, which may require a redefinition of parameters, this is however usually conservative and not done here.

The proposed stability result is based on a non-trivial extension of the dual-mode control scheme presented earlier for LTI systems. Indeed the focus is now on parameter dependent *terminal ingredients* in order to reduce the conservatism entailed by a common Lyapunov function in a broad operating region. In particular, a parameter dependent terminal controller $F(\rho_k)$ is expected to be able to stabilize the plant with a larger region of attraction. Furthermore, a parameter dependent terminal constraint set $\mathbb{X}(\rho)$ (now a set-valued function of ρ) has more degrees of freedom which could potentially lead to a larger terminal set thereby enlarging the feasible region of the control law. The following is the quasi-LPV extension of Theorem 2.2.

Theorem 3.1 (Stabilizing MPC for quasi-LPV systems)**.** Assume a nominal controller $\tilde{F}(x_k, \rho_k) = F(\rho_k)x_k$, a terminal state domain $\mathbb{X}(\rho_k)$, a terminal cost function $\Psi(x_k, \rho_k)$ and an admissible parameter set $\mathcal{P}$ exist such that $\forall \rho \in \mathcal{P}, \forall k \in \mathbb{Z}_{\geq 0}$ the following conditions hold:

1. $0 \in \mathbb{X}(\rho_k)$
2. $(A(\rho_k) + B(\rho_k)F(\rho_k))x_k \in \mathbb{X}(\rho_k), \qquad \forall x_k \in \mathbb{X}(\rho_k)$
3. $\Psi(0, \rho_k) = 0, \Psi(x_k, \rho_k) > 0 \quad \forall x_k \neq 0$
4. $\Psi((A(\rho_k) + B(\rho_k)F(\rho_k))x_k, \rho_{k+1}) - \Psi(x_k, \rho_k) \leq -\ell(x_k, F(\rho_k)x_k), \quad \forall x_k \in \mathbb{X}(\rho_k)$
5. $F(\rho_k)x_k \in \mathcal{U}, \qquad \forall x_k \in \mathbb{X}(\rho_k)$
6. $\mathbb{X}(\rho_k) \subset \varrho^{-1}[\mathcal{P}]$

Then, assuming feasibility of the initial state, an MPC controller solving the optimization problem (P.1)[2] guarantees asymptotic stability of the origin and recursive feasibility.

Proof. The proof is divided into two parts: Part I proves recursive feasibility whereas Part II establishes monotonicity of the cost function, these steps are standard in the MPC literature.

Part I: Assume that the optimal control sequence at time step k is

$$U_k^* = [u_k^{*\top}, u_{k+1}^{*\top}, ..., u_{k+N-1}^{*\top}]^\top,$$

with similarly defined state and parameters sequences X_k^*, P_k^*. A feasible (possibly suboptimal) solution at time step $k+1$ is given by

$$U_{k+1} = [u_{k+1}^{*\top}, u_{k+2}^{*\top}, ..., u_{k+N-1}^{*\top}, (F(\rho_{k+N}^*)x_{k+N}^*)^\top]^\top \tag{3.9}$$

[2]It is assumed that $\mathcal{X} = \mathbb{R}^n$ otherwise an additional condition is that $\mathcal{P} \subseteq \varrho[\mathcal{X}]$.

obtained by shifting the previous optimal sequence and appending the second-mode control law $u_{k+N} = F(\rho^*_{k+N})x^*_{k+N}$ at the end. Note that this input is feasible, according to conditions 2. and 5. in Theorem 3.1 coupled with the terminal state constraint in problem (P.1), i.e. at the end of the horizon the state is inside the terminal region, which by condition 2. is an invariant set; within this set, the parameter-dependent state feedback law gives admissible inputs (cond. 5.).

Part II: Given the feasible input sequence from Part I, the resulting value of the cost function at time step $k+1$ is

$$\begin{aligned} J_{k+1} = J_k - x_k^\top Q x_k - u_k^{*\top} R u_k^* \\ - \Psi(x_{k+N}) + x_{k+N}^\top (Q + F^\top R F) x_{k+N} + \Psi(x_{k+N+1}), \end{aligned} \tag{3.10}$$

where the first term and terminal cost of J_k are subtracted and the last term and terminal cost of J_{k+1} are added for the equality to hold. According to condition 3. in Theorem 3.1, the second line of (3.10) is negative; from this, and from the fact that $J^*_{k+1} \leq J_{k+1}$ it follows that

$$J^*_{k+1} \leq J_{k+1} \leq J_k - x_{k+1}^\top Q x_{k+1} - u_k^{*\top} R u_k^*$$

which proves that J^* is a Lyapunov function and x converges to the origin.

□

Remark 3.1. *Condition 6. is necessary in the quasi-LPV case since the rest of the conditions must hold $\forall \rho_k \in \mathcal{P}$. If the terminal constraint set is such that $\mathbb{X} \not\subset \varrho^{-1}[\mathcal{P}]$ then $\exists x \in \mathbb{X} : \varrho(x,u) \notin \mathcal{P}$, thereby violating the assumption that $\rho \in \mathcal{P}$ (cf. Remark 2.2).*

Remark 3.2. *Note that conditions 2. and 3. make $\Psi(x,u)$ a control Lyapunov function.*

Remark 3.3. *The proof of Theorem 3.1 uses optimality to establish that the cost function decays with time. However, it is unrealistic to assume that optimality is achieved when solving a non-convex optimization problem online. Close inspection of Part II of the proof reveals that optimality is not necessary, as the cost resulting from using the feasible input sequence* (3.9) *is lower than the cost obtained at the previous time step (assuming that conditions 1.-6. hold and that the iterative algorithm has converged so that the input and parameter sequences from time step k are valid at $k+1$). A practical way to use this* suboptimal *MPC is to compute this feasible input sequence at every time step (i.e. use the previous solution and append the fixed control law $F(\rho^*_{k+N})x^*_{k+N}$ at the end) and use it as an initial guess (warm start) for the optimization problem. By using this feasible initial guess, the cost after optimization would be at most the one obtained by this sequence. This observation was made in [106] and [93], where a suboptimal version of the standard stability result is presented and thoroughly discussed. A streamlined analysis is presented in [98] Section 2.8, where the same conclusion is reached and some of the nuances of this choice are briefly presented.*

Theorem 3.1 provides conditions for stability of the equilibrium at the origin for a nonlinear system in quasi-LPV form. However, conditions 1.-6. make assumptions on the existence of parameter dependent terminal ingredients and an admissible parameter set $\mathcal{P}$, these assumptions are not trivial and need to be enforced by appropriate selection of $F(\rho)$, $\Psi(x,\rho)$ and $\mathbb{X}(\rho)$. In the following, one choice is given which makes their computation possible via the solution of

an optimization problem to be solved offline. Consider the quadratic terminal cost $\Psi(x,\rho) = x^\top P(\rho)x$ and the ellipsoidal terminal region[3]

$$\mathbb{X}(\rho) = \{x | x^\top W(\rho)x \leq 1\}. \tag{3.11}$$

To guarantee that this set is invariant under the control law $u = F(\rho)x$, it must hold that $F(\rho)x \in \mathcal{U}\ \forall x \in \mathbb{X}(\rho)$. Furthermore the volume of the set should be as large as possible. It is easy to check that the volume of the ellipsoid $x^\top Wx \leq 1$ is proportional to the determinant of W^{-1}, so the (convex) problem is that of determinant maximization. In addition, the following lemma is used:

Lemma 3.2 (Lemma 4.1 from [77])**.** The maximum of $c^\top x$ in the ellipsoidal set $x^\top Wx \leq 1$ is $\sqrt{c^\top W^{-1}c}$

Considering these choices, the following result gives a systematic way to find optimal values for the terminal ingredients.

Theorem 3.2. Let $Y(\rho) = P(\rho)^{-1}$ and $Z(\rho) = W(\rho)^{-1}$, then $\Psi(x,\rho) = x^\top P(\rho)x$, $\mathbb{X}(\rho) = \{x | x^\top W(\rho)x \leq 1\}$ and $F(\rho)$ satisfy the conditions of Theorem 3.1 if $Y(\rho)$ and $Z(\rho)$ are chosen as the solutions to the following optimization problems:

Feasibility problem

$$\max_{Y,F,t} t \tag{3.12a}$$

s.t.

$$\begin{bmatrix} Y(\rho) & * & * & * \\ (A(\rho)+B(\rho)F(\rho))Y(\rho) & Y(\rho+\Delta\rho) & * & * \\ Y(\rho) & 0 & Q^{-1} & * \\ F(\rho)Y(\rho) & 0 & 0 & R^{-1} \end{bmatrix} \geq 0, \tag{3.12b}$$

$$Y(\rho) \succ tI \quad t > 0 \qquad \forall \rho \in \mathcal{P},\ \Delta\rho \in \mathcal{V} \tag{3.12c}$$

Terminal region problem

$$\max_{Z,F,t} t \tag{3.13a}$$

s.t.

$$\begin{bmatrix} Z(\rho) & * \\ (A(\rho)+B(\rho)F(\rho))Z(\rho) & Z(\rho+\Delta\rho) \end{bmatrix} \geq 0, \tag{3.13b}$$

$$\begin{bmatrix} u_{i,max}^2 & e_i F(\rho)Z(\rho) \\ * & Z(\rho) \end{bmatrix} \geq 0 \qquad i \in [1\ m] \tag{3.13c}$$

$$e_j Z(\rho) e_j^\top \leq \overline{x}_{j,max}^2 \qquad j \in [1\ n_\rho] \tag{3.13d}$$

$$\text{geomean}(Z(\rho)) > t \quad t > 0 \qquad \forall \rho \in \mathcal{P},\ \Delta\rho \in \mathcal{V} \tag{3.13e}$$

[3]More specifically, the family of ellipsoidal terminal regions, parameterized by ρ.

In the inequalities in Theorem 3.2, $u_{max,i}$ denotes an upper bound on $|u_i|$, e_j is the j^{th} row of the identity matrix, $\overline{x}_j$ is the upper bound on state x_j as given by (3.8) and geomean is the geometric mean of a matrix.

Remark 3.4. *If a parameter independent terminal constraint set is used, the objective of the terminal region problem* (3.13a) *can be replaced by* $\max \det(W)$ *and the constraints* (3.13e) *with* $Z > 0$*; in the parameter dependent case, this is replaced by the geometric mean since "the geometric mean of the eigenvalues of a symmetric positive semidefinite matrix is a monotonic function of the determinant, and can thus be used as an alternative in determinant maximization" [78]; convexity of the problem is preserved using this function.*

Remark 3.5. *The BMIs of Theorem 3.2 need to be solved for all* $\rho \in \mathcal{P}$*. A pragmatic way to handle the infinite dimensionality of this problem is by gridding the parameters over the admissible set* $\mathcal{P}$*. As* $A(\cdot), B(\cdot)$ *are assumed to be continuous functions of* ρ*, an appropriately dense grid guarantees the constraints hold for the whole parameter range, an implicit assumption is also that the maps* $A(\cdot), B(\cdot)$ *fulfill a Lipschitz condition.*

Proof. Satisfaction of condition 1 follows immediately from the definition of the ellipsoid (3.11).

(3.13b) $\Rightarrow$ *condition 2*

Apply Schur complement to (3.13b) to get

$$\begin{aligned} Z(\rho + \Delta\rho) &\succeq 0 \\ Z(\rho) - Z(\rho)(A(\rho) + B(\rho)F(\rho))^\top Z(\rho + \Delta\rho)^{-1}(A(\rho) + B(\rho)F(\rho))Z(\rho) &\succeq 0. \end{aligned}$$

Pre- and post-multiplication by $W(\rho) = Z(\rho)^{-1}$ gives

$$(A(\rho) + B(\rho)F(\rho))^\top W(\rho + \Delta\rho)(A(\rho) + B(\rho)F(\rho)) - W(\rho) \preceq 0,$$

By defining the transition matrix $\Phi(\rho) = A(\rho) + B(\rho)F(\rho)$ the invariance condition from [101] is recovered, i.e.

$$\Phi(\rho)^\top W(\rho + \Delta\rho)\Phi(\rho) - W(\rho) \preceq 0, \qquad \forall \rho_k \in \mathcal{P} \tag{3.14}$$

which is equivalent to condition 2.

(3.12b) $\Rightarrow$ *condition 4:*

Apply Schur complement to (3.12b)to get

$$\begin{bmatrix} Q^{-1} & 0 \\ 0 & R^{-1} \end{bmatrix} \succeq 0$$

$$\begin{bmatrix} Y(\rho) - Y(\rho)QY(\rho) - Y(\rho)F(\rho)^\top RF(\rho)Y(\rho) & Y(\rho)(A(\rho) + B(\rho)F(\rho))^\top \\ (A(\rho) + B(\rho)F(\rho))Y(\rho) & Y(\rho + \Delta\rho) \end{bmatrix} \succeq 0$$

The first inequality is trivially fulfilled. Another use of the Schur complement to the second inequality yields

$$Y(\rho + \Delta\rho) \geq 0$$
$$(*)^\top Y(\rho + \Delta\rho)^{-1}(A(\rho) + B(\rho)F(\rho))Y(\rho) - Y(\rho) \leq -Y(\rho)QY(\rho) - (*)^\top RF(\rho)Y(\rho).$$

Proceed by pre- and post-multiplying the second inequality by $P(\rho) = Y(\rho)^{-1}$, and subsequently by $x^\top$ and x,

$$(*)^\top P(\rho + \Delta\rho)(A(\rho) + B(\rho)F(\rho))x - x^\top P(\rho)x \leq -x^\top Qx - (*)^\top RF(\rho)x.$$

Within the terminal region, where these conditions must hold, $x_{k+1} = (A(\rho_k) + B(\rho_k)F(\rho_k))x_k$, substituting this in the inequality above yields

$$x_{k+1}^\top P(\rho_k + \Delta\rho_k)x_{k+1} - x_k^\top P(\rho_k)x_k \leq -x_k^\top Qx_k - u_k^\top Ru_k,$$

which is exactly condition 4.

(3.13b) $\Rightarrow$ *condition 5:*

Apply Schur complement to (3.13b)

$$u_{i,max}^2 - e_i F(\rho)Z(\rho)F(\rho)^\top e_i^\top \geq 0$$

rearrange to obtain

$$F_i(\rho)Z(\rho)F_i(\rho)^\top \leq u_{i,max}^2$$

where $F_i(\rho)$ is the i^{th} row of $F(\rho)$. Using Lemma 3.2, this gives a condition for the i^{th} input to be upper bounded by $|u_{i,max}|$ within the ellipsoidal set

$$x^\top W(\rho)x \leq 1.$$

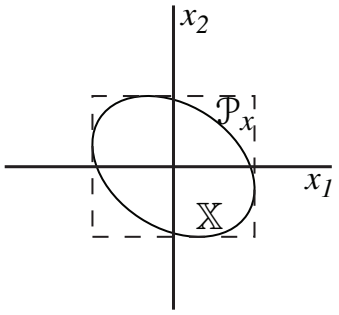
The same argument can be used to prove (3.13d) $\Rightarrow$ condition 6.

□

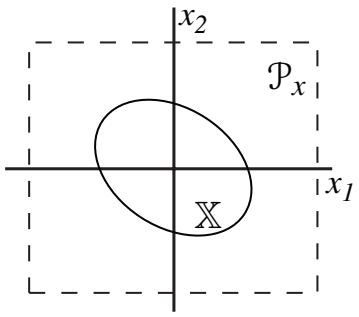
Remark 3.6. *Note that in Theorem 3.2 it is assumed that the input constraints are symmetric, i.e.* $-u_{i,max} \leq u_i \leq u_{i,max}$. *Asymmetric constraints can be included by using the smaller of the two bounds and assuming symmetric bounds, i.e.* $u_{i,max} = \min(|\underline{u}_i|, |\overline{u}_i|)$, *at the expense of conversevatism.*

Regarding condition 6. in Theorem 3.1, it imposes additional constrains on the size of the terminal ellipsoids. A graphical interpretation of this additional imposition is shown in Figure 3.4. In particular, assuming a small $\mathcal{P}$ makes the optimization problem in Theorem 3.2 easier to solve, as the operating region is smaller, it however limits the ellipsoid to lie within this small set given by $\mathcal{P}_x = \varrho^{-1}[\mathcal{P}]$ (Figure 3.4a). Assuming a large $\mathcal{P}$, might not directly limit the size

of $\mathbb{X}$, it does, however, limit it indirectly by making the optimization problem harder to solve (Figure 3.4b). In practice the admissible parameter set $\mathcal{P}$ is found either by physical intuition or by trial-and-error in order to find the maximum size ellipsoid; as it often happens, there is a trade-off to be made and an iterative procedure might be necessary. If $\mathcal{X} \neq \mathbb{R}^n$, then, in addition, $\mathcal{P}_x \subseteq \mathcal{X} \Rightarrow \mathbb{X} \subseteq \mathcal{X}$



(a) Assuming a small $\mathcal{P}$ limits $\mathbb{X}$ to lie within it



(b) Assuming a large $\mathcal{P}$ makes the optimization problem harder to solve, thereby indirectly limiting the size of $\mathbb{X}$

Figure 3.4: Effect of the choice of $\mathcal{P}$ in the size of $\mathbb{X}$

Note that both the feasibility problem (3.12) and the terminal region problem (3.13) are BMI problems, coupled by the state feedback law $F(\rho)$. These are, in the presented form, not convex and generally hard to solve. One possible simple way to solve them is sequentially:

- perform a linearizing change of variable $X(\rho) = F(\rho)Y(\rho)$ in (3.12b), thus converting it into an LMI, solve the feasibility problem for $Y(\rho)$ and $X(\rho)$
- compute $F(\rho) = X(\rho)Y(\rho)^{-1}$
- solve the terminal region problem, which given $F(\rho)$ computed in the previous step, is an LMI in the variables $Z(\rho)$, t. Note that this problem is always feasible if the feasibility problem had a solution.

Indeed this procedure is efficient and is recommended when parameter independent terminal ingredients are used. In the parameter dependent case, however, the feasibility problem often yields aggressive second-mode controllers (as compared to their parameter independent counterparts) which in turn result in a small terminal region. For this reason, an algorithm akin to $D-K$ iteration was proposed (see [37]) to iteratively find less aggressive controllers and hence larger terminal regions.

A more efficient way to find the terminal ingredients by solving a convex problem, in exchange for some conservatism, is to constrain the terminal region to be a sublevel set of the terminal cost function $\Psi(x, \rho)$, i.e.

$$\mathbb{X}(\rho) = \{x | x^\top P(\rho) x \leq \alpha\} \tag{3.15}$$

with $\alpha > 0$. Conservatism stems from the fact that invariance of the terminal region does not require contractivity, as opposed to the control Lyapunov function $\Psi(\cdot)$ which by condition 4. needs to be contractive. The trade-off however, makes the problem much easier to solve as both the feasibility and the terminal region problem are solved at once by a single LMI problem. Note that the size of the ellipsoid now depends on α (more specifically on P/α) so that it is now required to maximize the value of α.

Corollary 3.1. Let $Y(\rho) = P(\rho)^{-1}$, $X(\rho) = F(\rho)Y(\rho)$ and $\tilde{\alpha} = 1/\alpha$, then $\Psi(x, \rho) = x^\top P(\rho) x$, $\mathbb{X}(\rho) = \{x | x^\top P(\rho) x \leq \alpha\}$ and $F(\rho)$ satisfy the conditions of Theorem 3.1 if $Y(\rho)$ and $X(\rho)$ and $\tilde{\alpha}$ are chosen as the solutions to the following optimization problem

$$\min_{X,Y,\tilde{\alpha}} \ \tilde{\alpha} \tag{3.16a}$$

s.t.

$$\begin{bmatrix} Y(\rho) & * & * & * \\ A(\rho)Y(\rho) + B(\rho)X(\rho) & Y(\rho + \Delta\rho) & * & * \\ Y(\rho) & 0 & Q^{-1} & * \\ X(\rho) & 0 & 0 & R^{-1} \end{bmatrix} \geq 0, \tag{3.16b}$$

$$\begin{bmatrix} \tilde{\alpha} u_{i,max}^2 & e_i X(\rho) \\ * & Y(\rho) \end{bmatrix} \geq 0 \qquad i \in [1 \ m] \tag{3.16c}$$

$$e_j Y(\rho) e_j^\top \leq \tilde{\alpha} \bar{x}_{j,max}^2 \qquad j \in [1 \ n_\rho] \tag{3.16d}$$

$$Y(\rho) \succ 0 \quad \tilde{\alpha} > 0 \qquad \forall \rho \in \mathcal{P}, \ \Delta\rho \in \mathcal{V} \tag{3.16e}$$

Remark 3.7. *Whereas the controller $F(\rho)$ is a fictitious construct to prove stability, both P and $W = 1/\alpha P$ impact the feasibility and performance of the control law. In particular, it may happen that a large P is found, coupled with a large α, so that the ellipsoid (whose size is proportional to* $\det(\sqrt{\alpha} P^{-1/2})$ *is large, however a very large value of P is not desired since it may lead to a loss in performance. In order to keep the value of P within an acceptable range, the optimization objective* (3.16a) *may be modified, e.g. to be* $\min_{X,Y,\tilde{\alpha},t} \tilde{\alpha} - at$ *with the additional LMI $Y(\rho) > t$, $\forall \rho \in \mathcal{P}$, where $a > 0$ is a tuning parameter to weight the desire to make the ellipsoid large or P small.*

Proof. The proof is homologous to that of Theorem 3.1, it only remains to prove that the sublevel set is a valid terminal region.

(3.16c) $\Rightarrow$ *condition 4:*

Apply Schur complement to (3.16c) and substitute $F(\rho)Y(\rho) = X(\rho)$

$$\tilde{\alpha} u_{i,max}^2 - e_i F(\rho) Y(\rho) F(\rho)^\top e_i^\top \geq 0$$

rearrange to obtain

$$F_i(\rho)(\alpha Y(\rho)) F_i(\rho)^\top \leq u_{i,max}^2$$

where $F_i(\rho)$ is the i^{th} row of $F(\rho)$. Using Lemma 3.2, this gives a condition for the i^{th} input to be upper bounded by $|u_{i,max}|$ within the ellipsoidal set

$$x^\top \left(\frac{1}{\alpha} P(\rho)\right) x \leq 1$$

or equivalently $x^\top P(\rho) x \leq \alpha$. Invariance of this set (condition 2) follows from condition 3 (given that it is a sublevel set), proved above. □

Compared with the BMI formulation in Theorem 3.1 this approach requires the solution of an LMI problem, which is easily solvable, so that no sequential or iterative procedure for the solution is necessary. On the other hand, this comes at the cost of requiring the terminal set to fulfill a stronger condition (3.16b), rather than the milder condition (3.13b). This could potentially lead to conservatism in the sense that the optimal solution to the BMI formulation may be better than the one obtained using Corollary (3.1).

3.3 Offset-free set point tracking

In the discussion above, the regulator problem is treated, i.e. the objective is to stabilize the origin of the state space. Often it is desired to drive the system to different operating conditions thereby needing tracking capabilities. Define the measured (tracking) output as

$$y_k = C(\rho_k) x_k \tag{3.17}$$

$y \in \mathbb{R}^l$, the reference set point as $r \in \mathbb{R}^l$ and the steady state value for the state, input and parameter vector corresponding to r as x_s, u_s and $\rho_s = \varrho(x_s, u_s)$, respectively, i.e. x_s, u_s, ρ_s are such that

$$\begin{aligned} x_s &= A(\rho_s) x_s + B(\rho_s) u_s \\ r &= C(\rho_s) x_s. \end{aligned} \tag{3.18}$$

Note that equation (3.18) holds only in the case that $r \in C(\rho_s)[\mathcal{X} \cap F(\rho_s)^{-1}[\mathcal{U}]]$ [4]. If the reference is such that its corresponding steady state is not in the set of admissible states, or otherwise, if as in any realistic scenario, the system is subject to disturbances, an optimization problem is to be solved in order to find the optimum admissible steady state. The optimization problem that yields the optimal admissible steady state is called a *target selector*, the quasi-LPV extension of the LTI target selector reviewed in Section 2.3.1 is presented next.

Target selector. As mentioned above, the goal of a target selector is to calculate admissible steady state values for the input and the state vector, taking into account constraints and the effect of disturbances. In the quasi-LPV case, the optimization problem is given by

[4] The set $C(\rho_s)[\mathcal{X} \cap F(\rho_s)^{-1}[\mathcal{U}]]$ is the set of admissible steady state outputs, i.e. the set of outputs for which the corresponding steady states and inputs are admissible.

$$\min_{x_s,u_s} \frac{1}{2}||C(\rho_s)x_s + C_d\hat{d} - r||^2_{Q_s} + ||u_s||^2_{R_s}$$

subject to

$$\begin{bmatrix} I - A(\rho_s) & -B(\rho_s) \\ C(\rho_s) & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B_d\hat{d} \\ r - C_d\hat{d} \end{bmatrix} \tag{3.19}$$

$$x_s \in \mathcal{X}$$

$$u_s \in \mathcal{U}$$

where $Q_s \in \mathbb{R}^{l\times l}$, $R_s \in \mathbb{R}^{m\times m}$ are tuning parameters and $\hat{d}$ is the estimate of the disturbance, coming from the observer. In the quasi-LPV case, the equality constraint is now parameter dependent and indeed depends on the solution of the optimization problem itself, as $\rho_s = \varrho(x_s, u_s)$. Similar to how it is proposed for the MPC problem, iterations can be carried out to converge to (possibly sub-) optimal steady state x_s, u_s. In this case, however, the number of decision variables is comparatively small and hence one can expect fast convergence. As it was in the LTI case, the target selector problem uses the current disturbance estimate, so this problem is to be solved at every time step and not only when a change in reference takes place (cf. Figure 2.3).

Having obtained the admissible optimal steady-state values x_s, u_s, the stage cost can then be redefined as

$$\ell(x, u, x_s, u_s) = (x - x_s)^\top Q(x - x_s) + (u - u_s)^\top R(u - u_s), \tag{3.20}$$

and similarly for the terminal cost $\Psi(x, \rho, x_s) = (x_N - x_s)^\top P(x_N - x_s)$. The stability results presented in Theorems 3.1, 3.2 and Corollary 3.1 also need to be redefined, as the origin is shifted to (x_s, u_s). This would require the computation of a terminal set which fulfills conditions 1.-6. in Theorems 3.1 for all $\rho \in \mathcal{P}$ as well as for all $(x_s, u_s) \in \mathcal{X} \times \mathcal{U}$ (or at least $(x_s, u_s) \in \mathcal{X}_s \times \mathcal{U}$ where $\mathcal{X}_s \subset \mathcal{X}$ is a set of admissible steady-states). This has been studied e.g. in [121][13], it is however, limited to low order systems, since partitioning a high order state space would quickly make the LMIs intractable. This approach is not pursued here in the context of state space models because of its limited practicality; an input-output version of this is discussed in Section 5.3.1 where the partition is done in the output space which is typically considerably lower dimensional than the state space. Tracking in the state-space framework is discussed in more detail in the following chapter in the context of velocity-based qLMPC, where it is again more practical.

In practical applications, where the terminal constraint might be forgone in lieu of improved feasibility and lower computational complexity, it is still advisable to use an optimal terminal cost function $\Psi(\cdot)$ for improved performance. To construct a suitable terminal cost, a pragmatic choice is to solve the feasibility problem (3.12). Parameter dependency on P and F can be dropped if the system is quadratically stabilizable, i.e. if there exists a constant P such that the above condition is met $\forall \rho \in \mathcal{P}$ otherwise a parameter dependent Lyapunov function and state feedback gain can be used.

3.4 State and nonlinear constraints

In this section the previously made assumption that $\mathcal{X} = \mathbb{R}^n$ is relaxed. In this case, the additional condition that $\mathbb{X} \in \mathcal{X}$ would need to be added to Theorem 3.1; the characterization of this condition in LMI form is case-dependent and would possibly require an over bounding of $\mathcal{X}$ by a polyhedron in order to be able to express it as a convex constraint similar to (3.13d).

Inclusion of state constraints to the optimization problem in dense form (3.6) is carried out indirectly: the predicted state was eliminated from the cost function by making the substitution $X_k = \Phi(\mathsf{P}_k)x_k + \Gamma(\mathsf{P}_k)U_k$, using this same substitution state constraints can be included in the optimization problem as additional constraints on the input trajectory U_k.

Let the *constrained output* be defined as

$$y_c(k) = h(x_k) = C_c(\rho_k)x_k$$

$y_c \in \mathbb{R}^{l_c}$, note that this is in general different from the tracking output (3.17), although it might coincide in a given application. The predicted constrained output is thus

$$Y_c(k) = \hat{C}_c(\mathsf{P}_k^c)X_k = \hat{C}_c(\mathsf{P}_k^c)(\Phi(\mathsf{P}_k)x_k + \Gamma(\mathsf{P}_k)U_k)$$

where

$$\hat{C}_c(\mathsf{P}_k^c) = \mathrm{diag}(C_c(\rho_{k+1}^c), C_c(\rho_{k+2}^c), ..., C_c(\rho_{k+N}^c)),$$

and the parameter vector

$$\mathsf{P}_k^c = \varrho_c(X_k) = \begin{bmatrix} \rho_{k+1}^c \\ \rho_{k+2}^c \\ \vdots \\ \rho_{k_N}^c \end{bmatrix}$$

is used to schedule the constraint. Note that ρ^c is in general different from the scheduling vector ρ used for the plant dynamics (3.1); furthermore, if $\rho^c = \rho$, then $\mathsf{P}^c = q\mathsf{P}$ where q is the forward time shift operator.

Given a constraint $y_c(k+i) \leq b_c \ \forall i = 1, ..., N$ where b_c is constant, the definitions above can be used to constrain the predicted state and by extension the input trajectory as

$$C_c(\mathsf{P}_k^c)\Gamma(\mathsf{P}_k)U_k \leq \mathbf{1}b_c - C_c(\mathsf{P}_k^c)\Phi(\mathsf{P}_k)x_k. \tag{3.21}$$

Using this formulation, nonlinear constraints can be accommodated by finding a quasi-LPV representation of $h(x_k)$ and making C_c parameter dependent. By using the predicted parameter trajectory P_k^c this constraint becomes linear and is written in the standard form $Ax \leq b$, which preserves convexity of the original QP.

A straightforward way to find quasi-LPV representations of nonlinear constraints is by linearization. Indeed assuming $h(x)$ is continuously differentiable, then differentiating $h(x)$ w.r.t x in the inequality constraint $h(x) \leq b_c$ and setting $\rho_c = x$ results in

$$\underbrace{\nabla h(x)|_{\rho^c}}_{C_c(\rho^c)} x \leq \underbrace{b_c - h(\rho^c) + \nabla h(\rho_c)\rho_c}_{\tilde{b}_c(\rho^c)}$$

so that (3.21) can be written as

$$C_c(\mathsf{P}_k^c)\Gamma(\mathsf{P}_k)U_k \leq \hat{\hat{b}}_c(\mathsf{P}_c) - C_c(\mathsf{P}_k^c)\Phi(\mathsf{P}_k)x_k \tag{3.22}$$

where

$$\hat{\hat{b}}_c(\mathsf{P}_k^c) = \begin{bmatrix} b_c - h(\rho_{k+1}^c) + \nabla h(\rho_{k+1}^c)\rho_{k+1}^c \\ b_c - h(\rho_{k+2}^c) + \nabla h(\rho_{k+2}^c)\rho_{k+2}^c \\ \vdots \\ b_c - h(\rho_{k+N}^c) + \nabla h(\rho_{k+N}^c)\rho_{k+N}^c \end{bmatrix}.$$

Note that a quasi-LPV representation of constraints can be easily obtained by linearization because the affine term of the linearization is grouped with the affine term of the inequality, b_c. For the state equation, however, performing a linearization would bring an additional affine term to the RHS of (3.1), and the prediction equation (3.5) would need to carry over this affine term, increasing computational complexity. Furthermore, the stability analysis would need to account for this additional term as well. The next Chapter presents a more efficient procedure to obtain a quasi-LPV representation of the equations of motions via velocity-based linearization.

3.5 Application Example: qLMPC of an Arm-Driven Inverted Pendulum

This section presents an experimental application of the paradigm presented throughout the chapter. The chosen process is an Arm-Driven Inverted Pendulum (ADIP), also known as PenduBot [113]. This plant is a planar 2-DOF robot which is actuated at the shoulder and unactuated at the elbow, see Figure 3.5a. As opposed to other pendula often used as controller test benches, which only have two *distinct* equilibria (the stable down and the unstable up equilibria), the ADIP has a continuous locus of distinct (forced) equilibria, these can be categorized into

- DOWN-DOWN, $|\theta_1| > \pi/2$, $\theta_2 = \pi$
- DOWN-UP $|\theta_1| > \pi/2$, $\theta_2 = 0$
- UP-DOWN $|\theta_1| < \pi/2$, $\theta_2 = \pi$
- UP-UP $|\theta_1| < \pi/2$, $\theta_2 = 0$

For this application the goal is to stabilize the pendulum on the UP-UP configuration, and bring it back to the origin from any initial condition within the domain of attraction. The experiment is performed on a rotary inverted pendulum from Quanser (Figure 3.5b), installed in planar configuration, the equations of motion are given by

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = \begin{bmatrix} \tau \\ 0 \end{bmatrix}$$

(a) (b)

Figure 3.5: Arm-driven Inverted Pendulum

where $\theta = [\theta_1 \;\; \theta_2]^\top$,

$$M(\theta) = \begin{bmatrix} (m_1 + m_2)l_1^2 & m_2 l_1 l_2 \cos(\theta_1 - \theta_2) \\ m_2 l_1 l_2 \cos(\theta_1 - \theta_2) & m_2 l_2^2 \end{bmatrix}$$
$$C(\theta, \dot{\theta}) = \begin{bmatrix} f_{v1} & m_2 l_1 l_2 \sin(\theta_1 - \theta_2)\dot{\theta}_2 \\ -m_2 l_1 l_2 \sin(\theta_1 - \theta_2)\dot{\theta}_1 & f_{v2} \end{bmatrix}$$
$$G(\theta) = \begin{bmatrix} -(m_1 l_{g1} + (m_h + m_2)l_1)g \sin(\theta_1) \\ -m_2 l_2 g \sin(\theta_2) \end{bmatrix}$$

and physical parameters are listed in Table 3.1. Given that the goal is to maintain $\theta_2 \approx 0$, a linearization with respect to θ_2 can be carried out to simplify the model, furthermore using the substitution $\text{sinc}(\theta) = \sin\theta/\theta$, the gravitational forces vector $G(\cdot)$ can be written as $G(\cdot) = K(\cdot)\theta$ yielding

$$\tilde{M}(\theta)\ddot{\theta} + \tilde{C}(\theta, \dot{\theta})\dot{\theta} + K(\theta)\theta = \begin{bmatrix} \tau \\ 0 \end{bmatrix}$$

where

$$\tilde{M}(\theta) = \begin{bmatrix} (m_h + m_2)l_1^2 & m_2 l_1 l_2 \cos(\theta_1) \\ m_2 l_1 l_2 \cos(\theta_1) & m_2 l_2^2 \end{bmatrix} \qquad \tilde{C}(\theta, \dot{\theta}) = \begin{bmatrix} f_{v1} & 0 \\ -m_2 l_1 l_2 \sin(\theta_1)\dot{\theta}_1 & f_{v2} \end{bmatrix}$$
$$K(\theta) = \begin{bmatrix} -(m_1 l_{g1} + (m_h + m_2)l_1)g\,\text{sinc}(\theta_1) & 0 \\ 0 & -m_2 l_2 g \end{bmatrix}$$

which is readily turned into a qLPV model by defining the state $x = [\theta^\top \;\; \dot{\theta}^\top]^\top$ and the scheduling parameter vector $\rho = [\theta_1 \;\; \dot{\theta}_1]^\top$ leading to

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I \\ -\tilde{M}(\rho)^{-1}K(\rho) & -\tilde{M}(\rho)^{-1}\tilde{C}(\rho) \end{bmatrix}}_{A_c(\rho)} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \tilde{M}(\rho)^{-1} \end{bmatrix}}_{[B_c(\rho) \;\; *]} \begin{bmatrix} \tau \\ 0 \end{bmatrix}.$$

What makes the ADIP an interesting benchmark problem is that, not only do the locus of equilibria lie on a nonlinear manifold, the controllability of the system itself is strongly influenced by the pose of the arm to the point that it becomes uncontrollable if $\theta_1 = \pi/2$. This is illustrated

Table 3.1: ADIP physical parameters

l_1	Arm length	13.95	cm
l_{g1}	Arm length (center of gravity)	6.975	cm
l_2	Pendulum length	7.8	cm
m_1	Arm mass	115	g
m_h	Encoder 2 mass	130	g
m_2	Pendulum mass	73.1	g
f_{v1}	Arm friction coefficient	$1.3 \cdot 10^{-3}$	$\mathrm{N} \cdot \mathrm{s}$
f_{v2}	Pendulum friction coefficient	$2.2 \cdot 10^{-5}$	$\mathrm{N} \cdot \mathrm{s}$

in Figure 3.6 where the smallest singular value of the controllability Gramian[5] of the ADIP at equilibrium positions in the range $-\pi/2 < \theta_1 < \pi/2$ is shown. Evidently and intuitively, the plant becomes more difficult to control the more the operating condition approaches the uncontrollable equilibrium $\theta_1 = \pi/2$.

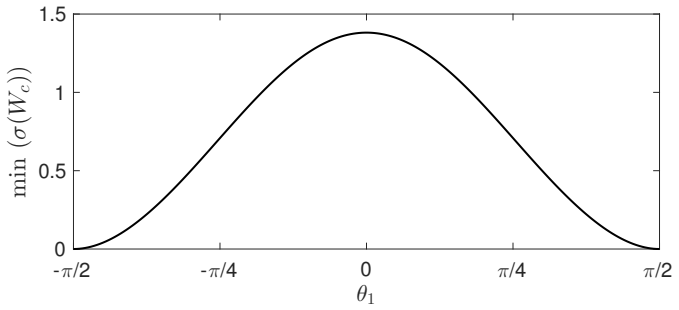


Figure 3.6: Controllability Gramian at equilibrium positions for $-\pi/2 < \theta_1 < \pi/2$

3.5.1 Stabilizing qLMPC

In order to compute suitable terminal ingredients, the approach in Corollary 3.1 is employed for a 4th order discretization of the quasi-LPV model in (3.5) with a sampling time of $T_s = 0.01$ s. The discretization is obtained by a polynomial approximation of the matrix exponential leading to

$$A(\rho) = \frac{T_s^4}{24}A_c(\rho)^4 + \frac{T_s^3}{6}A_c(\rho)^3 + \frac{T_s^2}{2}A_c(\rho)^2 + T_s A_c(\rho) + I$$
$$B(\rho) = \frac{T_s^4}{24}A_c(\rho)^3 B_c(\rho) + \frac{T_s^3}{6}A_c(\rho)^3 B_c(\rho) + \frac{T_s^2}{2}A_c(\rho)B_c(\rho) + T_s B(\rho).$$

[5]The standard controllability Gramian is only defined for stable linear systems, the present system has neither of these features. In order to quantify the controllability of the system, the scheduling parameters are frozen at equilibrium points parameterized by θ_1. The Gramian for unstable linear systems as proposed in [133] is used as an indicator of controllability of the resulting linear system.

The tuning parameters are chosen as $Q = \text{diag}(200, 1000, 0.1, 10)$, $R = 2000$, $N = 40$ and the admissible parameter set, and parameter rate set are chosen as

$$\mathcal{P} : \begin{cases} \rho_1 \in [-\pi/3 \;\; \pi/3] \\ \rho_2 \in [-1 \;\; 1] \end{cases} \qquad \mathcal{V} : \begin{cases} \Delta\rho_1 \in [-0.01 \;\; 0.01] \\ \Delta\rho_2 \in [-0.01 \;\; 0.01] \end{cases}$$

respectively. Note that admissible parameter values, and parameter rates are only specified for the second mode of the dual-mode controller (the fictitious state feedback controller within the terminal region), i.e. these sets do not limit the operating region as it is customary in LPV control; they can however limit the size of the terminal region (recall Figure 3.4).

Both parameter dependent, and independent terminal ingredients are obtained by solving the LMI problem (3.16), where initially the parameter $\tilde{\alpha}$ is minimized. As briefly discussed in Remark 3.7, the resulting terminal cost matrix P is indeed too large for practical implementation (largest singular value $\sigma_{\max}(P)$ in the order 10^{10}), for this reason the cost function is redefined to $a_1\tilde{\alpha} + a_2 t$ as per Remark 3.7 where $a_1 = 0.5, a_2 = 10000$ for the parameter independent case and $a_1 = 5, a_2 = 10000$ for the parameter dependent case[6], obtaining for both cases $\sigma_{\max}(P)$ in the order of 10^6. A comparison of the size of the resulting ellipsoids, projected onto planes of interest, is shown in Figure 3.7. In the figure, along with the ellipsoids, the projection of the admissible parameter values $\mathcal{P}$ onto the same planes is shown; in addition to admissible parameter values, both θ_2 and $\dot{\theta}_2$ are bounded as the equations of motion were linearized w.r.t these variables, so their values should not be large.

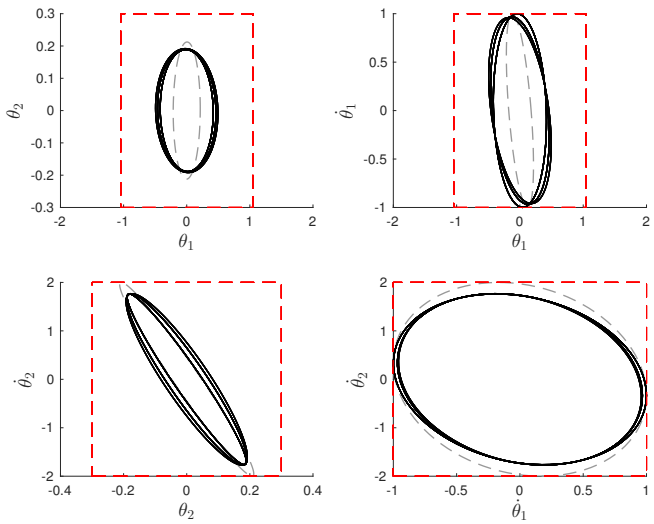


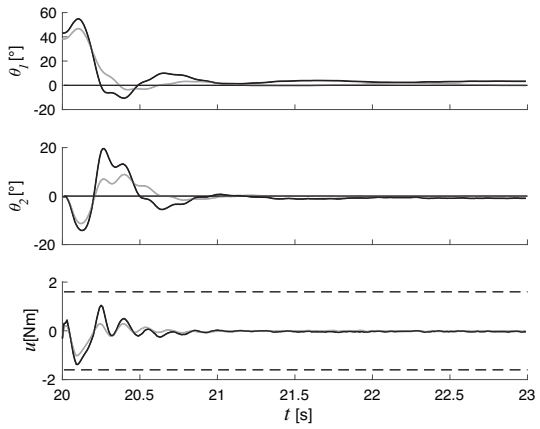
Figure 3.7: Parameter independent (- - - -) and parameter dependent (——) ellipsoids, $\mathcal{P}_\mathcal{X}$ (- - - -)

[6]The reason for the discrepancy in the cost function between parameter dependent and independent case is that, given the extra degrees of freedom, the parameter dependent case reaches a lower optimal cost; clearly priority is given to limiting the maximum eigenvalue of P so that this is done at the expense a larger $\tilde{\alpha}$. The used weightings were empirically obtained to yield $\sigma_{\max}(P)$ in the same order for both cases.

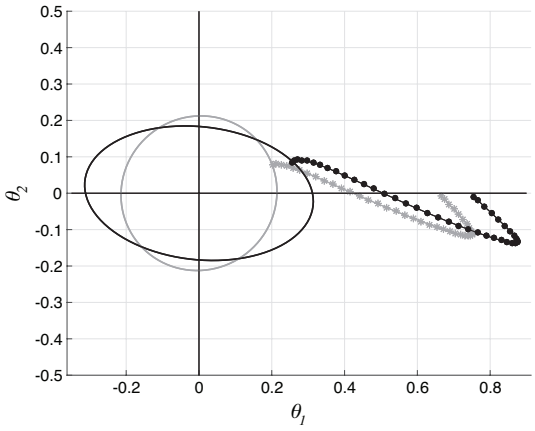
Experimental result

The algorithm in Figure 3.1a is applied using both parameter dependent and independent terminal ingredients as discussed in the previous section. For this experiment, 2 iterations of the algorithm are carried out, in order to speed up convergence and ensure satisfaction of the terminal constraints. As the experimental setup uses incremental encoders (meaning the zero position is calibrated the moment the experiment is switched on) it is not possible to give arbitrary initial conditions for the regulator. To solve this, the same MPC law, yet without terminal constraint is used to stabilize the system and drive it away from the origin (as disclosed in Section 3.3 forgoing the observer and target selector as there is no need for offset-free tracking) to give a non-zero initial condition to the MPC with terminal constraint and by extension stability guarantees, which is then enabled at $t = 20$ s.

The closed-loop behavior in the experiments can be seen in Figure 3.8a, note that although the MPC with parameter independent terminal ingredients appears to have better closed-loop performance, this is mainly due to the fact that the initial condition in the parameter dependent case is farther from the origin, as the comparatively larger ellipsoids enable a larger admissible set, if admittedly the difference is not excessive. In both cases performance is satisfactory and both stability and recursive feasibility are guaranteed by satisfaction of the terminal constraint at $k = 0$ (which corresponds to $t = 20$ s, i.e. when the MPC with stability guarantees is enabled). A graphical depiction of the terminal constraint is shown in Figure 3.8b in the $\theta_1 - \theta_2$-plane, where it is apparent that the predicted trajectory reaches the ellipsoid in both cases.



(a) Closed-loop response of MPC with parameter independent (——) and parameter dependent (——) terminal ingredients



(b) Predicted trajectories of MPC with parameter independent (——) and parameter dependent (——) terminal ingredients at $k = 0$ and corresponding ellipsoids

Figure 3.8: Comparison of experimental results for MPC with parameter dependent and independent terminal ingredients

3.5.2 Set point tracking - no terminal constraints

It is expected that stability guarantees come at a cost, in this case in addition to compromising feasibility on a wide operating region, redefining the terminal ingredients to apply to tracking is not practical in the SS framework as it would often necessitate a partition of the state space. If one relinquishes the terminal constraint, however, the algorithm becomes much more flexible and outstanding tracking performance can be achieved. To explore this, this section tackles the problem of stabilizing the ADIP in as wide an operating region as possible. Consider the stage cost (3.20) and assume initially that there is no expected disturbance. In this particular case, computing the steady state values via the target selector (3.19) is therefore not necessary, as they are obvious:

$$x_s = [r \;\; 0 \;\; 0 \;\; 0]^\top$$

and u_s is given by the solution to (note the absence of a disturbance)

$$\begin{bmatrix} I - A(\rho_s) & -B(\rho_s) \\ \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}.$$

Implementation of the control law as described above with tuning matrices $Q = \mathrm{diag}(700, 1000, 0.01, 10)$, $R = 2000$, a prediction horizon of $N = 40$ and only 1 iteration of the algorithm in Figure 3.1a results in the closed-loop response shown in Figure 3.9. While the range of operation is indeed remarkable (cf. [19], where the 70° mark was achieved using an identified model), tracking performance, particularly in steady-state is not nearly as impressive. Clearly the assumption that the experiment is disturbance-free is unrealistic and modelling mismatches prevent adequate reference tracking.

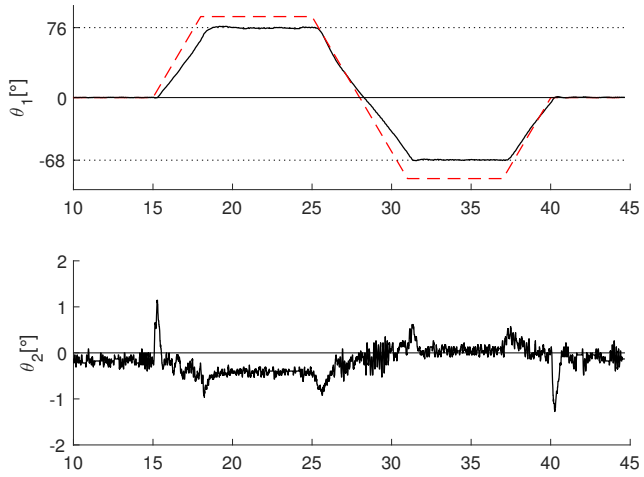


Figure 3.9: ADIP closed-loop response - no terminal constraint case

Disturbance models

A disturbance model can be included by augmenting the state with an integrating disturbance, i.e.

$$\begin{aligned}\begin{bmatrix} x_{k+1} \\ d_{k+1} \end{bmatrix} &= \begin{bmatrix} A(\rho_k) & B_d(\rho_k) \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} + \begin{bmatrix} B(\rho_k) \\ 0 \end{bmatrix} u_k \\ y_k &= \begin{bmatrix} I & C_d(\rho) \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix}.\end{aligned} \tag{3.23}$$

where $B_d(\cdot)$, $C_d(\cdot)$ specify how the disturbance enters the state equation, and the output, respectively. Initially, it is assumed that the disturbance enters through the input so that

$$B_d(\rho) = B(\rho) \qquad C_d = 0.$$

An observer based on the augmented model (3.23) is used to estimate the disturbance with $Q_e = 100I_5$ and $R_e = 0.01I_2$[7] and the steady state input is redefined to be the solution to

$$\begin{bmatrix} I - A(\rho_s) & -B(\rho_s) \\ \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} B(\rho_k)\hat{d} \\ r \end{bmatrix},$$

where $\hat{d}$ is the disturbance estimate. The steady state vector x_s, on the other hand, is not changed as in this case the reference always remains within the admissible region[8]. An MPC law using the augmented model (3.23), with tuning matrices $Q = \text{diag}(1000, 1000, 0.1, 10, 0)$, $R = 2000$, considering the effect of input disturbances results in the closed-loop behavior shown in Figure 3.10. The addition of integral action through model augmentation together with the disturbance estimator yields vastly improved tracking performance, even though there is still steady-state error.

A close inspection of the time response plots leads to the conclusion that θ_2 is subject to an output disturbance given that the steady state value for $18 < t < 25$ is $\theta_2 \neq 0$ which is not physically possible. This disturbance is likely caused by an erroneous read of the sensor and it negatively impacts tracking performance. To capture this effect, a new disturbance model is included by defining

$$B_d(\rho) = \begin{bmatrix} B(\rho) & 0 \end{bmatrix} \quad C_d = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

which corresponds to a disturbance entering through the input and a disturbance directly acting on the output. As before, an observer is used for the augmented model in order to estimate the disturbances, the tuning parameters are $Q_e = 100I_6$ and $R_e = 0.001I_2$ and the MPC tuning matrices are set to $Q = \text{diag}(1000, 1000, 0.1, 10, 0, 0)$, $R = 2000$, $P = 1000Q$. The closed-loop response using this controller is shown in Figure Figure 3.11, where it can be seen that the steady state error is not present anymore and satisfactory results are achieved.

[7]The observer used is a quasi-LPV version of a Kalman filter, see Section 4.4.1 for more details.

[8]This means that the reference is such that the corresponding steady state input is always admissible

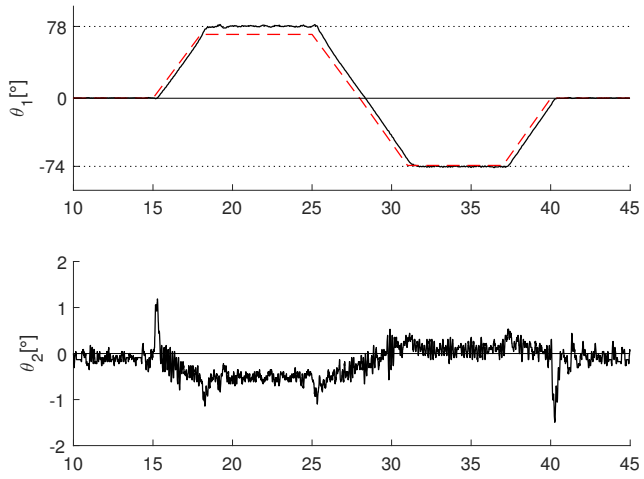


Figure 3.10: ADIP closed-loop response - no terminal constraint case using model augmented with integrating disturbance

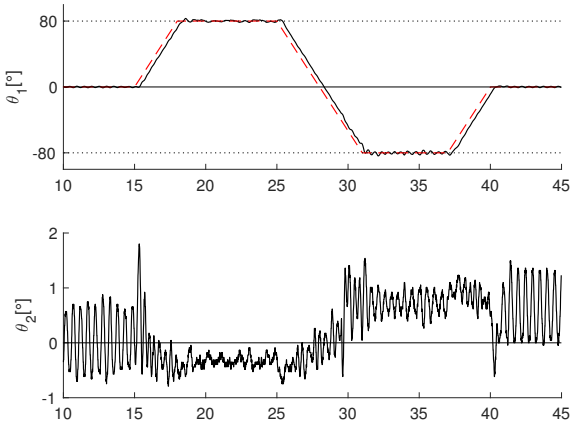


Figure 3.11: ADIP closed-loop response - no terminal constraint case using model augmented with 2 integrating disturbances

Numerical Efficiency

An important feature of the presented algorithm is its computational efficiency, indeed this application example, a mechatronic system with relatively fast dynamics, sampled at a fairly large sampling frequency (100 Hz), requires the computation of the control law within a time windows which is well within the sampling time. As measuring execution time during experimental implementation can be inaccurate and otherwise interrupt the real-time execution, timing characteristics of the approach are measured in simulation. These are summarized in Table 3.2,

where a distinction is made between the case with- and without terminal constraints, since this could potentially impact in execution time.[9]

Table 3.2: Execution times. ADIP example

	Average time	Max. time	St. deviation
Terminal constraints	519 μs	795 μs	53.0 μs
No terminal constraints	303 μs	948 μs	44.7 μs

3.6 Summary

Quasi-LPV Model Predictive Control (qLMPC) is a practical approach to nonlinear MPC which, within the same framework provides the tools necessary to establish stability guarantees and to efficiently implement the control law in real-time for plants with fast dynamics (in the milli- or sub-millisecond range). The former goal is achieved by exploiting the rich body of literature regarding stability of LPV and quasi-LPV systems and the steps undertaken to extend LTI results to the nonlinear realm; in the same way, results from LTI MPC literature are extended to quasi-LPV in order for them to be readily available for nonlinear systems. The latter objective, regarding efficient implementation, makes use of quasi-LPV modelling to recast the nonlinear optimization problem as a linear but time-varying one, making it easily solvable by efficient QP algorithms.

An important assumption made thus far is that a state space quasi-LPV model of the plant is available. A derivation of the model might in some cases prove to be complex and non-uniqueness of the LPV parameterization might lead to complications (cf. Example 2.3). Another limitation of the framework as presented throughout this chapter is that tracking problems are not as readily handled with stability guarantees, since deriving terminal ingredients would often require a partition of the state space, which is impractical for higher order systems. Both of these issues are handled in subsequent chapters in the context of velocity-based qLMPC (Chapter 4) and input-output qLMPC (Chapter 5).

[9]The terminal constraint case uses 2 iterations, whereas the terminal-constraint-free case only 1.

Chapter 4

Velocity-Based Nonlinear Model Predictive Control

Chapter 3 introduced the state space version of the quasi-Linear Model Predictive Control (qLMPC) framework. This approach enables predictive control to be applied to systems with relatively fast nonlinear dynamics. Stability conditions are derived by building upon well-known methods from the LPV literature to minimize conservatism. The provided stability analysis does, however, not tailor well to tracking problems and is in practice often carried out by partitioning the state space [121], making it only applicable to low-order systems. Forgoing stability guarantees makes the approach more flexible, however a parametrization of all equilibria is necessary; alternatively, a target selector problem can be solved which, given a desired set point, computes the corresponding steady state input and state vectors. In this chapter, a velocity form nonlinear MPC is proposed which addresses both of these issues, while also being simpler to implement. In the velocity space, all equilibria are mapped to the origin ($\dot{x} = 0, \dot{u} = 0$) so that a translation of the stabilizing terminal constraint is not necessary; furthermore, a parameterization of equilibria is likewise not needed, due to the same fact. The resulting velocity algorithm has built-in integral action (which results in offset-free control) without the need for disturbance estimators. An observer or numerical differentiation is however, often needed, since only rarely are velocities (and possibly accelerations) measured.

In order to obtain the model in velocity form, velocity-based linearization [73] is used. The resulting state-dependent model can readily be expressed as quasi-LPV, making it suitable for the methodology presented in this work. When compared to ad-hoc LPV parameterization, velocity-based linearization has the advantage that it is easy to compute and that it does not rely on the designer's experience and intuition. Jacobian linearization-based approaches, on the other hand, while easy to obtain, suffer from approximation errors and inconvenient affine state space representations. In contrast, velocity-based linearization yields a state dependent linear (quasi-LPV) model which is both exact and linear (as opposed to affine).

This chapter is structured as follows: Section 4.1 briefly discusses velocity algorithms and how they have been used in MPC. Section 4.2 discusses the velocity-based linearization in continuous and discrete-time and the resulting representation as quasi-LPV systems. Section 4.3 discusses the predictive control law and stability analysis is carried out. Finally, Section 4.4 illustrates the approach on an application, a 2-DOF robotic manipulator and a summary is given in Section 4.5.

4.1 Velocity algorithms

Velocity algorithms arose in the process industry, where they were used to avoid windup of digital PID controllers [60]. Indeed assuming a simplified discrete-time PID controller of the form[1]

$$u_k = K_P e_k + K_D(e_k - e_{k-1}) + K_I \sum_{i=0}^{k} e_i$$

a velocity implementation of the controller is obtained by multiplying both sides of the equation by $\Delta = 1 - q^{-1}$, yielding

$$u_k = u_{k-1} + K_P(e_k - e_{k-1}) + K_D(e_k - 2e_{k-1} + e_{k-2}) + K_I e_k.$$

A caveat is that in the derivation above, it is assumed that $e_k = 0$, $k \leq 0$; already in this simple example it is clear that initial conditions should be chosen appropriately otherwise an error would arise. This is clear when using velocity implementations, since the steady state information is lost in the process of differentiation or, as in this case, multiplication by Δ. This information can be included by choosing appropriate initial conditions.

In the context of LPV gain-scheduled control, it has been proved that using a velocity implementation of the gain-scheduled control leads to linearization family matching; this matching essentially eliminates hidden coupling [102], an effect present in classical gain-scheduling, which can cause instability of the closed-loop and which stems from the fact that the scheduling parameter is assumed constant during design, but it is time-varying when implementing the controller. Such a velocity implementation was proposed in [62] and has since become a powerful framework for gain scheduled control.

4.1.1 Velocity-form MPC

Velocity algorithms have been applied in the context of MPC to LTI systems. Analysis of the algorithm is treated thoroughly in the tutorial [122], while a stability result focusing on unreachable set points is given in [16]. In the LTI case, a velocity model can be easily obtained by observing that

$$x_{k+1} - x_k = Ax_k + Bu_k - (Ax_{k-1} + Bu_{k-1}),$$

as the model is LTI, i.e. the transition and input matrices are constant, one can factorize them to obtain an incremental model which satisfies the same dynamic equation as the non-incremental one, i.e.

$$\Delta x_{k+1} = A\Delta x_k + B\Delta u_k$$

state augmentation can be carried out to obtain a model more suitable for tracking, i.e.

$$\begin{bmatrix} y_{k+1} \\ \Delta x_{k+1} \end{bmatrix} = \begin{bmatrix} I & CA \\ 0 & A \end{bmatrix} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix} + \begin{bmatrix} CB \\ B \end{bmatrix} \Delta u_k \tag{4.1}$$

$$y_k = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix}.$$

[1]The simplified PID controller is used for exposition purposes; regardless of the complexity of the discretization used for the digital PID, the effect of the velocity algorithm is preserved.

A crucial characteristic of LTI models, which was emphasized above and that is needed to obtain a velocity model with this simple method, is the fact that the model matrices are time-invariant. This is obviously not true for (q)LPV, or otherwise nonlinear models so that this same procedure cannot be applied, at least not without introducing dynamic parameter dependence and substantially increasing the complexity of the model. In the nonlinear case, a velocity-form model can be obtained by use of the velocity-based linearization which is reviewed in the next section. In the literature there are a few examples of velocity algorithms in the context of MPC applied to nonlinear systems using velocity-based linearization, e.g. [43], [6] however no analysis is carried out, and is presented as an ad-hoc practical result, stability can therefore not be inferred. A stability result for velocity-based nonlinear MPC is given in Section 4.3.1

4.2 Velocity-based linearization

The Jacobian (otherwise known as first order series expansion-) linearization framework has several downsides. First and foremost it entails an approximation around an equilibrium point and is only valid in the vicinity of said equilibrium. The neighborhood around the equilibrium in which the linearization is valid is hard to characterize and may be excessively small. Another disadvantage is that this procedure yields an affine representation of the dynamics, which may complicate, or even preclude, its application for linear control design. In contrast, a velocity-based linearization [73] has the advantage that it is valid around any operating point (not only on equilibria), it is exact as it gives an exact dynamic equation in a space tangent to the original state space (the velocity space), and it is linear, as opposed to affine.

Consider a system governed by the following continuous-time nonlinear model[2]

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x). \end{aligned} \tag{4.2}$$

Definition 4.1 (Velocity-based linearization [73])**.** Assume $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^l$ in (4.2) are differentiable, then a velocity-based linearization is obtained by differentiating the equations of motion w.r.t. time, i.e.

$$\begin{aligned} \ddot{x} &= \nabla_x f(x,u)\dot{x} + \nabla_u f(x,u)\dot{u} \\ \dot{y} &= \nabla_x h(x)\dot{x}. \end{aligned} \tag{4.3}$$

The model (4.3) is linear in the (derivative of) the states and inputs and the state-space matrices are now state/inputs dependent. As mentioned earlier, the velocity-linearization must include appropriate selection of initial conditions $x(0), u(0), y(0) = h(x(0)), \dot{x}(0) = f(x(0), u(0))$, for it to be meaningful, as this information is lost in the differentiation process. Alternatively, the steady-state information on the output can be recovered by state augmentation. Indeed define the augmented state as $[y^\top \quad \dot{x}^\top]^\top$, then a velocity-form state-dependent linear model is given

[2]for simplicity it is assumed that there is no immediate effect from input to output, i.e. there is no feedthrough

by

$$\begin{aligned}\begin{bmatrix}\dot{y}\\ \ddot{x}\end{bmatrix} &= \begin{bmatrix}0 & \nabla_x h(x)\\ 0 & \nabla_x f(x,u)\end{bmatrix}\begin{bmatrix}y\\ \dot{x}\end{bmatrix} + \begin{bmatrix}0\\ \nabla_u f(x,u)\end{bmatrix}\dot{u}\\ y &= \begin{bmatrix}I & 0\end{bmatrix}\begin{bmatrix}y\\ \dot{x}\end{bmatrix}.\end{aligned} \tag{4.4}$$

By defining $\rho_i = e_j \begin{bmatrix}x\\ u\end{bmatrix}$, $i = 1, ..., n_\rho$, where e_j is the j^{th} row of an identity matrix, this state-dependent linear system can be expressed as a qLPV system. In particular $\rho = \begin{bmatrix}x\\ u\end{bmatrix}$ if there is no linear subsystem in $f(\cdot)$, i.e. if $f(\cdot)$ is nonlinear in every state and input so that $n_\rho = n + m$.[3]

The augmented model (4.4) should then be discretized for it to be used within the framework presented in the previous chapter. There are several methods to do so in the LPV literature, see e.g. [117], one of them which is particularly tailored to models obtained via velocity-based linearization is presented in [116]. The chosen method notwithstanding, a discrete-time version of the model can be expressed as

$$\begin{aligned}\begin{bmatrix}y_{k+1}\\ \dot{x}_{k+1}\end{bmatrix} &= A(\rho_k)\begin{bmatrix}y_k\\ \dot{x}_k\end{bmatrix} + B(\rho_k)\Delta u_k\\ y_k &= \begin{bmatrix}I & 0\end{bmatrix}\begin{bmatrix}y_k\\ \dot{x}_k\end{bmatrix}.\end{aligned} \tag{4.5}$$

Note that the state vector in equation (4.5) is the sampled version of the continuous-time state in (4.4): with a slight abuse of notation the dot ($\dot{}$) notation is used, not as a short-hand for the time derivative operator $\frac{d}{dt}$ but rather as the sampled rate of change of the un-dotted signal, i.e. $\dot{x}_k = \frac{d}{dt}x(t)\big|_{t=kT_s}$, where T_s is the sampling time.

4.2.1 Discrete-time nonlinear models

Assume now that a discrete-time nonlinear model is given

$$\begin{aligned}x_{k+1} &= \tilde{f}(x_k, u_k)\\ y_k &= \tilde{h}(x_k),\end{aligned} \tag{4.6}$$

the discussion in the previous section assumed a continuous-time nonlinear model (4.2) is given; linearity of the resulting velocity-based linearization is thus a consequence of the chain rule when obtaining the time-derivative of the state and output equations. In the discrete-time case the difference operator does not share this useful property and linearity is therefore not immediate

[3]This can be seen as a disadvantage of velocity-based qLPV parameterization, whereas an ad hoc parameterization could result in $n_\rho < n$, often all states and inputs appear as parameters when performing a velocity-based linearization. In the present context this does not represent a big disadvantage, however, in the LPV gain-scheduled controller synthesis case where LMIs are solved, often by gridding the parameter space, having a large number of parameters could render the problem intractable

by applying a difference operation to (4.6). There is, however, a result in differential calculus which can be used in this case, the following lemma is the multivariable extension of the Mean Value Theorem as proposed in [132]:

Lemma 4.1 (Multivariable Mean Value Theorem (MMVT) [132])**.** Let $g(x) : \mathbb{R}^n \to \mathbb{R}^q$, assume that g is differentiable on $\text{Co}(a, b)$[a], then there exist constant vectors $c_i \in \text{Co}(a, b)$, $c_i \neq a$, $c_i \neq b$ $i = 1, ..., q$ such that:

$$g(a) - g(b) = \left(\sum_{i,j=1}^{q,n} e_q(i) e_n(j)^\top \frac{\partial g_i}{\partial x_j}(c_i) \right) (a - b)$$

where $e_s(i)$ is the i^{th} column of I_s.

[a]Recall $\text{Co}(a, b)$ is the convex hull of vectors a, b, Definition 2.12

In light of Lemma 4.1 a velocity (or incremental) form of (4.6) is given by

$$\begin{aligned} \Delta x_{k+1} &= \nabla_x \tilde{f}(\tilde{x}, \tilde{u}) \Delta x_k + \nabla_u \tilde{f}(\tilde{x}, \tilde{u}) \Delta u_k \\ \Delta y_k &= \nabla_x \tilde{h}(\tilde{x}) \Delta x_k \end{aligned}$$

where $\tilde{x} \in \text{Co}(x_k, x_{k+1})$, $\tilde{u} \in \text{Co}(u_k, u_{k+1})$, which can in practice not be determined; an approximation can thus be carried out by assuming $\tilde{x} = x_k$, $\tilde{u} = u_k$, yielding an incremental model

$$\begin{aligned} \Delta x_{k+1} &\approx \nabla_x \tilde{f}(x_k, u_k) \Delta x_k + \nabla_u \tilde{f}(x_k, u_k) \Delta u_k \\ \Delta y_k &\approx \nabla_x \tilde{h}(x_k) \Delta x_k \end{aligned} \tag{4.7}$$

which together with the initial conditions $x(0), u(0), y(0) = \tilde{h}(x(0)), \Delta x(0) = \tilde{f}(x(0), u(0))$ describes the dynamics of (4.6) as a state-dependent linear model. As in the continuous-time case, state augmentation can be carried out to include output steady-state information and to have a model with absolute (as opposed to incremental) output. This can be done in two different ways: if the output equation is linear, i.e. $y_k = \bar{h} x_k$, the following derivation can be obtained

$$\begin{aligned} y_{k+1} = y_k + \Delta y_{k+1} &= y_k + \bar{h} \Delta x_{k+1} \\ &= y_k + \bar{h}(\nabla_x \tilde{f}(x_k, u_k) \Delta x_k + \nabla_u \tilde{f}(x_k, u_k) \Delta u_k) \end{aligned}$$

the augmented incremental model can thus be written, analogously to its LTI counterpart in Equation (4.1), as

$$\begin{aligned}\begin{bmatrix} y_{k+1} \\ \Delta x_{k+1} \end{bmatrix} &\approx \underbrace{\begin{bmatrix} I & \bar{h}\nabla_x \tilde{f}(x_k, u_k) \\ 0 & \nabla_x \tilde{f}(x_k, u_k) \end{bmatrix}}_{A(\rho_k)} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix} + \underbrace{\begin{bmatrix} \bar{h}\nabla_u \tilde{f}(x_k, u_k) \\ \nabla_u \tilde{f}(x_k, u_k) \end{bmatrix}}_{B(\rho_k)} \Delta u_k \\ y_k &= \underbrace{\begin{bmatrix} I & 0 \end{bmatrix}}_{C} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix},\end{aligned} \tag{4.8}$$

which can, as before, be expressed as a quasi-LPV model by defining $\rho_i(k) = e_j \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}$, $i = 1, ..., n_\rho$. In the case the output equation is nonlinear, this would introduce dynamic dependence on the $A(\cdot)$ and $B(\cdot)$ matrices. This issue can be circumvented by writing the model as follows instead

$$\begin{aligned}\begin{bmatrix} y_k \\ \Delta x_{k+1} \end{bmatrix} &\approx \underbrace{\begin{bmatrix} I & \nabla_x \tilde{h}(x_k) \\ 0 & \nabla_x \tilde{f}(x_k, u_k) \end{bmatrix}}_{A(\rho_k)} \begin{bmatrix} y_{k-1} \\ \Delta x_k \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \nabla_u \tilde{f}(x_k, u_k) \end{bmatrix}}_{B(\rho_k)} \Delta u_k \\ y_k &= \underbrace{\begin{bmatrix} I & \nabla_x \tilde{h}(x_k) \end{bmatrix}}_{C(\rho_k)} \begin{bmatrix} y_{k-1} \\ \Delta x_k \end{bmatrix}.\end{aligned} \tag{4.9}$$

Note that state augmentation is done using the backward-shifted output; as expected the output equation becomes parameter dependent.

Both (4.4) and (4.8) (or (4.9)) are velocity-based state dependent (qLPV) models derived from a nonlinear model, only difference being if the original nonlinear model is continuous-time, or discrete-time, respectively. In order to generalize notation and treat both cases within a unifying framework, the notation δx_k is used for the remainder of this chapter, changing its meaning accordingly:

A quasi-LPV model obtained using velocity-based linearization is given by

$$\begin{aligned}\begin{bmatrix} y_{k+1} \\ \delta x_{k+1} \end{bmatrix} &= A(\rho_k) \begin{bmatrix} y_k \\ \delta x_k \end{bmatrix} + B(\rho_k)\Delta u_k \\ y_k &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} y_k \\ \delta x_k \end{bmatrix}.\end{aligned} \tag{4.10}$$

where

$$\delta x_k = \begin{cases} \dot{x}_k & \text{if original model is continuous-time (Eq. (4.2))} \\ \Delta x_k & \text{if original model is discrete-time (Eq. (4.6)).} \end{cases}$$

correspondingly, $A(\rho_k)$ and $B(\rho_k)$ are given in equation (4.5) and equations (4.8) and (4.9), respectively.

4.3 Predictive Controller

Consider a nonlinear model of the form (4.2) or (4.6) and its corresponding augmented velocity-based linearization (4.10). Assume that both the state and input are subject to constraints $x \in \mathcal{X}$, $u \in \mathcal{U}$, the goal is to design an optimal feedback control law which tracks a given constant reference signal y_{sp} while respecting constraints.

Throughout this chapter, the following assumptions are made

A.1 Model (4.10) is stabilizable $\forall \rho \in \mathcal{P}$

A.2 $l \leq m$, that is, the number of tracking outputs is less than or equal to the number of inputs.

A.3 The system has a continuous locus of forced equilibria (i.e. the system does not have isolated forced equilibria).

Note that Assumption A.1 is equivalent to a local stabilizability condition of $f(x,u)$ ($\tilde{f}(x,u)$), i.e. the linearization of $f(x,u)$ ($\tilde{f}(x,u)$) at all equilibria $0 = f(\bar{x},\bar{u})$ ($\bar{x} = \tilde{f}(\bar{x},\bar{u})$) $(\bar{x},\bar{u}) \in \mathcal{X}\times\mathcal{U}$ is stabilizable. Assumption A.3 might seem restrictive, however it is implicitly made in most standard tracking MPC algorithms, when unreachable reference set points are considered.

In order to characterize admissible set points, the set of reachable steady-states is defined as

$$\mathcal{R}_{\mathcal{X}} = \Bigg\{ x_s \mid \exists T_f \in \mathbb{Z}_{\geq 0}, \exists (u(k),...,u(k+T_f)), : x_a(k+1) = A_d(\rho(k))x_a(k) + B_d(\rho(k))\Delta u(k),$$

$$x(T_f) = x_s,\ \delta x(T_f) = 0, u(k+j) = u(k-1) + \sum_{i=0}^{j} \Delta u(k+i) \in \mathcal{U},\ x(k+j) \in \mathcal{X}\ \forall j \in [0\ \ T_f] \Bigg\}$$

correspondingly, the set of steady states reachable in N steps as

$$\mathcal{R}_{\mathcal{X}}^{N} = \Bigg\{ x_s \mid \exists (u(k),...,u(k+N)) : x_a(k+1) = A_d(\rho(k))x_a(k) + B_d(\rho(k))\Delta u(k),$$

$$x(N) = x_s,\ \delta x(N) = 0, u(k+j) = u(k-1) + \sum_{i=0}^{j} \Delta u(k+i) \in \mathcal{U},\ x(k+j) \in \mathcal{X}\ \forall j \in [0\ \ N] \Bigg\}.$$

The projections of these sets onto the output can be defined as $\mathcal{R}_{\mathcal{Y}} = \text{Proj}_{\mathbb{R}^l}(\mathcal{R}_{\mathcal{X}})$, $\mathcal{R}_{\mathcal{Y}}^{N} = \text{Proj}_{\mathbb{R}^l}(\mathcal{R}_{\mathcal{X}}^{N})$. Furthermore, the following sets are defined

$$\mathfrak{R}_{\mathcal{Y}} = \Bigg\{ \tilde{y} \in \mathcal{R}_{\mathcal{Y}} \mid \forall\, \lambda \in [0\ 1]\ \lambda y_k + (1-\lambda)\tilde{y} \in \mathcal{R}_{\mathcal{Y}} \Bigg\},$$

$$\mathfrak{R}_{\mathcal{Y}}^{N} = \Bigg\{ \tilde{y} \in \mathcal{R}_{\mathcal{Y}}^{N} \mid \forall\, \lambda \in [0\ 1]\ \lambda y_k + (1-\lambda)\tilde{y} \in \mathcal{R}_{\mathcal{Y}}^{N} \Bigg\}.$$

Note that these sets can be understood as the set of reachable steady state outputs for which there exists a line of sight which is itself a set of admissible steady state outputs. For illustration purposes, an example of these sets is depicted in Figure 4.1.

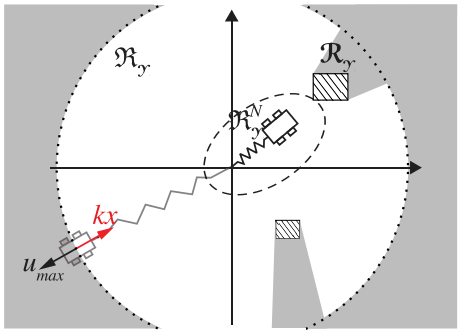


Figure 4.1: Characterization of reachable sets. The shaded regions represent non-admissible areas due to state and input constraints. The reachable set $\mathcal{R}_\mathcal{Y}$ corresponds to the entire circle (excluding obstacles), whereas $\mathfrak{R}_\mathcal{Y}$ is the white area.

The stage cost to be considered throughout this chapter is

$$\ell(y, y_{sp}, \delta x, \Delta u) = ||y - y_{sp}||^2_{Q_1} + ||\delta x||^2_{Q_2} + ||\Delta u||^2_R \tag{4.11}$$

yielding the cost function

$$J_k(Y_k, y_{sp}, \delta X_k, \Delta U_k) = \sum_{i=0}^{N-1} \ell(y_{k+i}, y_{sp}, \delta x_{k+i}, \Delta u_{k+i}) + ||y_s - y_{sp}||^2_P \tag{4.12}$$

where $Y_k = [y_k^\top \dots y_{k+N-1}^\top]^\top$, $\delta X_k = [\delta x_k^\top \dots \delta x_{k+N-1}^\top]^\top$, $\Delta U_k = [\Delta u_k^\top \dots \Delta u_{k+N-1}^\top]^\top$.

It was noted in [97] that a stage cost penalizing the error w.r.t. the real set point (y_{sp} in this case) is more meaningful than penalizing the error to the artificial set point (y_s) as it is standard, see e.g. [75][16][45], as the latter can change due to transient disturbances resulting in excessive control action and potential loss in performance.

The optimization problem to be solved online is given by

$$\min_{\Delta u} J_k(Y_k, y_{sp}, \delta X_k, \Delta U_k) \tag{4.13a}$$

s.t.

$$\begin{bmatrix} y_{i+1} \\ \delta x_{i+1} \end{bmatrix} = A(\rho_i) \begin{bmatrix} y_i \\ \delta x_i \end{bmatrix} + B(\rho_i)\Delta u_i \tag{4.13b}$$

$$x_{k+i} \in \mathcal{X} \tag{4.13c}$$

$$u_{k+i} = u_{k-1} + \Sigma_{j=0}^{i} \Delta u_{k+j} \in \mathcal{U} \qquad i = [k \;\; k+N-1] \tag{4.13d}$$

$$y_{k+N} = y_s \tag{4.13e}$$

$$\delta x_{k+N} = 0 \tag{4.13f}$$

Note that constraint (4.13f) guarantees that the terminal state corresponding to y_s is a steady state, as it is often assumed in set point tracking strategies (cf. [45], [75]). Note also that constraint (4.13e) can be eliminated from the optimization problem by substituting it in the cost function, the reason it is included is that it will prove useful for stability analysis.

Define the optimal sequence of input increments from the solution of problem (4.13) as $\Delta U^* = (\Delta u_k^*, \Delta u_{k+1}^*, ..., \Delta u_{k+N-1}^*)$, then applying the receding horizon principle defines the implicit control law as $\kappa(y_k, \delta x_k, y_{sp}) = u_{k-1} + \Delta u_k^*$.

4.3.1 Stability of the velocity algorithm for nonlinear MPC

As was briefly mentioned earlier, particularly for tracking problems, a velocity algorithm is more meaningful than the standard implementation of MPC since all equilibria are mapped to the origin $(\delta x = 0, \Delta u = 0)$; because of this, compared to other methods (e.g. [45]), a parameterization of all equilibria, which might prove to be very complex for nonlinear systems, is not necessary.[4] The optimization problem as proposed in (4.13) does not require the calculation of steady state values for the state vector and input, however, the terminal constraint (4.13e) still necessitates the explicit computation of fictitious intermediate set points y_s. This section presents a stability result which exploits the velocity form in order to relinquish this constraint.

Theorem 4.1. Let Assumptions A.1, A.2, A.3 hold, and let the terminal offset penalty $P = \alpha Q_1$ for some $\alpha > 0$. Given a set point y_{sp}, the control law $\kappa(y_k, \delta x_k, y_{sp})$ derived from the solution of optimization problem (4.13) starting from a feasible state $[y^\top \ \delta x^\top]^\top$ stabilizes the system (4.10), is recursively feasible and makes the output converge to one of the following

1. y_{sp} if $y_{sp} \in \mathfrak{R}_{\mathcal{Y}}$
2. $\tilde{y}$ if $y_{sp} \notin \mathfrak{R}_{\mathcal{Y}}$, where
$$\tilde{y} = \arg\min_{y \in \mathfrak{R}_{\mathcal{Y}}} ||y - y_{sp}||_P^2$$

Proof. The proof is divided into three steps: Step I establishes recursive feasibility, Step II proves the monotonicity of the cost function and hence stability and Step III proves that the only equilibria compatible with the optimization are those listed in items (1)-(2) of Theorem 4.1. Step I and II are standard in the MPC literature and are shown here – with appropriate modifications tailored to the velocity algorithm – for completeness. Step III closely follows [45][75] where convergence to an optimal steady state is proved using convexity arguments.

Step I Assuming feasibility of the initial state, the optimal input increment sequence from the solution of problem (4.13) is $\Delta U_k^* = (\Delta u_k^*, \Delta u_{k+1}^*, ..., \Delta u_{k+N-1}^*)$; a feasible, possibly suboptimal sequence at $k+1$ is given by

[4] Equilibrium parameterization refers to the problem of finding the steady state values of the state vector and input, i.e. x_s and u_s, as a function of the set point y_{sp}. In the previous chapter this was done via a target selector, c.f Section 3.3.

$$\Delta U_{k+1} = (\Delta u^*_{k+1}, \Delta u^*_{k+2}, ..., \Delta u^*_{k+N-1}, 0)$$

obtained by shifting the solution from time step k and appending $\Delta u_{k+N} = 0$ at the end, i.e. setting $u_{k+N} = u_{k+N-1}$. Given constraint (4.13f), the state $x(k+N)$ is a forced equilibrium, therefore keeping the same input keeps the system in steady state and is feasible.

Step II Assume the steady state problem

$$\tilde{y}_k = \arg\min_{y \in \mathfrak{R}^N_{\mathcal{Y}}} ||y - y_{sp}||^2_{Q_1+P} \tag{4.14}$$

is solved online at each time step (the subscript $_k$ indicates it is the optimal solution computed at time k) and assume that y_s in (4.12) is set to $y_s = \tilde{y}$. Note that this is feasible, since the steady state problem is constrained to $\mathfrak{R}^N_{\mathcal{Y}}$. Clearly at timestep $k+1$ a feasible (possibly suboptimal) solution to (4.14) is $\tilde{y}_{k+1} = \tilde{y}_k$, given that $\mathfrak{R}^{N-1}_{\mathcal{Y}} \subseteq \mathfrak{R}^N_{\mathcal{Y}}$. Considering this and the feasible input sequence for timestep $k+1$ given in Step I, a feasible value for the cost function at $k+1$ is given by

$$J_{k+1} = J_k - ||y_k - y_{sp}||^2_{Q_1} - |\delta x_k||^2_{Q_2} - ||\Delta u_k||^2_R - ||\tilde{y}_k - y_{sp}||^2_P + ||y_N - y_{sp}||^2_{Q_1} + ||\tilde{y}_k - y_{sp}||^2_P$$

where $y_N = \tilde{y}_k$. Given optimality of (4.14) from time step k and $P = \alpha Q_1$, it holds that $||y_N - y_{sp}||^2_{Q_1} + ||\tilde{y}_k - y_{sp}||^2_P - ||y_0 - y_{sp}||^2_{Q_1} - ||\tilde{y}_k - y_{sp}||^2_P \leq 0$ with equality only if $y_0 = \tilde{y}_k$. It therefore follows that $J_{k+1} \leq J_k$, so the cost function is a decaying sequence, and the closed-loop is stable.

Observe now that the solution of problem (4.13) is not changed by substituting the terminal constraint (4.13e) into the cost function. Doing this eliminates the variable y_s from the problem, thus avoiding the online solution of (4.14); it also turns the offset function into a terminal cost.

Step III This step is based on convexity arguments; by contradiction it is shown that convergence to a feasible steady state that is not optimal in the sense of item 2 in Theorem 4.1 is not possible. Assume y^* is the optimal steady state output such that the cost function (4.12) is minimized but that there exists a different steady state $\tilde{y}$ such that $||\tilde{y} - y_{sp}||^2_P < ||y^* - y_{sp}||^2_P$. Considering the properties of $\mathfrak{R}_{\mathcal{Y}}$ and Assumption A.3, there exists a $\lambda \in [0,\ 1]$ such that $\bar{y} = \lambda y^* + (1-\lambda)\tilde{y}$ is a feasible steady state.

By optimality of y^*

$$J^*_k = N||y^* - y_{sp}||^2_{Q_1} + ||y^* - y_{sp}||^2_P \leq J_k = \sum_{i=0}^{N-1} \left(||y_i - y_{sp}||^2_{Q_1} + ||\delta x_i||^2_{Q_2} + ||\Delta u_i||^2_R \right) + ||\bar{y} - y_{sp}||^2_P$$

Similar to [45] define $W(Y_k, y_{sp}, \delta X_k, \Delta U_k, y_s, \lambda, y^*, \tilde{y}) = J_k(Y_k, y_{sp}, \delta X_k, \Delta U_k, \bar{y})$ and note that $W(Y_k, y_{sp}, \delta X_k, \Delta U_k, y_s, 1, y^*, \tilde{y}) = J^*_k$, where $\lambda = 1$ has been substituted. Take the partial derivative of W w.r.t λ

$$\frac{\partial W}{\partial \lambda} = g(\lambda)^\top (y^* - \tilde{y})$$

where $g(\lambda)^\top = \nabla_\lambda ||\bar{y} - y_{sp}||_P^2$ is the gradient of the terminal (offset) cost. From convexity of $||\bar{y} - y_{sp}||_P^2$ we have

$$g(1)^\top (\tilde{y} - y^*) < ||\tilde{y} - y_{sp}||_P^2 - ||y^* - y_{sp}||_P^2$$

or

$$g(1)^\top (y^* - \tilde{y}) > ||y^* - y_{sp}||_P^2 - ||\tilde{y} - y_{sp}||_P^2 > 0 \tag{4.15}$$

where the second inequality in (4.15) follows from the initial assumption, namely $||\tilde{y} - y_{sp}||_P^2 < ||y^* - y_{sp}||_P^2$ and $g(1) = \nabla_\lambda ||y^* - y_{sp}||_P^2$.

Note that the left hand side of (4.15) is equal to $\partial W / \partial \lambda|_{\lambda=1}$. As this is positive and from continuity of W it follows that there exist $\lambda < 1$ such that W and hence J_k is less than J_k^*, which contradicts the optimality of J_k^*. Therefore, the algorithm converges to the optimal steady state according to items 1-2 in Theorem 4.1.

□

Considering the last remark on Step II of the proof above, an equivalent modified optimization problem is

$$\min_{\Delta u} \sum_{i=0}^{N-1} \ell(y_{k+i}, y_{sp}, \delta x_{k+i}, \Delta u_{k+i}) + ||y_{k+N} - y_{sp}||_P^2 \tag{4.16a}$$

s.t.

$$\begin{bmatrix} y_{i+1} \\ \delta x_{i+1} \end{bmatrix} = A(\rho_i) \begin{bmatrix} y_i \\ \delta x_i \end{bmatrix} + B(\rho_i) \Delta u_i \tag{4.16b}$$

$$x_{k+i} \in \mathcal{X} \tag{4.16c}$$

$$u_{k+i} = u_{k-1} + \Sigma_{j=0}^{i} \Delta u_{k+j} \in \mathcal{U} \qquad i = [k \;\; k+N-1] \tag{4.16d}$$

$$\delta x_{k+N} = 0 \tag{4.16e}$$

where the equality constraint (4.13e) was eliminated from the optimization problem. This is a consequence of the fact that in the velocity-form MPC, the fictitious intermediate set points need not be explicitly computed, it is only required that they exist, as the terminal constraint (4.16e) is still considered.

Remark 4.1. *The proposed velocity algorithm need not be used in conjunction with qLMPC, the resulting nonlinear optimization problem can be solved by any means without affecting the stability result (assuming the solution is exact). In this case the equality constraint corresponding to the model* (4.16b) *can be replaced by their state dependent counterparts* (4.4) *(after discretization) or* (4.8)*. However, the structure of the problem is such that applying qLMPC is straightforward as the state dependent system can readily be modelled as quasi-LPV.*

4.4 Application Example: Velocity-based qLMPC of a 2-DOF Robotic Manipulator

This section presents an experimental implementation of velocity-form qLMPC to control a 2-DOF robotic manipulator. The system is subject to both input and state constraint via physical limitations and, to demonstrate the flexibility of the approach, strongly non-convex state constraints are introduced in the form of obstacles. The goal is thus to track a desired set point by computing feasible trajectories which avoid collisions. The experimental platform is a CRS A465 6-degree of freedom robotic manipulator (Figure 4.2a) of which only the degrees of freedom corresponding to θ_1 and θ_2 as shown in Figure 4.2b are used in this example.

(a) (b)

Figure 4.2: 2-DOF Robotic Manipulator

A nonlinear model of the robotic manipulator can be expressed as a second order multivariable differential equation

$$M(\theta(t))\ddot{\theta}(t) + c(\theta(t), \dot{\theta}(t)) + g(\theta(t)) + \tau_f(\dot{\theta}(t)) = \tau(t) \tag{4.17}$$

where $\theta = [\theta_1 \;\; \theta_2]^\top$ are the joint angles and $\dot{\theta}, \ddot{\theta}$ are the corresponding velocities and accelerations, respectively; M is the inertia matrix, c is the Coriolis force vector, g is the gravity vector, τ is the applied torque and τ_f is the friction torque which is modelled solely as viscous. Note that the nonlinear model (4.17) is quite general and can be used to model almost any mechanical system. For the case of the 2-DOF robot, the model matrices are given in Equation (4.18) where $b_1, \ldots b_9$ are lumped parameters, the values of which are given in Table 4.1.

$$M(\theta) = \begin{bmatrix} b_1 + b_2 + 2b_3\cos(\theta_2) & b_2 + b_3\cos(\theta_2) \\ b_7 - b_8 + b_3\cos(\theta_2) & b_7 \end{bmatrix}, \quad c(\theta) = \begin{bmatrix} -b_3\dot{\theta}_2^2\sin(\theta_2) - 2b_3\dot{\theta}_1\dot{\theta}_2\sin(\theta_2) \\ b_3{\dot{\theta}_1}^2\sin(\theta_2) \end{bmatrix},$$
$$g(\theta) = \begin{bmatrix} -b_4\sin(\theta_1+\theta_2) - b_5\sin(\theta_1) \\ -b_4\sin(\theta_1+\theta_2) \end{bmatrix} \qquad \tau_f = \begin{bmatrix} b_6\dot{\theta}_1 \\ b_9\dot{\theta}_2 \end{bmatrix}. \tag{4.18}$$

Defining the state vector $x = \begin{bmatrix}\theta^\top & \dot{\theta}^\top\end{bmatrix}^\top$, a velocity-form quasi-LPV model of (4.17) can be obtained by expressing the nonlinear model as a first order differential equation

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = \underbrace{\left[\begin{array}{c}\dot{\theta}_1\\ \dot{\theta}_2\\ \hline -M^{-1}(\theta)(c(\theta,\dot{\theta})+g(\theta)+\tau_f(\dot{\theta})-\tau)\end{array}\right]}_{f(x,\tau)},$$

defining the tracking output $y = Cx = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^\top$ and applying velocity-based linearization with state augmentation, viz. (4.3) yielding the continuous-time qLPV model

$$\begin{bmatrix}\dot{y}\\ \ddot{x}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & C\\ 0 & \nabla f_x(\rho)\end{bmatrix}}_{A_c(\rho)}\begin{bmatrix}y\\ \dot{x}\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ \nabla f_\tau(\rho)\end{bmatrix}}_{B_c(\rho)}\dot{\tau}. \tag{4.19}$$

where $\rho = \begin{bmatrix}\rho_1 & \rho_2 & \rho_3 & \rho_4 & \rho_5 & \rho_6\end{bmatrix}^\top = \begin{bmatrix}q_1 & q_2 & \dot{q}_1 & \dot{q}_2 & \tau_1 & \tau_2\end{bmatrix}^\top$. Finally the model is discretized, for simplicity an Euler discretization is chosen for both the state and the input, leading to the discrete-time LPV model

$$x_{a,k+1} = A(\rho_k)x_{a,k} + B(\rho_k)\Delta\tau_k \tag{4.20}$$

where $x_a = \begin{bmatrix}y^\top & \dot{x}^\top\end{bmatrix}^\top$, $A(\rho_k) = I + T_s A_c(\rho_k)$, $B(\rho_k) = B_c(\rho_k)$ and $\Delta\tau_k = \tau_k - \tau_{k-1}$, and a sampling time of $T_s = 0.01$s is chosen.

4.4.1 Observer design

The use of velocity algorithms often comes at the cost of having to employ either numerical differentiation (using e.g. differential filters) or a state estimator. In this application, the measured output is $y = [\theta_1 \;\; \theta_2]^\top$ so that obtaining accelerations by means of two consecutive numerical differentiations would potentially lead to high sensitivity to measurement noise, which could in turn excite harmful high frequency vibrations. For this reason, the latter option is a better choice and a quasi-LPV Kalman Filter is used. This filter is essentially an Extended Kalman Filter (EKF) but the prediction for both the state and the covariance is done using the velocity qLPV model (4.20), as opposed to EKF where the state prediction is made using the nonlinear model (4.17) and the covariance prediction using its linearization. Nevertheless the

Table 4.1: Estimated inertial parameters (non-SI units), [54] and lengths

Parameter	Value	Parameter	Value
b_1	0.0715	b_7	0.0749
b_2	0.0058	b_8	0.0705
b_3	0.0114	b_9	1.1261
b_4	0.3264	l_1	0.3048 m
b_5	0.3957	l_2	0.4064 m
b_6	0.6254		

same prediction-correction structure is used, the quasi-LPV estimation can be summarized as follows (based on EKF from [112]):

Predict:

$$\hat{x}^a_{k|k-1} = A(\hat{\rho}_{k-1})\hat{x}^a_{k-1|k-1} + B(\hat{\rho}_{k-1})\Delta u_{k-1}$$
$$P_{k|k-1} = A(\hat{\rho}_{k-1})P_{k-1|k-1}A(\hat{\rho}_{k-1}) + Q_e$$

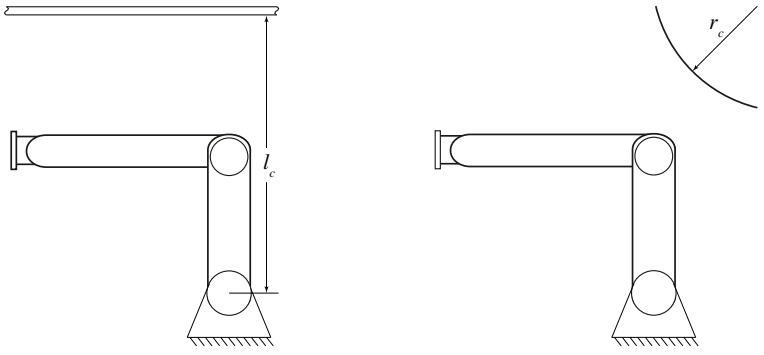
Update:

$$\tilde{y}_k = y_k - C\hat{x}_{k|k-1}$$
$$K_k = P_{k|k-1}C^\top(CP_{k|k-1}C^\top + R_e)^{-1}$$
$$\hat{x}^a_{k|k} = \hat{x}^a_{k|k-1} + K_k\tilde{y}_k$$
$$P_{k|k} = (I - K_kC)P_{k|k-1}$$

where Q_e, R_e are the process and measurement noise covariances, often used as tuning parameters for estimation problems. In this case they are set to $Q_e = I_6$, $R_e = 0.001I_2$, given that the encoders that measure joint angles are relatively noise-free. Note that this observer uses a simple 1-step-ahead prediction and the usual correction step (i.e. it is not a moving-horizon estimator).

4.4.2 Nonlinear state constraints

Nonlinear constraints can be considered in the qLMPC framework by expressing them in qLPV form as described in Section 3.4. In this application example, the challenging task of obstacle avoidance is investigated using two test cases: a horizontal short ceiling above the manipulator and a circular obstacle (which could represent an over-bounding of any arbitrary obstacle) as shown in Figure 4.3. How to obtain suitable qLPV representations of the constraints is presented next.



(a) (b)

Figure 4.3: Obstacles considered for the experiments

Ceiling

Assume that only the end effector is constrained to avoid the obstacle (as opposed to the entire geometry of the manipulator), if $l_c > l_1$ this is sufficient to keep the robot away from the ceiling; the converse case lacks practical interest as it would preclude existence of a trajectory around the obstacle. The nonlinear constraint is thus given by

$$h(\theta) = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \leq l_c. \tag{4.21}$$

As discussed in Section 3.4, a linearization can be carried out to find a qLPV representation:

$$\underbrace{l_1 \cos(\bar{\theta}_1) + l_2 \cos(\bar{\theta}_1 + \bar{\theta}_2)}_{h(\bar{\theta})} + \underbrace{\begin{bmatrix} -l_1 \sin(\bar{\theta}_1) - l_2 \sin(\bar{\theta}_1 + \bar{\theta}_2) & -l_2 \sin(\bar{\theta}_1 + \bar{\theta}_2) \end{bmatrix}}_{\nabla h_\theta(\bar{\theta})} \begin{bmatrix} \theta_1 - \bar{\theta}_1 \\ \theta_2 - \bar{\theta}_2 \end{bmatrix} \leq l_c$$

By defining the constraint scheduling parameter vector as the moving linearization point $\rho^c = \bar{\theta}$, a qLPV representation can be written as

$$\underbrace{\left[\begin{array}{c|cccc} \nabla h_\theta(\rho^c) & 0 & 0 & 0 & 0 \end{array}\right]}_{C_c(\rho^c)} \begin{bmatrix} y \\ \dot{x} \end{bmatrix} \leq \underbrace{l_c + \nabla h_\theta(\rho^c)\rho^c - h(\rho^c)}_{\tilde{b}_c(\rho^c)} \tag{4.22}$$

For the experiment the distance from the base of the manipulator to the ceiling is set to $l_c = 0.55$ m.

Circle

As before, assume that only the end effector is constrained away from the obstacle, depending on the distance to the obstacle and its size, this might prove inadequate, however the purpose of this example is to highlight nonlinear constraint handling and not to provide a comprehensive control strategy for robotic manipulators tailored for obstacle avoidance. Under consideration of this disclaimer, the nonlinear constraint is given by

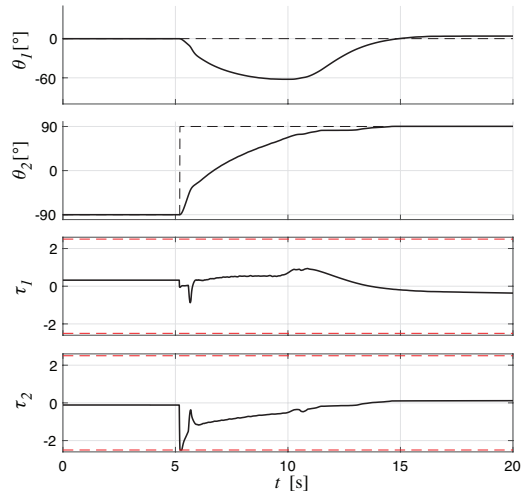
$$-(l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) - c_x)^2 - (l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) - c_y)^2 \leq -r_c^2,$$

where (c_x, c_y) are the Cartesian coordinates of the center of the circle w.r.t the base of the robot and r_c is the radius of the circle. Proceeding as before, the nonlinear constraint can be written in the form (4.22) where

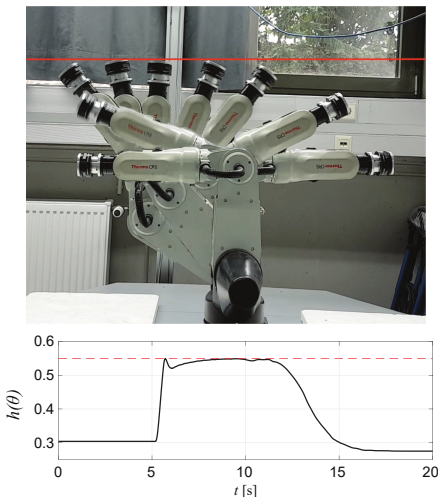
$$h(\rho^c) = -(l_1 \sin(\rho_1^c) + l_2 \sin(\rho_1^c + \rho_2^c) - c_x)^2 - (l_1 \cos(\rho_1^c) + l_2 \cos(\rho_1^c + \rho_2^c) - c_y)^2, \tag{4.23}$$

$$\begin{aligned} \nabla h_\theta(\rho^c) = [&2c_x(l_1 \cos(\rho_1) + l_2 \cos(\rho_1 + \rho_2) - 2c_y(l_1 \sin(\rho_1) + l_2 \sin(\rho_1 + \rho_2)) \\ &2l_1 l_2 \sin(\rho_2) + 2c_x l_2 \cos(\rho_1 + \rho_2) - 2c_y l_2 \sin(\rho_1 + \rho_2)]. \end{aligned}$$

The parameters of the circle obstacle are set to $c_x = 0.4$ m, $c_y = 0.8$ m and $r_c = 0.4$ m.



(a) Time domain plots of experiment with ceiling obstacle



(b) Robot motion during the experiment (top), evaluation of the ceiling constraint (bottom)

Figure 4.4: Experimental result - ceiling obstacle

4.4.3 Experimental Result

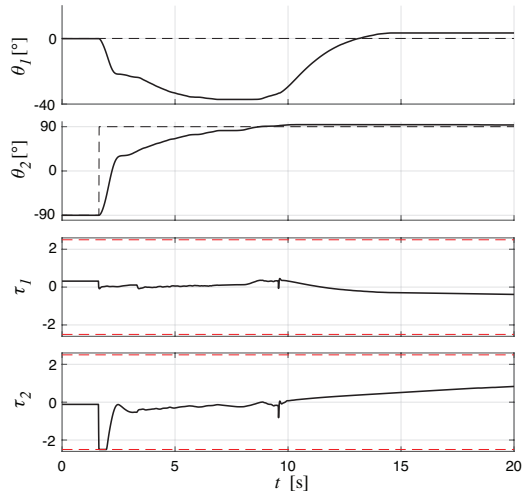
A set-point tracking scenario is investigated in which the goal is to reach $y_{sp} = [0 \ \pi/2]^\top$ starting from the position $y_0 = [0 \ \ -\pi/2]^\top$ (as depicted in Figure 4.3). Intuitively the unconstrained optimal solution is to move θ_2 as fast as possible and leave $\theta_1 = 0$, however, in the presented scenario the constrains preclude such a solution and θ_1 must move to make way if the second link is to reach its desired position. In addition to the nonlinear constraints discussed above, input constraints $|\tau_i| \leq 2.5, i = 1, 2$ are also considered. The prediction horizon is set to $N = 15$.

Ceiling

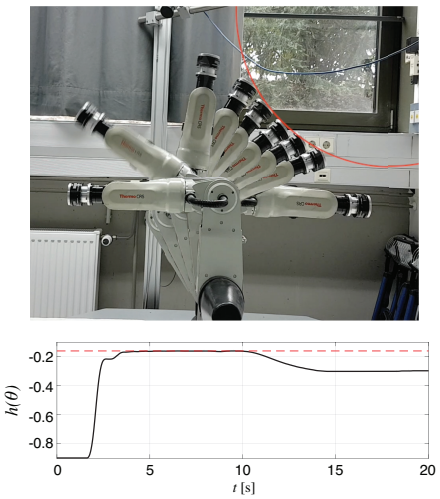
For this obstacle, the weighting matrices are chosen as $Q_1 = \text{diag}(1, 2)$, $Q_2 = \text{diag}(0.5, 0.5, 0, 0)$, $R = I$, $P = 2Q$. The value of R is chosen as such to avoid rapid changes in the input which might induce undesired vibrations (recall that unlike the usual case, here R penalizes Δu, whereas u is not directly penalized). The time domain plots of the experiment are shown in Figure 4.4a whereas the evolution of the position of the robot at several time instants along with the evaluation of constraint (4.21) at each time instant is shown in Figure 4.4b.

Circle

In this case, the weighting matrices are slightly retuned to $Q_1 = \text{diag}(1, 4)$, $Q_2 = \text{diag}(0.5, 0.5, 0, 0)$, $R = I$, $P = 2Q$, this was done to obtain a smoother response, since Coulomb friction caused the robot the stop at some points using the previous tuning. The time domain plots of the experiment are shown in Figure 4.5a whereas the evolution of the position of the



(a) Time domain plots of experiment with circle obstacle



(b) Robot motion during the experiment (top), evaluation of the circle constraint (bottom)

Figure 4.5: Experimental result - circle obstacle

robot at several time instants along with the evaluation of constraint (4.23) at each time instant is shown in Figure 4.5b.

Note that both constraints are strongly non-convex so Theorem 4.1 does not guarantee convergence to the desired set point, rather only to the optimal set point with a "line of sight" within the set described by the nonlinear constraints, which in this case would likely be as soon as the obstacle is seen within the prediction horizon. Indeed with certain values of the tuning parameters, the controller made the robot stop as soon as it reached the obstacle. A proof of convergence for arbitrary non-convex admissible sets is much more cumbersome and is, to the best of the author's knowledge, still an open problem. Furthermore, the use of an observer introduces uncertainty in the prediction, which can also make the output converge to a suboptimal setpoint.

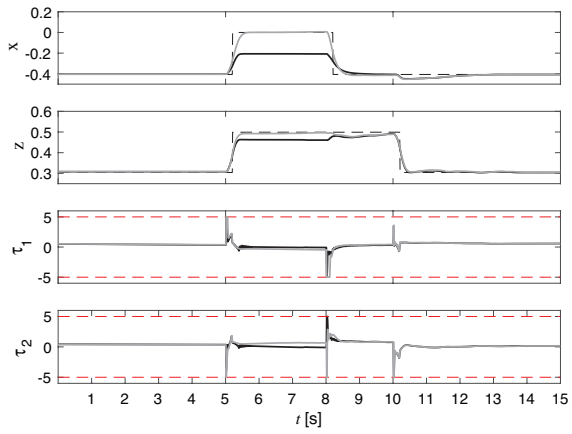
Nonlinear output

In practice, robotic manipulators usually carry out a task with their end-effectors; a practically relevant problem is therefore tracking in Cartesian space. This can be achieved by defining the nonlinear output to be[5]

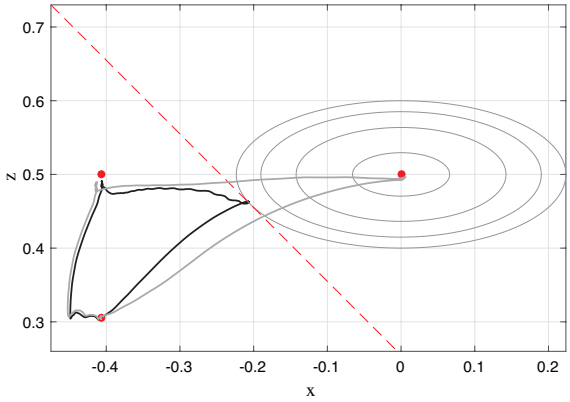
$$y(t) = \begin{bmatrix} \mathrm{x}(t) \\ \mathrm{z}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} l_1 \sin(\theta_1(t)) + l_2 \sin(\theta_1(t) + \theta_2(t)) \\ l_1 \cos(\theta_1(t)) + l_2 \cos(\theta_1(t) + \theta_2(t)) \end{bmatrix}}_{h(x)}$$

which can readily be included in the velocity model by augmenting the state as in (4.4). Two test

[5]note that roman x, z are used to denote the Cartesian coordinates, in order to distinguish them from the state variable x.



(a) Closed-loop response in Test 1 (——) and Test 2 (——)



(b) Closed-loop response in Test 1 (——) and Test 2 (——), output constraint (- - - -), sublevel sets of $||y - y_{sp,1}||_T^2$ (——)

Figure 4.6: Nonlinear output tracking: closed-loop response for reachable and unreachable set points

scenarios are considered which correspond to the two cases listed in Theorem 4.1; in *Test 1* the set points $y_{sp,i}$, $i = 1, 2, 3$ are all reachable, whereas in *Test 2* an output constraint is introduced, making set point $y_{sp,1}$ unreachable. The set points are the same for both tests:

$$
\begin{aligned}
y_{sp,1} &= [0 \;\; 0.5]^\top, \\
y_{sp,2} &= [-0.4064 \;\; 0.5]^\top, \\
y_{sp,3} &= [-0.4064 \;\; 0.3048]^\top,
\end{aligned}
$$

and the output constraint for Test 2 is $x + z \leq 0.25$; for both tests the input constraints are $-5 \leq \tau_1 \leq 5$, $-5 \leq \tau_2 \leq 5$. Tuning parameters are set to $Q = \text{diag}(1000, 5000, 20, 20, 0, 0)$, $R = \text{diag}(0.05, 0.05)$, $T = 250Q$ and the prediction horizon to $N = 20$.

Time domain plots of the closed-loop response during both experiments are shown in Figure 4.6a. Evidently, the outcome of Test 1 is as expected since both outputs reach their desired steady states; in Test 2 however, the set point is not reached due to the output constraint. To illustrate the convergence properties of the closed-loop in this scenario, the response in the x-z plane is shown in Figure 4.6a together with the output constraint and sublevel sets of $||y - y_{sp,1}||_T^2$. According to Theorem 4.1, the output should converge to the optimum within the set $\mathfrak{R}_y^N$ (the optimal point with a line of sight), given in this case by the point where the sublevel sets first intersect the diagonal line characterizing the constraint.

Stability is enforced by the terminal velocity constraint (4.13f), this means that the predicted trajectory should come to a steady state at the end of the horizon. The predicted velocity trajectories at $t \in [20 \;\; 20.05]$ s (i.e. when the first reference change takes place) are shown in Figure 4.7.

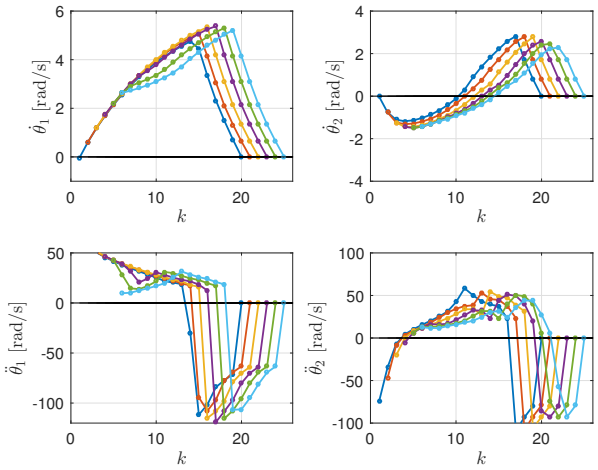


Figure 4.7: Predicted velocity trajectories at several time instants

4.5 Summary

The use of velocity algorithms has several benefits: it enables parameterization-free set point tracking, given that all equilibria are mapped to the origin[6]; it provides offset-free tracking thanks to its built-in integral action; it is able to track unreachable set points under mild assumptions on the set of reachable steady states; and stability can be guaranteed straightforwardly without the need for complex offline computations. A further benefit of the velocity algorithm is that it greatly simplifies the modelling phase by providing a simple, intuitive way to obtain a quasi-LPV model which can be readily automated for code generation. While the qLPV model resulting from velocity-linearization generally has a relatively high scheduling order, this does not represent a problem for the qLMPC framework; the fact that no terminal ingredients are used (other than the terminal velocity equality constraint) means that no offline LMI problem needs to be solved, which might otherwise prove difficult for systems with high scheduling order.

The main drawback of the approach presented in this chapter is that it requires the use of observers, or otherwise of numerical differentiation which could be detrimental to performance if the measured signals are too noisy. Furthermore it requires state augmentation, making computations more complex (although as seen in the previous chapter, offset-free control always requires state augmentation).

[6]Parameterization in the sense of finding equilibrium state and input values, x_s, u_s as a function of the set point y_{sp}

Chapter 5

Nonlinear MPC Using Input-Output qLPV Models

Chapters 3 and 4, as most of the MPC literature, focused on Model Predictive Control strategies on a state space (SS) framework. This is mainly due to the fact that Lyapunov arguments, on which the vast majority of stability results are based, naturally arise within this setting; however, there are practical reasons to consider using input-output (IO) models in lieu of state space models. Practical applications of SS methods often require the use of observers, which make both design and implementation more complex; tuning, for instance, becomes a bigger hurdle, since there is dynamic interplay within the controller and it is not always clear if it is the dynamics of the controller or the observer which should be re-tuned in order to improve performance, particularly in the nonlinear case where pole-location analysis is not available. Observers also introduce uncertainty, given that the state is estimated; in predictive control this can affect satisfaction of constraints, which can in turn void stability guarantees. Further motivation for IO models is that, in the (q)LPV case, identification techniques extending the prediction error method [14] are simpler and similarly accurate [105] than their SS counterparts, since the latter methods suffer from the *curse of dimensionality* and are difficult to apply to high order/high complexity systems. Moreover, as mentioned in previous chapters, in the stabilizing MPC SS framework, tracking problems are often handled by partitioning the state space, which limits its applicability to low-order systems; in the IO framework a partition of only the output space is necessary, which is typically considerably lower dimensional than the state space.

For the afore mentioned reasons, this chapter turns attention to the IO setting. Following recent developments in the IO-LPV literature to derive convex stability conditions in the form of LMIs [127], the methodology presented in Chapter 3 is extended to IO models. Likewise, an extension of GPC [40] – where LTI-IO models are routinely used – to qLPV systems in the context of qLMPC is presented, thus encompassing both stability analysis and implementation under the same framework henceforth referred to as IO-qLMPC.

The discussion starts with the introduction of several different, but equivalent representations of IO-LPV systems in Section 5.1; these representations become relevant when deriving stability conditions in Section 5.3, since the structure enables the development of convex conditions. Section 5.2 introduces the optimization problem and prediction equations to be used in the IO framework. Finally, Section 5.4 illustrates the approach experimentally on a 2-DOF robotic manipulator and Section 5.5 summarizes the chapter.

5.1 Input-Output LPV Representations

This section presents several alternative but equivalent IO qLPV representations, in particular kernel and image representations are discussed and their applicability for deriving convex stability conditions in the form of LMIs is briefly discussed.

Consider a nonlinear system whose dynamics can be represented as an input-output (IO) quasi-LPV model of the form

$$y(k) = \sum_{i=1}^{n_{d_y}} A_i(\rho(k))q^{-i}y(k) + \sum_{j=1}^{n_{d_u}} B_j(\rho(k))q^{-j}u(k) \tag{5.1}$$

as introduced in Definition 2.20 where $y(k) \in \mathbb{R}^l$, and $u(k) \in \mathbb{R}^m$ are the measured output and the control input, respectively, $\rho(k) = \varrho(y(k), u(k))$ is the parameter vector and q^{-1} is the backward time-shift operator. An output feedback controller in input-output form can be defined as

$$u(k) = \sum_{i=1}^{n_{Ka}} A_{Ki}(\rho(k))q^{-i}u(k) + \sum_{i=1}^{n_{Kb}} B_{Kj}(\rho(k))q^{-j}e(k) \tag{5.2}$$

where $e(k) = y(k) - r(k)$ is the error with respect to a reference signal $r(k)$.

Explicit IO representations

The usual modelling approach corresponds to finding an operator which explicitly maps inputs into outputs, e.g. in the LTI case a transfer function $G(p) : \mathbb{C}^m \to \mathbb{C}^l$ is often used such that $y(p) = G(p)u(p)$ where p is a complex variable ($p = s$ for continuous-time, $p = z$ for discrete-time). Similarly, the IO model (5.1) and controller (5.2) admit explicit representations as the transfer operators

$$\mathcal{G} = (I - \sum A_i q^{-i})^{-1}(\sum B_i q^{-i}), \quad \mathcal{K} = (I - \sum A_{Ki} q^{-i})^{-1}(\sum B_{Ki} q^{-i}),$$

respectively, where $\mathcal{G} : \ell_{2e}^m \to \ell_{2e}^l$, $\mathcal{K} : \ell_{2e}^l \to \ell_{2e}^m$. However, this representation has the disadvantage that when interconnected with other LPV systems (e.g. in feedback as in Figure 5.1), the resulting transfer operator displays dynamic parameter dependence in the sense of Definition 2.21, even if each model exhibits static parameter dependence. This happens whenever products of parameter dependent polynomials are carried out, since the time-shift operator does not commute with parameter dependent terms. This issue can be illustrated with the following example

Example 5.1. Given two models $y_i(k) = a_i(\rho_k)q^{-1}y(k) + b_iq^{-1}u_i(k)$, $i = 1, 2$ connected in series, so that $u_2(k) = y_1(k)$, an explicit representation of the interconnection given by

$$\begin{aligned} y_2(k) &= \left(\frac{1}{1 - a_2(\rho_k)q^{-1}}b_2q^{-1}\right)\left(\frac{1}{1 - a_1(\rho_k)q^{-1}}b_1q^{-1}\right)u_1(k) \\ &= \frac{b_2b_1}{(1 - a_2(\rho_k)q^{-1})(1 - a_1(\rho_{k-1})q^{-1})}u_1(k-2) \end{aligned}$$

which depends on time-shifted parameter values.

Implicit IO representations

Example 5.1 highlights an issue particular to LPV systems when using explicit representations. The reason why this represents a problem is that, when stability conditions are to be derived (often as LMI or BMI problems) shifted parameter values are difficult to handle and they need to conservatively be considered as taking their worst-case values (e.g. $\rho_k = \rho_{k-1} + \Delta\rho, \forall\Delta\rho \in \mathcal{V}$). To circumvent this issue, an implicit representation has been used in e.g. [126][1]. An implicit *kernel* representation of the closed-loop system formed in Figure 5.1 can be obtained by writing the model (5.1) and controller (5.2) as a system of equations on the inputs and outputs, i.e.

$$\underbrace{\begin{bmatrix} \bar{A}(\rho_k) & -\bar{B}(\rho_k) \\ -\bar{B}_K(\rho_k) & \bar{A}_K(\rho_k) \end{bmatrix}}_{D(\rho_k)} \underbrace{\begin{bmatrix} \bar{y}(k) \\ \bar{u}(k) \end{bmatrix}}_{\xi(k)} = \begin{bmatrix} 0 \\ -\bar{B}_K(\rho_k) \end{bmatrix} \bar{r}(k) \tag{5.3}$$

where

$$\begin{aligned} \bar{y}(k) &= \begin{bmatrix} y(k)^\top & y(k-1)^\top & \dots & y(k-n_{dy})^\top \end{bmatrix}^\top, \\ \bar{u}(k) &= \begin{bmatrix} u(k)^\top & u(k-1)^\top & \dots & u(k-n_{du})^\top \end{bmatrix}^\top, \\ \bar{r}(k) &= \begin{bmatrix} r(k)^\top & r(k-1)^\top & \dots & r(k-n_{dy})^\top \end{bmatrix}^\top \end{aligned}$$

and correspondingly

$$\begin{aligned} \bar{A}(\rho_k) &= \begin{bmatrix} I & -A_1(\rho_k) & \dots & -A_{n_{dy}}(\rho_k) \end{bmatrix} \\ \bar{B}(\rho_k) &= \begin{bmatrix} 0 & B_1(\rho_k) & \dots & B_{n_{du}}(\rho_k) \end{bmatrix} \\ \bar{A}_K(\rho_k) &= \begin{bmatrix} I & -A_{K1}(\rho_k) & \dots & -A_{Kn_{du}}(\rho_k) \end{bmatrix} \\ \bar{B}_K(\rho_k) &= \begin{bmatrix} 0 & B_{K1}(\rho_k) & \dots & B_{Kn_{dy}}(\rho_k) \end{bmatrix}. \end{aligned} \tag{5.4}$$

In case $n_{Ka} \neq n_{du}, n_{Kb} \neq n_{dy}$, i.e. if the order of the controller is different from that of the plant, the corresponding matrices in (5.3) are padded with zeros.

The use of kernel representation in [126] leads to stability conditions in the form of BMIs which, despite avoiding dynamic parameter dependence, are non-convex and generally hard to solve. A different implicit representation, deemed *image* representation, has been used in [127] to derive

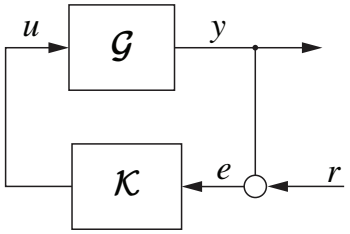


Figure 5.1: Closed-loop interconnection

convex stability conditions. This representation is based on the image of a matrix M which determines all feasible trajectories that satisfy (5.3) (or more specifically the autonomous part, i.e. $D\xi = 0$). Such an *image* representation is given by

$$\xi = M(\rho)n \tag{5.5}$$

such that

$$D\xi = DMn = 0, \forall n \neq 0.$$

This means that M spans the null space of D, i.e. $M = D_\perp$ and is clearly non-unique. However, given the structure of D, a choice of M can be constructed as

$$M(\rho) = \begin{bmatrix} A_1(\rho) & \dots & A_{n_{dy}}(\rho) & B_1(\rho) & \dots & B_{n_{du}}(\rho) \\ & I & & & 0 & \\ B_{K1}(\rho) & \dots & B_{Kn_{dy}}(\rho) & A_{K1}(\rho) & \dots & B_{Kn_{du}}(\rho) \\ & 0 & & & I & \end{bmatrix}. \tag{5.6}$$

The image representation (5.5) is further used in Section 5.3 to derive convex conditions that guarantee closed-loop stability of the MPC control law.

5.2 Predictive Controller

Consider a nonlinear model in IO form as in (5.1), assume output and input are subject to constraints $y \in \mathcal{Y}$, $u \in \mathcal{U}$. The predictive controller for models in IO form is essentially identical to its state space counterpart, the only difference is the equality constraint corresponding to the dynamics of the system. Before presenting how the prediction equation is included in the optimization, the optimization problem is presented. Assume the goal is to track a given reference set point y_s, the steady state input corresponding to y_s can be defined as $u_s = v(y_s)$. The stage cost is thus

$$\ell(e, \tilde{u}) = e^\top Q e + \tilde{u}^\top R \tilde{u} \tag{5.7}$$

where $e = y - y_s$, $\tilde{u} = u - u_s$ represent the deviation of the output from the reference value, and the deviation of the input from the steady state value, respectively. The matrices $Q \in \mathbb{R}^{l \times l}$ and $R \in \mathbb{R}^{m \times m}$ are both positive definite. The finite horizon cost function is defined as

$$J_k = \sum_{j=0}^{N-1} \ell(e_{k+j}, \tilde{u}_{k+j}) + \Psi(x_{k+N}) \tag{5.8}$$

where $\Psi(x_{k+N})$ is the terminal cost function, which takes as an argument a yet to be defined *state* x; the reason for this will become clear when deriving stability conditions in the next section. The optimization problem can thus be defined as

$$\begin{aligned}
&\min_{u} J_k(e, \tilde{u}) \\
&\text{s.t.} \\
&y_{k+j} = \sum_{i=1}^{n_{dy}} A_i(\rho(k)) q^{-i} y_{k+j} + \sum_{i=1}^{n_{du}} B_i(\rho(k)) q^{-i} u_{k+j} \\
&u_{k+j} \in \mathcal{U} \qquad\qquad j = [0 \quad N-1] \\
&x_{k+N} \in \mathbb{X}
\end{aligned} \tag{5.9}$$

where $\mathbb{X}$ is a terminal *state* constraint set.

5.2.1 Prediction with IO model

The proposed prediction equation is the quasi-LPV extension of the LTI input-output model prediction, commonly known as Generalized Predictive Control (see e.g. [101], [40]). Define the sequence of future outputs, future inputs and future parameters as

$$Y_k = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+N} \end{bmatrix} \qquad U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix} \qquad \mathsf{P}_k = \begin{bmatrix} \rho_k \\ \rho_{k+1} \\ \vdots \\ \rho_{k+N-1} \end{bmatrix}.$$

Similarly, define the vectors of past outputs and inputs

$$Y_k^p = \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-n_{dy}} \end{bmatrix} \qquad U_k^p = \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-n_{du}} \end{bmatrix}.$$

As done in the state space framework, an expression relating future inputs and outputs can be obtained by time-shifting (5.1) for $k = 1, 2, ..., N$ yielding

$$\mathcal{A}_f(\mathsf{P}_k) Y_k + \mathcal{A}_p(\mathsf{P}_k) Y_k^p = \mathcal{B}_p(\mathsf{P}_k) U_k^p + \mathcal{B}_f(\mathsf{P}_k) U_k \tag{5.10}$$

where

$$\mathcal{A}_f(\mathsf{P}_k) = \begin{bmatrix} I & 0 & 0 & \dots \\ A_1(\rho_{k+1}) & I & 0 & \dots \\ A_2(\rho_{k+2}) & A_1(\rho_{k+2}) & I & \dots \\ \vdots & & \ddots & \dots \\ A_{n_{dy}}(\rho_{k+n_{dy}}) & A_{n_{dy}-1}(\rho_{k+n_{dy}}) & \dots & \\ 0 & A_{n_{dy}}(\rho_{k+n_{dy}+1}) & \dots & \\ \vdots & & \ddots & \end{bmatrix} \in \mathbb{R}^{Nl \times Nl}$$

$$\mathcal{A}_p(\mathsf{P}_k) = \begin{bmatrix} A_1(\rho_k) & A_2(\rho_k) & \dots & A_{n_{dy}}(\rho_k) \\ A_2(\rho_{k+1}) & A_3(\rho_{k+1}) & \dots & 0 \\ \vdots & & \ddots & \vdots \\ A_{n_{dy}}(\rho_{k+n_{dy}-1}) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \end{bmatrix} \in \mathbb{R}^{Nl \times n_{dy} l}$$

$$\mathcal{B}_f(\mathsf{P}_k) = \begin{bmatrix} B_1(\rho_k) & 0 & \dots \\ B_2(\rho_{k+1}) & B_1(\rho_{k+1}) & \dots \\ B_3(\rho_{k+2}) & B_2(\rho_{k+2}) & \dots \\ \vdots & \ddots & \dots \\ B_{n_{du}}(\rho_{k+n_{du}-1}) & B_{n_{du}-1}(\rho_{k+n_{du}-1}) & \dots \\ 0 & B_{n_{du}}(\rho_{k+n_{du}}) & \dots \\ \vdots & & \ddots \end{bmatrix} \in \mathbb{R}^{Nl \times Nm}$$

$$\mathcal{B}_p(\mathsf{P}_k) = \begin{bmatrix} B_2(\rho_k) & B_3(\rho_k) & \dots & B_{n_{du}}(\rho_k) \\ B_3(\rho_{k+1}) & B_4(\rho_{k+1}) & \dots & 0 \\ \vdots & & \ddots & \vdots \\ B_{n_{du}}(\rho_{k+n_{du}-2}) & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \end{bmatrix} \in \mathbb{R}^{Nl \times n_{du} m}$$

Substituting the equality constraint (5.10) into the cost function of the optimization problem (5.9) results in a dense formulation, where the output trajectory is eliminated as a variable. To this end, define the maps

$$\begin{aligned} \Phi(\mathsf{P}_k) &= -\mathcal{A}_f(\mathsf{P}_k)^{-1}\mathcal{A}_p(\mathsf{P}_k) \\ \Phi_u(\mathsf{P}_k) &= -\mathcal{A}_f(\mathsf{P}_k)^{-1}\mathcal{B}_p(\mathsf{P}_k) \\ \Gamma(\mathsf{P}_k) &= -\mathcal{A}_f(\mathsf{P}_k)^{-1}\mathcal{B}_f(\mathsf{P}_k), \end{aligned}$$

so that the predicted output can explicitly be computed by

$$Y_k = \Phi(\mathsf{P}_k)Y_k^p + \Phi_u(\mathsf{P}_k)U_k^p + \Gamma(\mathsf{P}_k)U_k. \tag{5.11}$$

The dense optimization problem can then be written as

$$\begin{aligned} &\min_{U_k} \frac{1}{2} U_k^\top H(\mathsf{P}_k) U_k + g(\mathsf{P}_k)^\top U_k \\ &\text{subject to} \\ &\quad y_{k+i+1} \in \mathcal{Y} \\ &\quad u_{k+i} \in \mathcal{U} \qquad \forall i \in [0 \quad N-1] \\ &\quad x_{k+N} \in \mathbb{X} \end{aligned} \tag{5.12}$$

where

$$\begin{aligned} H(\mathsf{P}_k) &= 2(\hat{R} + \Gamma(\mathsf{P}_k)^\top \hat{Q} \Gamma(\mathsf{P}_k)) \\ g(\mathsf{P}_k)^\top &= 2\Gamma(\mathsf{P}_k)^\top \hat{Q} (\Phi Y_k^p(\mathsf{P}_k) + \Phi_u(\mathsf{P}_k) U_k^p - r_k) \end{aligned} \tag{5.13}$$

and the yet to be defined terminal state x_{k+N} is left as a variable to avoid messy expressions; for now it suffices to say that the constraint can be expressed as additional constraints on the input sequence, as done in (3.6).

5.3 Stability

In this section, the stability result as presented in Section 3.2 is reformulated to be used in the IO framework. The idea can be summarized as follows:

- create a suitable state vector with past output and input data,
- derive stability conditions in a state space framework,
- adapt stability conditions to the newly defined state, considering the IO model and cost function.

For the last step, an essential tool in linear algebra is Finsler's Lemma, the relevant part of which is presented next.

Lemma 5.1 (Finsler's Lemma [91])**.** Given $Q \in \mathbb{S}^n$ and $B \in \mathbb{R}^{m \times n}$ such that $\operatorname{rank}(B) < n$, the following statements are equivalent

i) $x^\top Q x < 0, \forall x : Bx = 0, x \neq 0$

ii) $B_\perp^\top Q B_\perp \prec 0$

For simplicity it is assumed that $\mathcal{Y} = \mathbb{R}^l$, i.e. there are no output constraints. This assumption is made since terminal constraint sets should lie within $\mathcal{Y}$, however, especially if the set is non-convex, characterizing this condition as an LMI is case dependent and cannot be generalized, in an effort to keep the derivations general, the assumption is made.

In order to ease the transition from SS to IO, this section focuses on the regulator problem, in which the objective is to regulate the origin, i.e. the zero-input zero-output equilibrium, the extension to tracking is presented in the next section. As a first step, a latent variable is defined

$$x(k) = \Pi_1 \begin{bmatrix} \bar{y}(k) \\ \bar{u}(k) \end{bmatrix} \in \mathbb{R}^n \tag{5.14}$$

where $n = ln_{dy} + mn_{du}$, which can be used as a state variable for the choice

$$\Pi_1 = \begin{bmatrix} \Pi_{1y} & 0 \\ 0 & \Pi_{1u} \end{bmatrix},$$

$$\Pi_{1y} = [0_{ln_{dy} \times l} \quad I_{ln_{dy}}], \quad \Pi_{1u} = [0_{mn_{du} \times m} \quad I_{mn_{du}}].$$

Note that this results in

$$x(k) = \begin{bmatrix} y^\top(k-1) & \dots & y^\top(k-n_{dy}) & u^\top(k-1) & \dots & u^\top(k-n_{du}) \end{bmatrix}^\top.$$

Similarly, the succeeding state $x(k+1)$ can be written as

$$x(k+1) = \Pi_2 \begin{bmatrix} \bar{y}(k) \\ \bar{u}(k) \end{bmatrix}, \text{ with } \Pi_2 = \begin{bmatrix} \Pi_{2y} & 0 \\ 0 & \Pi_{2u} \end{bmatrix},$$

$$\Pi_{2y} = [I_{ln_{dy}} \quad 0_{ln_{dy} \times l}], \quad \Pi_{2u} = [I_{mn_{du}} \quad 0_{mn_{du} \times m}],$$

resulting in $x(k+1) = \begin{bmatrix} y^\top(k) & \dots & y^\top(k-n_{dy}+1) & u^\top(k) & \dots & u^\top(k-n_{du}+1) \end{bmatrix}^\top$.

Recall the stability result in Theorem 3.1, having defined a *state* variable for the IO model, the result can be applied in the IO framework, however certain subtleties need to be addressed: consider the terminal cost function $\Psi(x) = x^\top P x$ as before, condition (iv) in Theorem 3.1 requires [1]

$$x_{k+1}^\top P x_{k+1} - x_k^\top P x_k \leq -y_{k-1}^\top Q y_{k-1} - u_{k-1}^\top R u_{k-1}. \tag{5.15}$$

This expression can also be written in a more compact form in terms of past inputs and outputs by substituting the definition of the latent variable (5.14) and the following selector matrices

$$\Pi_y = [I \quad 0 \quad \dots \quad 0][\Pi_{1y} \quad 0], \quad \Pi_u = [I \quad 0 \quad \dots \quad 0][0 \quad \Pi_{1u}].$$

Note that $y_{k-1} = \Pi_y \begin{bmatrix} \bar{y}_k \\ \bar{u}_k \end{bmatrix}$ and $u_{k-1} = \Pi_u \begin{bmatrix} \bar{y}_k \\ \bar{u}_k \end{bmatrix}$ therefore (5.15) can also be written as

$$[*]^\top \begin{bmatrix} P & 0 & 0 & 0 \\ 0 & -P & 0 & 0 \\ 0 & 0 & Q & 0 \\ 0 & 0 & 0 & R \end{bmatrix} \begin{bmatrix} \Pi_2 \\ \Pi_1 \\ \Pi_y \\ \Pi_u \end{bmatrix} \begin{bmatrix} \bar{y}_k \\ \bar{u}_k \end{bmatrix} \leq 0 \tag{5.16}$$

[1] Note that the stage cost ℓ depends only on the output y in the IO case, whereas in the SS case depends on the full state.

which must hold for all trajectories of the system given by (cf. Equation (5.3))

$$D(\rho)\begin{bmatrix}\bar{y}_k\\ \bar{u}_k\end{bmatrix} = 0,\ \forall \rho \in \mathcal{P}.$$

Using Finsler's Lemma, equation (5.16) can also be written as

$$[*]^\top \begin{bmatrix} P & 0 & 0 & 0\\ 0 & -P & 0 & 0\\ 0 & 0 & Q & 0\\ 0 & 0 & 0 & R\end{bmatrix}\begin{bmatrix}\Pi_1\\ \Pi_2\\ \Pi_y\\ \Pi_u\end{bmatrix} M(\rho) \prec 0\ \ \forall \rho \in \mathcal{P} \tag{5.17}$$

where $M(\rho)$ is defined in (5.6). Note that, given the structure of $M(\rho)$, the following holds

$$\Pi_1 M(\rho) = I,$$

$$\Pi_y M(\rho) = [I \quad 0 \quad \dots \quad 0],$$

$$\Pi_u M(\rho) = [0 \quad \dots \quad I \quad 0 \quad \dots],$$

$$\Pi_2 M(\rho) = \begin{bmatrix} A_1(\rho) & \dots & A_{n_{dy}}(\rho) & B_1(\rho) & \dots & B_{n_{du}}(\rho)\\ I & & 0 & & 0 & \\ & I & 0 & & 0 & \\ B_{K1}(\rho) & \dots & B_{Kn_{dy}}(\rho) & A_{K1}(\rho) & \dots & A_{Kn_{du}}(\rho)\\ & 0 & & I & & 0\\ & 0 & & & I & 0\end{bmatrix}.$$

The last expression can also be written as

$$\Pi_2 M(\rho) = \hat{A}(\rho) + \hat{B}\bar{K}(\rho),$$

where

$$\hat{A}(\rho) = \begin{bmatrix} A_1(\rho) & \dots & A_{n_{dy}}(\rho) & B_1(\rho) & \dots & B_{n_{du}}(\rho)\\ I & & 0 & & 0 & \\ & I & 0 & & 0 & \\ 0 & \dots & 0 & 0 & \dots & 0\\ & 0 & & I & & 0\\ & 0 & & & I & 0\end{bmatrix}, \qquad \hat{B} = \begin{bmatrix}0\\ \vdots\\ 0\\ I\\ 0\\ \vdots\\ 0\end{bmatrix}$$

$$\bar{K}(\rho) = \begin{bmatrix} B_{K1}(\rho) & \dots & B_{Kn_{dy}}(\rho) & A_{K1}(\rho) & \dots & A_{Kn_{du}}(\rho)\end{bmatrix}.$$

Hence the problem of finding an input-output terminal controller that makes the terminal cost a Lyapunov function boils down to a standard scheduled state feedback problem on the state defined in (5.14).

Having obtained a Lyapunov inequality for the terminal cost function in an input-output setting, it remains to establish how a terminal constraint set can be constructed. Consider the sublevel set of $\Psi(x)$

$$\mathbb{X} = \{x | x^\top P x \leq \alpha\},$$

with $\alpha > 0$. To guarantee that this set is invariant under the control law $\kappa(x) = \bar{K}x$, it must hold that $\bar{K}x \in \mathcal{U},\ \forall x \in \mathbb{X}$. i.e. within the terminal region, the terminal controller gives admissible inputs. In order to enlarge the feasible region, the terminal set needs to be as large as possible, therefore α is to be maximized, subject to the input constraints on $\kappa(x)$.

The following result summarizes the discussion above.

Theorem 5.1 (Stability of input-output qLMPC)**.** The MPC law arising from the solution of (5.9) in a receding horizon fashion, for $\Psi(x_{k+N}) = x_{k+N}^\top P x_{k+N}$ and $\mathbb{X} = \{x | x^\top P x \leq \alpha\}$ guarantees asymptotic stability of the origin if P and α are chosen as the solutions to the following LMI problem

$$\min_{X,Y,\tilde{\alpha}} \ \tilde{\alpha} \tag{5.18a}$$

s.t.

$$\begin{bmatrix} -Y & * & * & * \\ \hat{A}(\rho)Y + \hat{B}X(\rho) & -Y & * & * \\ [I\ 0\ \dots\ 0]Y & 0 & -Q^{-1} & 0 \\ [0\ \dots\ I\ 0\ \dots]Y & 0 & 0 & -R^{-1} \end{bmatrix} \leq 0\ \forall \rho \in \mathcal{P} \tag{5.18b}$$

$$\begin{bmatrix} \tilde{\alpha} u_{i,max}^2 & e_i X(\rho) \\ * & Y \end{bmatrix} \geq 0 \quad i \in [1\ m] \tag{5.18c}$$

$$[0_{1\times lk}\ \varrho_j\ 0_{1\times n-l(k+1)}]Y[*]^\top \leq \tilde{\alpha}\rho_{j,max}^2 \quad j \in [1\ n_\rho] \quad k \in [0\ n_{dy}-1] \tag{5.18d}$$

$$[0_{1\times ln_{dy}+mk}\ e_j\ 0_{1\times m(n_{du}-k-1)}]Y[*]^\top \leq \tilde{\alpha} u_{j,max}^2 \quad j \in [1\ m] \quad k \in [0\ n_{du}-1] \tag{5.18e}$$

$$Y \succ 0 \quad \tilde{\alpha} > 0 \tag{5.18f}$$

where $Y = P^{-1}$, $\tilde{\alpha} = 1/\alpha$, $X(\rho) = \bar{K}(\rho)Y$, $u_{i,max}$ is the maximum admissible value of the i^{th} input, e_i is the i^{th} column of the identity matrix and ϱ_j is the j^{th} row of the linear map $\varrho : \mathbb{R}^l \to \mathbb{R}^{n_\rho}$.[a]

[a] LMIs (5.18d) assume the map ϱ is linear, i.e. $\rho = \varrho x$. Given that there are no restrictions on the functional dependence of the model on ρ there is no reason *not* to pick ϱ to be linear.

Remark 5.1. *The inequalities in* (5.18) *are infinite dimensional in ρ as they must hold $\forall \rho \in \mathcal{P}$. This issue is pragmatically solved by freezing the parameter ρ over a grid in $\mathcal{P}$ and inferring validity of the conditions given continuity of the model.*

Proof. The proof is based on the fact that the optimization problem (5.18) fulfills the conditions in Theorem 3.1 on a state space of the state (5.14)

Part I: (5.18b) $\Rightarrow$ condition 4. of Theorem 3.1

Apply Schur complement to (5.18b) and substitute $\Pi_y M = [I\ \ 0\ \ \dots\ \ 0]$, $\Pi_u M = [0\ \ \dots\ \ I\ \ 0\ \ \dots]$ to obtain

$$\begin{bmatrix} -Y + [*]^\top Q\Pi_y MY + [*]^\top R\Pi_u MY & * \\ AY + BX & -Y \end{bmatrix} \preceq 0$$

Applying Schur complement a second time yields

$$-Y + [*]^\top Y^{-1}(\hat{A}Y + BX) + [*]^\top Q\Pi_y MY + [*]^\top R\Pi_u MY \preceq 0$$

Perform a congruence transformation by pre- and post-multiplying by $P = Y^{-1}$, and substitute $\bar{K} = XY^{-1}$

$$-P + [*]^\top P(\hat{A} + B\bar{K}) + [*]^\top Q\Pi_y M + [*]^\top R\Pi_u M \preceq 0$$

This expression can then be written in the form of (5.17), which was seen to be equivalent to (5.15), completing the proof.

Part II: (5.18c) $\Rightarrow$ condition 5. of Theorem 3.1

Apply Schur complement to (5.18c) and substitute $X = \bar{K}Y$ to obtain

$$\tilde{\alpha}u_{i,max}^2 - e_i\bar{K}Y\bar{K}^\top e_i^\top \geq 0$$

or

$$\bar{K}_i(\alpha Y)\bar{K}_i^\top \leq u_{i,max}^2$$

where $\bar{K}_i$ is the i^{th} row of $\bar{K}$. According to Lemma 3.2, this is the condition for the i^{th} input to be upper bounded by $|u_{i.max}|$ within the ellipsoidal set characterized by

$$x^\top \left(\frac{1}{\alpha}P\right) x \leq 1$$

or equivalently by $x^\top Px \leq \alpha$. That this ellipsoid is invariant follows from the validity of condition 4. for all x.

Finally, condition 1. follows trivially from the definition of the ellipsoid.

Part III: (5.18d), (5.18e) $\Rightarrow$ condition 6. of Theorem 3.1

The first step in this part of the proof is to interpret these LMIs. As the stability conditions are only imposed over the admissible parameter set $\mathcal{P}$, the projection of the ellipsoid onto $\mathbb{R}^{n_\rho}$ must be a subset of $\mathcal{P}$ (LMI (5.18d)). The index k in this LMI, serves to impose this constraint on all past values of the output, since past values of the parameter must also be contained in $\mathcal{P}$. Furthermore, as the state also contains past input values, the ellipsoid projected onto $\mathbb{R}^m$ should also be limited to grow at most to the admissible input values (LMI (5.18e)), again, past values of the input must also be admissible, so this is imposed for all dimensions of the ellipsoid corresponding to past inputs (indexed by k).

These LMIs are direct consequences of Lemma 3.2. □

Remark 5.2. *In the presented form, the resulting dual controller $\bar{K}(\rho)$ inherits the functional dependence of $X(\rho)$ which is a design parameter (one normally chooses a set of basis functions, i.e. $X(\rho) = X_0 + X_1 f_1(\rho) + ...$), the additional degrees of freedom can help achieve better performance, or in this case, a larger ellipsoid. An omission worth mentioning here is the possibility to consider a parameter dependent Lyapunov function $Y(\rho)$, which also results in having scheduled ellipsoids (cf. [37]). This issue is not discussed here, in order to keep the discussion brief.*

Remark 5.3. *Maximization of α (minimization of $\tilde{\alpha}$) in* (5.18) *is not necessary to provide stability, since it is guaranteed by the existence of the terminal ingredients and feasibility of the online optimization. Maximization of the volume of the ellipsoid is done to enlarge the feasible region (or to allow for shorter horizons). If a* smaller *P is desired (smaller in the sense of its determinant) regardless of the size of the ellipsoid (e.g. if no stability guarantees are required and no terminal constraint is used) the optimization objective can be changed in order to discourage large eigenvalues of P by modifying the objective function to be* $\min_{X,Y,\tilde{\alpha},t} \tilde{\alpha} - at$ *with the additional LMI $Y(\rho) > t$, $\forall \rho \in \mathcal{P}$, where $a > 0$ is a tuning parameter to weight the desire to make the ellipsoid large or P small.*

Remark 5.4. *Note that the* state *vector as defined in* (5.14) *is ordered in reverse with respect to Y_k, U_k; for this reason, the terminal ingredient matrices P, W should undergo a coordinate transformation (permutation) when implementing the control law, for them to be consistent with Y_k, U_k, i.e. $\tilde{P} = T^\top P T$, $\tilde{W} = T^\top W T$ where*

$$T = \left[\begin{array}{ccc|ccc} 0 & \dots & I & & & \\ \vdots & \iddots & \vdots & & 0 & \\ I & \dots & 0 & & & \\ \hline & & & 0 & \dots & I \\ & 0 & & \vdots & \iddots & \vdots \\ & & & I & \dots & 0 \end{array}\right].$$

5.3.1 Set point tracking

So far the discussion has focused on stability conditions for the origin of a state space formed by past input and output data, which implies stability of the zero-input zero-output equilibrium, i.e. if $y_s = 0$ and $u_s = 0$ in (5.7). However, it is often of practical interest to consider set point tracking. This section discusses how the stability conditions given earlier are to be modified to consider set point tracking; intuitively the terminal region has to be translated, such that it contains the set point in its interior, rather than the origin. However, when doing so one has to consider that, when one has a non-zero steady state input (i.e. a forced equilibrium), the admissible input within the terminal region is further constrained and that this might reduce the size of the terminal set, in the extreme case down to a point.

Define the set of admissible reference set points $\mathcal{R}$ as the set of steady state outputs, for which the corresponding steady state input is admissible, i.e.

$$\mathcal{R} = \left\{ r : \left(I - \sum_i A_i(\rho_s) \right) r = \sum_j B_j(\rho_s) u_s, \;\; u_s \in \mathcal{U} \right\}.$$

where $\rho_s = \varrho(y_s)$, $y_s = r$. Note that, given y_s, one can calculate the corresponding steady state input by

$$u_s = \left(\sum_j B_j(\rho_s) \right)^{-1} \left(I - \sum_i A_i(\rho_s) \right) y_s$$

provided the inverse exists. If the system is not invertible (e.g. if $l \neq m$) then the solution can be calculated by solving the minimum input norm problem

$$\begin{aligned}
&\min_{u_s} \; u_s^\top R u_s \\
&\text{s.t.} \\
&\sum_j B_j(\rho_s) u_s = \left(I - \sum_i A_i(\rho_s) \right) y_s \\
&u_s \in \mathcal{U}
\end{aligned}$$

Regardless of the method, a function $u_s = v(r) : \mathbb{R}^l \to \mathbb{R}^m$ can be defined which maps an admissible reference to an admissible steady state input. With these definitions, the conditions presented in the previous section can be modified to consider an ellipsoid that takes the set of admissible set points into account and is viable for tracking. Consider the following modification to conditions (5.18c), (5.18d) and (5.18e):

$$\begin{bmatrix} \tilde{\alpha}(u_{i,max} - v(r))^2 & e_i X \\ * & Y \end{bmatrix} \geq 0 \quad i \in [1\ m] \quad \forall r \in \mathcal{R} \tag{5.19a}$$

$$[0_{1\times lk}\ \varrho_j\ 0_{1\times n-l(k+1)}] Y [*]^\top \leq \tilde{\alpha}(\rho_{j,max} - \varrho_{j,max}(r))^2 \quad j \in [1\ n_\rho] \quad k \in [0\ n_{dy} - 1] \tag{5.19b}$$

$$[0_{1\times ln_{yd}+mk}\ e_j\ 0_{1\times m(n_{du}-k-1)}] Y [*]^\top \leq \tilde{\alpha}(u_{j,max} - v_{j,max}(r))^2 \quad j \in [1\ m] \quad k \in [0\ n_{du} - 1] \tag{5.19c}$$

and the translated ellipsoid

$$\mathbb{X}_r = \left\{ x : (x - \tilde{r})^\top P (x - \tilde{r}) \leq \alpha \right\}$$

with the augmented reference $\tilde{r} = [r^\top \ \dots \ r^\top \ u_s^\top \ \dots \ u_s^\top]^\top$. The subindex $_{max}$ indicates the maximum value of the signal or function, respectively. Substituting these modified conditions (5.19) in place of (5.18c), (5.18d) and (5.18e) in problem (5.18) to be solved offline and using the translated ellipsoid in the online implementation makes the stability condition apply to piece-wise constant reference tracking. Conditions (5.19b) and (5.19c) impose the constraint $\text{Proj}_{\mathbb{R}^{n_\rho} \times \mathbb{R}^m}(\mathbb{X}_r) \subset \mathcal{P} \times \mathcal{U}$, which must hold $\forall r \in \mathcal{R}$, this is illustrated in Figure 5.2.

Remark 5.5. *Given that recursive feasibility is established by assuming a single ellipsoid, this property could be lost after every reference change (i.e. when switching from one ellipsoid to the next). It therefore follows that recursive feasibility is only guaranteed in the subsequent steps after a set point change occurs, if the optimization problem is feasible at the first time step after a set point change. Said otherwise, this approach does not have the property of tracking of unreachable set points (within N steps), so that intermediate set points would need to be computed.*

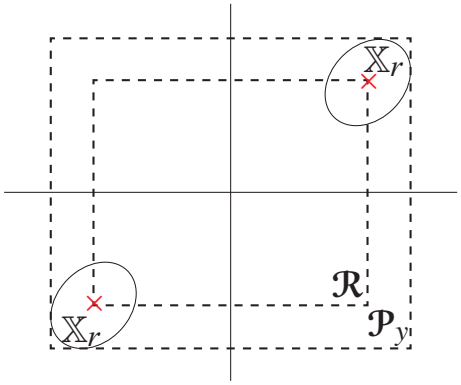


Figure 5.2: The projected ellipsoid ($\mathbb{X}_r = \text{Proj}_{\mathbb{R}^l}(\mathbb{X})$) translated to all admissible reference set points must be contained in the image of $\mathcal{P}$ in $\mathbb{R}^l$ ($\mathcal{P}_y$)

Relationship to other methods The idea of translating and scaling the terminal set is used, e.g. in [111]. In this work a nominal terminal set is computed offline, and a scaling factor is optimized online by adding additional decision variables. The presented approach attempts to alleviate the online computational load by shifting this to an offline computation, furthermore, the ellipsoidal sets do not lend themselves to a relatively simple condition as the polyhedral sets from [111] do. A different approach which is used e.g. in [13] and [28] is to partition the state space and compute a terminal set for each partition. This approach is similar to the one presented here, but has the clear disadvantage that for high order systems, the curse of dimensionality would lead to an unrealistic number of partitions. In this case the input-output framework is much more suitable, as the "partitions" (the grid) is to be done on the output space only. In the presented form, however, a disadvantage is that the same ellipsoid is used for all admissible set points, which is potentially very conservative if the system is such that set points on the boundary of the admissible set are considered. To reduce this conservatism, a parameter dependent ellipsoid can be considered as in [37] such that it can be scaled as a function of the reference set point.

5.4 Application Example: Input-Output qLMPC of a 2-DOF Robotic Manipulator

In the previous chapter the tracking problem of a 2-DOF robotic manipulator was presented in the context of velocity-based qLMPC. The same example is used in this case as it has the nice feature of having a continuous locus of equilibria, thereby lending itself directly to set point tracking using terminal constraints. Recall that a nonlinear model for the 2-DOF configuration in Figure 4.2b is given by

$$M(\theta(t))\ddot{\theta}(t) + c(\theta(t), \dot{\theta}(t)) + g(\theta(t)) + \tau_f(\dot{\theta}(t)) = \tau(t) \tag{5.20}$$

where

$$M(\theta) = \begin{bmatrix} b_1 + b_2 + 2b_3\cos(\theta_2) & b_2 + b_3\cos(\theta_2) \\ b_7 - b_8 + b_3\cos(\theta_2) & b_7 \end{bmatrix}, \quad c(\theta) = \begin{bmatrix} -b_3\dot{\theta}_2^2\sin(\theta_2) - 2b_3\dot{\theta}_1\dot{\theta}_2\sin(\theta_2) \\ b_3{\dot{\theta}_1}^2\sin(\theta_2) \end{bmatrix},$$
$$g(\theta) = \begin{bmatrix} -b_4\sin(\theta_1+\theta_2) - b_5\sin(\theta_1) \\ -b_4\sin(\theta_1+\theta_2) \end{bmatrix} \qquad \tau_f(\dot{\theta}) = \begin{bmatrix} b_6\dot{\theta}_1 \\ b_9\dot{\theta}_2 \end{bmatrix}. \tag{5.21}$$

and physical parameters are listed in Table 4.1.

Whereas in the previous chapter, the SS quasi-LPV model of the robot is obtained via velocity-based linearization, in this case an appropriate discretization of the nonlinear differential equation (5.20) can be carried out to obtain an IO model. Indeed using the central difference method, i.e. making the substitutions

$$\ddot{\theta} = \frac{\theta_{k+1} - 2\theta_k + q_{k-1}}{T_s^2} \qquad \dot{\theta} = \frac{\theta_{k+1} - \theta_{k-1}}{2T_s}$$

with $T_s = 0.01$ s on the equations of motion yields the discrete-time IO model[2]

$$\underbrace{\left(\frac{1}{T_s^2}M(\theta_k) + \frac{1}{2T_s}D\right)}_{A_0(\theta)}\theta_{k+1} + \underbrace{\left(K(\theta_k) - \frac{2}{T_s^2}M(\theta_k)\right)}_{A_1(\theta)}\theta_k + \underbrace{\left(\frac{1}{T_s^2}M(\theta_k) - \frac{1}{2T_s}D\right)}_{A_2(\theta)}\theta_{k-1} = \underbrace{I}_{B_1}\tau_k \tag{5.22}$$

where D is the damping matrix, such that $\tau_f(\dot{\theta}) = D\dot{\theta}$ and $K(\theta)$ is a *stiffness* matrix obtained by factorizing the gravity force vector as $g(\theta) = K(\theta)\theta$ using the substitution $\sin(\theta) = \mathrm{sinc}(\theta)\theta$. Both the output and the parameter vector are defined to be the angles, i.e. $y = \theta$, $\rho = y$.

Note that throughout the chapter it is implicitly assumed that $A_0 = I$ (cf. (5.1), (5.4)). In order to solve the LMI in Theorem 5.1 (i.e. to establish stability) this indeed needs to be the case, so that the matrices need to be redefined to $\tilde{A}_1 = A_0^{-1}A_1$, $\tilde{A}_2 = A_0^{-1}A_1$, $\tilde{B}_1 = A_0^{-1}$. However for the implementation of the control law, this is not necessary, instead the matrix $\mathcal{A}_f$ in (5.10) can be modified to have A_0 along the diagonal instead of I. Note that, given its structure, A_0 is always invertible.

5.4.1 Stabilizing IO-qLMPC

Terminal ingredients are computed by solving the LMI in Theorem 5.1 (with suitable modifications for set point tracking as discussed in Section 5.3.1) for the model (5.22). The augmented input and output vectors are

$$\bar{y}(k) = \begin{bmatrix} y(k) \\ y(k-1) \\ y(k-2) \end{bmatrix} \qquad \bar{u}(k) = \begin{bmatrix} u(k) \\ u(k-1) \end{bmatrix},$$

[2]For simplicity, Coriolis terms are neglected as it is well known that their contribution is minor.

consequently,

$$x(k) = \begin{bmatrix} y(k-1) \\ y(k-2) \\ u(k-1) \end{bmatrix} \qquad \hat{A}(\rho) = \begin{bmatrix} \tilde{A}_1(\rho) & \tilde{A}_2(\rho) & \tilde{B}_1(\rho) \\ I & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \hat{B}(\rho) = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}.$$

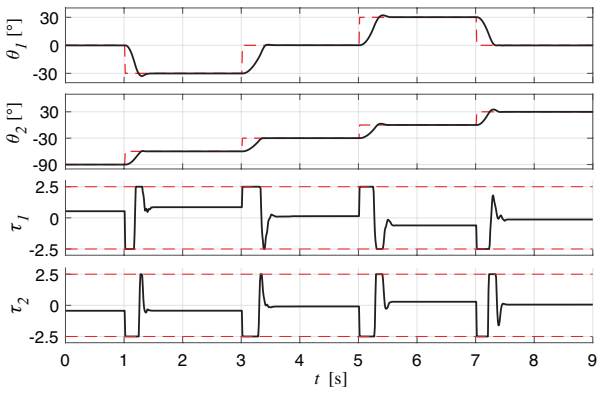
The tuning parameters are chosen as $Q = \text{diag}(150, 100)$, $R = \text{diag}(0.05, 0.01)$, $N = 25$ and admissible reference, and parameter sets are defined as

$$\mathcal{P} = \begin{cases} \rho_1 \in [-\pi \quad \pi] \\ \rho_2 \in [-\pi \quad \pi] \end{cases} \qquad \mathcal{R} = \begin{cases} r_1 \in [-\pi/3 \quad \pi/3] \\ r_2 \in [-\pi/2 \quad \pi/2] \end{cases}$$

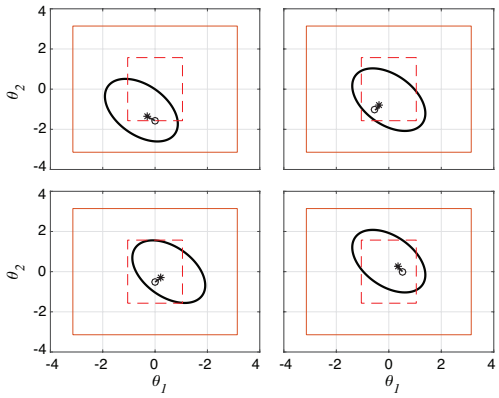
respectively. Note that $\mathcal{P}$ is substantially larger than $\mathcal{R}$, in order to avoid a small ellipsoid (recall Figure 5.2). The objective function of the LMI is redefined as per Remark 5.3 to $\tilde{\alpha} - 5t$.

Experimental results

The reference trajectory consists of a sequence of steps, and the terminal constraint is translated accordingly. Time domain plots of the closed-loop response during the experiments are shown in Figure 5.3a. The controller is able to track the reference trajectory swiftly making full use of the available torque, evidenced by the input trajectories which are driven into saturation at every step change. The ellipsoids projected onto the $\theta_1 - \theta_2$ plane at every step change together with the predicted trajectory are illustrated in Figure 5.3b. Given the large size of the ellipsoid, the terminal constraint can be easily fulfilled, despite the relatively short horizon, thereby making the region of attraction large.



(a) 2-DOF robotic manipulator closed-loop response.



(b) Translated ellipsoid for each of the step reference changes. The circle (o) represents $y(k)$ and the star (*) the predicted $y(k+N)$. Admissible sets $\mathcal{P}$ (——) and $\mathcal{R}$ (- - - - -).

Figure 5.3: Closed-loop response of 2-DOF robotic manipulator using IO-qLMPC

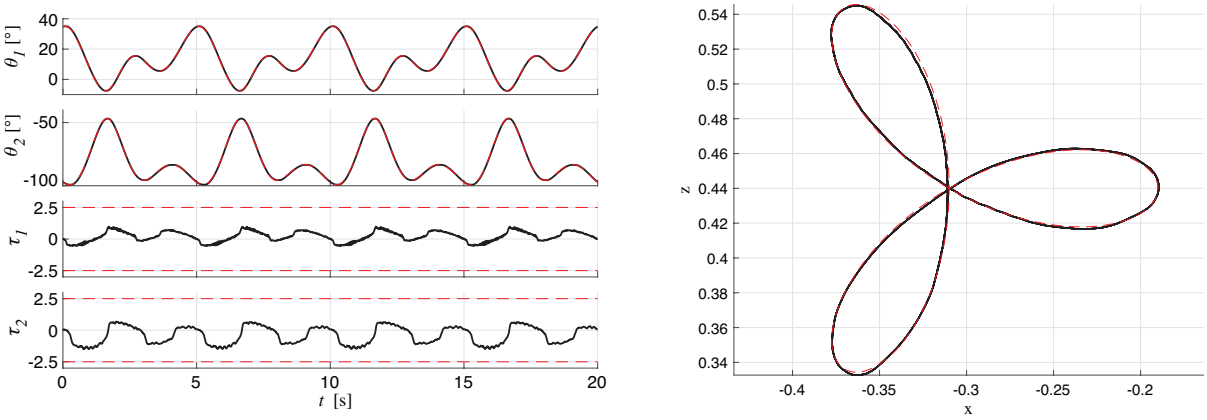
5.4.2 Terminal-Constraint-Free Case

Although the IO framework enables the tracking problem to be practically handled with stability guarantees (i.e. with a stabilizing terminal constraint), as gridding of the admissible set points is only required in the output space, it is still limited to set point tracking; that is, tracking of piece-wise constant reference trajectories. Tracking of arbitrary smooth trajectories can however be accomplished by relinquishing said stabilizing terminal constraint. This issue is briefly explored in this section.

The trajectory in the joint space (θ_1, θ_2) is calculated given the following trajectory in Cartesian coordinates:

$$r_{\mathrm{x}} = 0.06(\sin(0.4\pi t) + \cos(0.8\pi t)) + 0.36$$
$$r_{\mathrm{z}} = 0.06(\cos(0.4\pi t) + \sin(0.8\pi t)) + 0.47$$

which corresponds to a 3-petal rose; the joint coordinates are subsequently calculated using inverse kinematics. Define the terminal cost to be $\Psi(y_{k+N}) = y_{k+N}^{\top} P_y y_{k+N}$ with $P_y = 100Q$ and as before take $Q = \mathrm{diag}(150, 100)$, $R = \mathrm{diag}(0.05, 0.01)$, $N = 25$. The response of the closed-loop experiment with this controller is shown in Figure 5.4a in both joint and Cartesian spaces, where it can be seen that the IO-qLMPC law achieves virtually perfect tracking.



(a) Response in joint space

(b) Response in Cartesian space

Figure 5.4: 2-DOF robotic manipulator closed-loop response, terminal-constraint-free case

5.5 Summary

The IO-qLMPC framework proves to be a very good alternative to its SS counterpart. It features a comparatively simpler design with fewer tuning parameters; design and implementation are further simplified as the need for observers is forgone. Furthermore, the operations entailed by the prediction equation (5.10) scale better with prediction horizon than their SS counterparts (Equation (3.5)) since the number of linear algebra operations is substantially lower. In the

application example it is made clear that there is no trade-off in performance when compared to SS methods, as tracking is seen to be virtually perfect. On the other hand, the stabilizing framework is readily applicable to tracking problems, since partitioning (gridding) is only necessary in the output space and not in the state space as it would in the SS framework.

Limitations or downsides of the IO framework are the comparative difficulty to obtain analytic IO-qLPV models, since discretization can in some cases make the model more complex. Moreover, the fact that there are fewer tuning parameters (weightings only for inputs and outputs), despite making design simpler, it also makes it less flexible. Take mechanical systems as an example, the outputs are often taken as positions so that there is usually no tuning knob to penalize velocities, so increasing damping is not as straightforward. This can be compensated by augmenting the output with the velocities at the cost of increasing complexity of the control law.

Chapter 6

Data-Driven qLMPC Using Koopman Operators

Thus far the methods presented in this work have striven for a healthy trade-off between rigorous stability guarantees and practicality of the approach. In the treated examples it has been concluded that stability guarantees often translate in loss of flexibility of the approach by e.g. limiting applicability to the regulator case or to tracking of piece-wise constant reference trajectories. This chapter presents a data-driven approach to the previously presented methods which is strongly inclined towards practicality. Indeed as far as purely data-driven techniques go (as opposed to adaptive schemes where a model is given but parameters are adapted online) establishing stability is a complicated matter that is often done only under unrealistic assumptions about a priori knowledge of the system.

The main goal of the Koopman operator framework is to discover (identify) the dynamics of a nonlinear system as a linear model on an infinite-dimensional augmented space of so-called *observables*, which are nonlinear functions of the state and input. In practice, the Koopman operator is truncated and the model becomes finite- but typically high-dimensional. The reason why this framework is of interest is that the truncated Koopman operator acts as a transition matrix which propagates the observables forward in time and can be found using data-driven techniques by fitting the model to measured data; as the model is linear, this process is simple and efficient. The lifted model, together with the observables, can readily be used within the velocity-qLMPC framework from Chapter 4 by obtaining a model as presented in [124][5], thus enabling a data-driven model to be used with qLMPC.

As the Koopman framework represents a departure from the topics that have been discussed so far, a brief overview about Koopman operator theory is given in Section 6.1; this is not intended to provide a comprehensive survey of the framework but rather focus of the relevant concepts to be applied. Section 6.2 discusses how the truncated Koopman operator is practically computed using data-driven techniques with emphasis on online computation. In Section 6.3, the issue of finding a velocity quasi-LPV model (akin to the methodology in Chapter 4) is discussed. The data-driven qLMPC control scheme is illustrated in Section 6.4, on an Control Moment Gyroscope experimental platform, which given its nonlinear and highly complex dynamics represents an interesting problem for this technique. Finally, Section 6.5 concludes the chapter.

6.1 Koopman Operator Theory

In this section a brief overview of Koopman operator theory is given, a comprehensive study is not presented for brevity, but the interested reader is referred to [95], [23] for a more thorough discussion. In the context of this thesis only the relevant issues are discussed and subtleties and nuances of the formalism are left out.

In its original definition, the Koopman operator is applied to autonomous systems (i.e. systems without input); in order to provide a segue to the use of Koopman operators for data-driven control, the discussion is first focused on such systems. Consider an autonomous system whose dynamics are described by the model

$$z_{k+1} = F(z_k) \tag{6.1}$$

where $z \in \mathbb{R}^{n_z}$ and the map $F : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ propagates the state forward in time. In addition, a set of scalar valued functions, referred to as observables, is defined as

$$\psi_i : \mathbb{R}^{n_z} \to \mathbb{R},\ i \in \{1, 2, \cdots, n_\psi\}$$

which belong to an infinite dimensional Hilbert space $\mathcal{H}$ of Lebesgue square-integrable functions. The Koopman operator $\mathcal{K} : \mathcal{H} \to \mathcal{H}$ which is defined on the system (6.1) acts on these observables $\psi_i \in \mathcal{H}$ and is characterized by

$$\mathcal{K}(\psi_i) = \psi_i \circ F,$$

as a consequence the following holds

$$\psi_i(z_{k+1}) = \psi_i(F(z_k)) = (\mathcal{K}\psi_i)(z_k) \qquad \forall z_k \in \mathbb{R}^{n_z}.$$

The observables can be stacked to obtain a vector-valued observable $\Psi \in \mathcal{H}^{n_\psi}$ as a map $\Psi : \mathbb{R}^{n_z} \to \mathbb{R}^{n_\psi}$ defined as $\Psi(z) = [\psi_1(z) \ \ \psi_2(z) \cdots \psi_{n_\psi}(z)]^T$. A Koopman operator can be defined as $\mathcal{K}_{n_\psi} : \mathcal{H}^{n_\psi} \to \mathcal{H}^{n_\psi}$ on system (6.1) characterized by

$$\mathcal{K}_{n_\psi}(\Psi) = \Psi \circ F$$

which yields

$$\Psi(z_{k+1}) = \Psi(F(z_k)) = (\mathcal{K}_{n_\psi}\Psi)(z_k) \qquad \forall z_k \in \mathbb{R}^{n_z}. \tag{6.2}$$

Note that the Koopman operator propagates the observables forward in time, thus serving as a transition operator, i.e. a one-step ahead *prediction* model for the observables. However, in practice the Koopman operator as previously defined is linear but infinite-dimensional making its applicability difficult. It is therefore of interest to find a finite dimensional *approximation* of the Koopman operator which can practically be computed.

Consider a finite dimensional approximation $K \in \mathbb{R}^{n_\psi \times n_\psi}$ of the the Koopman operator $\mathcal{K}_{n_\psi}$ such that

$$\Psi(z_{k+1}) = \Psi(F(z_k)) = (\mathcal{K}_{n_\psi}\Psi)(z_k) \approx K\Psi(z_k) \quad \forall z_k \in \mathbb{R}^{n_z} \tag{6.3}$$

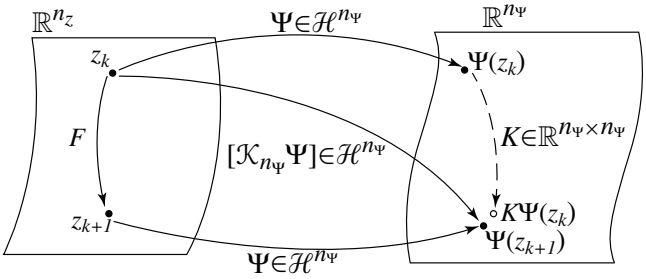


Figure 6.1: Illustration of the exact and approximate Koopman operators

Note that the finite dimensional approximation K approaches the exact infinite dimensional operator $\mathcal{K}_{n_\psi}$ as the dimension n_ψ of the image space of Ψ is increased by stacking more linearly independent functions (in the sense of the inner product on $\mathcal{H}$) ψ_i to Ψ. An illustration highlighting the relationship between the different maps presented above can be seen in Figure 6.1.

6.1.1 Systems with Inputs

The discussion above, in line with the original definition of the Koopman operator, focused on autonomous systems. An extension to systems with inputs is presented in [95] and a brief summary is presented next. Consider a system whose dynamics are governed by the discrete-time non-linear model

$$x_{k+1} = f(x_k, u_k) \tag{6.4}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. The idea is to lift the system by defining an augmented state $z_k \in \mathbb{R}^{n_z}$ with $n_z = n + m$ as

$$z_k = \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \tag{6.5}$$

The succeeding lifted state is defined to be

$$z_{k+1} = \begin{bmatrix} x_{k+1} \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} f(x_k, u_k) \\ * \end{bmatrix} = F(z_k) \tag{6.6}$$

where $F : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ as defined in (6.1); the same framework as presented above can thus be used. However, as is elaborated in [95], depending on the particular application, one could replace $*$ in the above equation by either 0 or u_{k+1}, depending if the input is external, so that no dynamics may want to be discovered, or if it obeys dynamic equations (e.g. a controller) that may be of interest to be discovered by the Koopman operator. In this case, the interest is on discovering the dynamics in (6.4) which is characterized by the projected map $\hat{F} : \mathbb{R}^{n_z} \to \mathbb{R}^n$ defined by $\hat{F} = [I \;\; 0]F$. It is therefore valid to choose either 0 or u_{k+1}, as in either case the projection is not affected.

6.1.2 Obtaining a State Prediction Model

Consider the rightmost equality in equation (6.3) as a prediction equation for the observables on the augmented state (6.5). It is conceivable that having such a prediction model, an appropriate choice of observables (basis functions) can be made in order to obtain an expression for the prediction of the state itself, i.e. an approximation of (6.4). Indeed by selecting

$$\Psi\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) = \begin{bmatrix} x^\top & u^\top & \psi_{n+m+1}\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) & \dots & \psi_{n_\psi}\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) \end{bmatrix}^\top \tag{6.7}$$

an approximation of the model (6.4) is given by

$$x_{k+1} \approx \hat{K}\Psi\left(\begin{bmatrix} x_k \\ u_k \end{bmatrix}\right) \tag{6.8}$$

where $\hat{K} = \begin{bmatrix} I_n & 0 \end{bmatrix} K$.

Note that, as expected, the system dynamics are linear in a lifted space spanned by the observables, which are now treated as basis functions. In this way, nonlinearities can be captured by an appropriate choice of $\Psi(\cdot)$. This means that these functions play a crucial role in the accuracy of the approximated model, so their choice must be done with care. An often used choice applies to systems for which the structure of the dynamic equations is partially known, so that basis functions can be selected by picking some or all of the nonlinear functions of the underlying dynamical model, i.e. the R.H.S of (6.4). Alternatively, polynomial, Fourier, radial or any other standard basis functions can be used instead.

6.2 Computation of approximate Koopman Operator

There is only a limited number of applications for which an analytic Koopman operator can be computed. However, where the interest in Koopman operator theory lies, is in its capability to discover nonlinear system dynamics using data-driven techniques. In this sense, the Koopman operator extends the Dynamic Mode Decomposition ([104]) approach to nonlinear systems. In this section, an algorithm to compute the Koopman operator is presented which is tailored for online-based learning, as opposed to system identification-like algorithms in which computational load is not critical.

Assume that a data-set $\mathcal{Z} = \{z_i, z_i^+\}_{i=0,1,\dots P}$ is collected, where z_i and z_i^+ are consecutive data points. It is not necessary (indeed not even desired) for the data collected in $\mathcal{Z}$ to correspond to a single trajectory, i.e. any two data pairs $\{\{z_i, z_i^+\}, \{z_{i+1}, z_{i+1}^+\}\}$ need not be consecutive. In order to find the Koopman operator which better approximates (6.3) given a set of basis functions, an optimization problem is solved. The cost function can be chosen in several ways, in the present context a quadratic function of the approximation residual is used, i.e.

$$J = \frac{1}{2}\sum_{i=0}^{P-1} ||\Psi(z_i^+) - K\Psi(z_i)||^2.$$

Depending on the application, other choices might be more meaningful, for instance adding a sparsity-promoting L_1-regularization term [22] can be used in cases in which there is no prior knowledge about the system and a large number of basis functions are attempted in order to prioritize the most important ones. On the other hand, choosing a more complex cost function precludes to some extent an online-adaptive implementation and is, for this reason, not pursued in this thesis. A quadratic cost function is an attractive choice as it enables a closed-form solution and can therefore be efficiently implemented. In light of this, an analytic solution to the optimization problem can be obtained as follows: define the data matrices $\mathcal{D}, \mathcal{D}^+ \in \mathbb{R}^{n_\psi \times P}$ as

$$\mathcal{D} = [\Psi(z_0) \;\; \Psi(z_1) \cdots \Psi(z_P)],$$

$$\mathcal{D}^+ = [\Psi(z_0^+) \;\; \Psi(z_1^+) \cdots \Psi(z_P^+)]$$

obtained by stacking the basis functions corresponding to the collected data column-wise. Note that, according to the definition of z and z^+, $\mathcal{D}^+$ is just the time-shifted version of $\mathcal{D}$.

It can be shown that the cost function previously defined is equivalent to

$$J = \frac{1}{2}||\mathcal{D}^+ - K\mathcal{D}||_F^2$$

where $||X||_F = \sqrt{\text{Trace}(X^T X)}$ is the Frobenius norm. The optimization problem is therefore

$$\min_K \quad \frac{1}{2}||\mathcal{D}^+ - K\mathcal{D}||_F^2$$

which is unconstrained and has a differentiable cost function, therefore an analytic solution can be found by solving

$$\frac{\partial J}{\partial K} = K\mathcal{D}\mathcal{D}^T - \mathcal{D}^+\mathcal{D}^T \stackrel{!}{=} 0_{n_\psi \times n_\psi} \tag{6.9}$$

for the minimizing K, thus given by

$$K\mathcal{D}\mathcal{D}^T = \mathcal{D}^+\mathcal{D}^T$$
$$K = AG^\dagger$$

where the following matrices have been defined for convenience [124][1]

$$\begin{aligned} G &= \frac{1}{P}\mathcal{D}\mathcal{D}^T = \frac{1}{P}\sum_{p=0}^{P} \Psi(z_p)\Psi(z_p)^T \\ A &= \frac{1}{P}\mathcal{D}^+\mathcal{D}^T = \frac{1}{P}\sum_{p=0}^{P} \Psi(z_p^+)\Psi(z_p)^T. \end{aligned} \tag{6.10}$$

[1]The G and A matrices are normalized by the total number of collected data points P. This is done to avoid large entries on the matrices which could result in numerical problems at implementation time.

6.2.1 Recursive Computation

An online solution of (6.9) using the definitions (6.10) would require the storage and update of the data vectors $\mathcal{D}^+$, $\mathcal{D}$. Whereas this is often done in many data-driven techniques, particularly those based on kernel methods [125], it has the clear disadvantage that memory allocation and limitation could represent a problem. Moreover, the computation would become increasingly demanding with the number of stored data points P. Alternatively, (6.10) could be expressed in a such way that incoming data is added to the model, without the need to explicitly recompute the previous outer products; this can be done by storing $G, A \in \mathbb{R}^{n_\psi \times n_\psi}$ and $P \in \mathbb{Z}_{\geq 0}$ and at time step k computing

$$\begin{aligned} P^+ &= P + 1 \\ G^+ &= \frac{1}{P^+}\left((P^+ - 1)G + \Psi(z_{k-1})\Psi(z_{k-1})^\top\right) \\ A^+ &= \frac{1}{P^+}\left((P^+ - 1)A + \Psi(z_k)\Psi(z_{k-1})^\top\right) \end{aligned} \tag{6.11}$$

where $^+$ is used to denote the updated variables. Using these definitions, the Koopman operator can be updated online with consistent and relatively low computational complexity, regardless of the number of data points that have been added to the, now fictitious, data set $\mathcal{Z}$. Furthermore, memory allocation poses no limitation as the size of the matrices being stored (G, A, P) is constant.

Koopman Operator Update

Although as discussed above, adding new information to the model (in the form of data pairs $\{z_{k-1}, z_k\}$) does not result in considerable computational burden, doing so at every time step is not advisable, as adding e.g. steady-state data-points which do not contribute to enrich the model, would indeed result in actual new information having diminished influence on updating the model (the weight $1/P$ of any new data point would make the effect negligible for large values of P). For this reason, incoming state and input data need to be analyzed in some way to determine if they represent *new* information to enrich the model. There are several ways to do this, the one presented here is intuitive and easy to implement. The principle at the core is to "learn" only if the current model disagrees (more than a certain threshold) with the observation. To this end, assume the state and input at the previous $h + 1$ time instants is stored in a vector

$$\phi = \begin{bmatrix} x_{k-1}^\top & u_{k-1}^\top & \dots & x_{k-h-2}^\top & u_{k-h-2}^\top \end{bmatrix}^\top,$$

a decision whether to add the data pair $\{z_{k-1}, z_k\}$ to the model is made by comparing the maximum prediction error of the approximated system (6.8) within this time window. If this error is greater than a given threshold, the data point is added, i.e. add data point if

$$\left\| \begin{bmatrix} x_{k-h} \\ x_{k-h+1} \\ \vdots \\ x_k \end{bmatrix} - \begin{bmatrix} \hat{K}\Psi(z_{k-h-1}) \\ \hat{K}\Psi\left(\begin{bmatrix} \hat{K}\Psi(z_{k-h-1}) \\ u_{k-h} \end{bmatrix}\right) \\ \vdots \\ \hat{K}\Psi\left(\begin{bmatrix} \hat{K}\Psi\left(\begin{bmatrix} \hat{K}\Psi(\ldots) \\ u_{k-2} \end{bmatrix}\right) \\ u_{k-1} \end{bmatrix}\right) \end{bmatrix} \right\|_\infty \geq \nu \tag{6.12}$$

where $\|\cdot\|_\infty$ is the vector infinity norm[2]. Note that any other norm may be used instead, however this choice offers the advantage that it is independent of the window length h and it enables the use of physical insight to determine a suitable ν (i.e. an admissible error tolerance in physical units). Note that in equation (6.12), the state is computed recursively rather than by the linearity-exploiting form $x_k = [I_n \ 0]K^h\Psi(z_{k-h-1})$, the reason for this is that the dynamic behavior of the input is not being discovered by the Koopman operator, so that the input trajectory is virtually exogenous.

6.3 Koopman-Operator-Based quasi-LPV Model

This section brings the Koopman operator framework into the context of qLMPC by showing how a quasi-LPV model can be obtained from the prediction equation (6.8). The method closely follows the velocity-based approach from Section 4.2, to this end, the following assumption is made

A.1 The basis functions $\Psi\left(\begin{bmatrix} x \\ u \end{bmatrix}\right)$ are differentiable w.r.t $\begin{bmatrix} x \\ u \end{bmatrix}$.

State space models

In a state space setting, the same procedure to obtain a discrete-time velocity model as presented in Section 4.2.1 can be followed. Indeed an incremental model can be obtained by using velocity-based linearization (in its discrete-time version using Multivariable Mean Value Theorem) on (6.8) to obtain

$$\Delta x_{k+1} \approx \hat{K}\frac{\partial\Psi(\tilde{x},\tilde{u})}{\partial x}\Delta x_k + \hat{K}\frac{\partial\Psi(\tilde{x},\tilde{u})}{\partial u}\Delta u_k$$

where $\tilde{x} \in \mathrm{Co}(x_k, x_{k+1})$, $\tilde{u} \in \mathrm{Co}(u_k, u_{k+1})$. Note that the expression above does not entail an additional approximation, i.e. the trajectories of the incremental model exactly match those of (6.8). However, $\tilde{x}$ and $\tilde{u}$ cannot be computed in practice since they represent future inter-time step information. For this reason, at the expense of an additional approximation error, the incremental state can be computed from

[2]If the states have different physical units, they can be normalized; otherwise a different bound ν, one for each physical unit can be used

$$\Delta x_{k+1} \approx \underbrace{\hat{K}\frac{\partial \Psi(x_k, u_k)}{\partial x}}_{A(\rho_k)} \Delta x_k + \underbrace{\hat{K}\frac{\partial \Psi(x_k, u_k)}{\partial u}}_{B(\rho_k)} \Delta u_k \tag{6.13}$$

where $\rho = H[x^\top \quad u^\top]^\top$ and H is a selector matrix. Finally (6.13) is augmented with the (non-incremental) state giving the dynamic equation

$$\begin{bmatrix} x_{k+1} \\ \Delta x_{k+1} \end{bmatrix} \approx \begin{bmatrix} I & A(\rho_k) \\ 0 & A(\rho_k) \end{bmatrix} \begin{bmatrix} x_k \\ \Delta x_k \end{bmatrix} + \begin{bmatrix} B(\rho_k) \\ B(\rho_k) \end{bmatrix} \Delta u_k \tag{6.14}$$

Remark 6.1. *Note that equation* (6.13) *is nonlinear, whereas the model* (6.8) *based on the approximate Koopman operator is linear; at this point, one might rightfully ask the motivation for this apparent increase in complexity. The main advantage is that the quasi-LPV model* (6.13) *obtained by velocity-based linearization is usually of much lower order than the Koopman linear model. If the model is to be updated online, as proposed, this makes the computations much cheaper, as one of the main sources of overhead is building the prediction matrices in* (3.5)*. Admittedly, if the Koopman operator is found offline and is not adapted online, the Hessian and gradient matrices in* (3.7) *could be precomputed offline and linear MPC based on the linear Koopman model would likely be the best solution.*

Input-output models

Even though the goal of this framework is to obtain a data-driven model with as little a priori knowledge of the system as possible, in practice one strong assumption is made, namely that the state is known and measurable (given that the state is part of the observable functions, cf. (6.7)). This imposition might prove prohibitive for high order systems where it is impractical to assume even knowledge of the number of states, let alone being able to measure them. Building upon the input-output framework from Chapter 5, this strong assumption can be relaxed to an assumption regarding the order of the system. Assume the dynamics of the system are governed by an IO model of the form

$$y_{k+1} = \tilde{f}(y_k, u_k, q^{-1}) \tag{6.15}$$

Note that (6.15) is a general discrete-time input-output model for a system governed by differential algebraic equations. In this case, a state vector can be defined as

$$x_k = \begin{bmatrix} y_k^\top & y_{k-1}^\top & \cdots & y_{k-n_y}^\top & u_{k-1}^\top & \cdots & u_{k-n_u}^\top \end{bmatrix}^\top \tag{6.16}$$

where $y \in \mathbb{R}^l$ and $n = n_y + n_u$ is the assumed order of the system. Recall that a similar approach of creating a state vector from previous IO data was used in Chapter 5 to enable the stability result developed for state space qLPV models in Chapter 3 to be applicable in an IO setting, indeed this *trick* is often encountered in the polynomial/IO setting in order to apply Lyapunov-like results to IO models. Whereas in Chapter 5 a mathematically rigorous approach is followed to obtain a state variable based on image representations of IO models and Finsler's Lemma, in

the context of machine learning where little a priori information about the system is assumed, a more pragmatic choice is followed by arbitrarily choosing the order $n_y + n_u$.

Using the state vector (6.16), the basis functions are defined as before, as

$$\Psi\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) = \begin{bmatrix} x^\top & u^\top & \psi_{n+m+1}\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) & \dots & \psi_{n_\psi}\left(\begin{bmatrix} x \\ u \end{bmatrix}\right) \end{bmatrix}^\top,$$

and the Koopman operator can equally be computed recursively via (6.11). Analogous to the state space approximate state prediction (6.8), an approximation of system (6.15) is given by

$$y_{k+1} \approx \hat{K}_{IO}\Psi(x_k, u_k) \tag{6.17}$$

where $\hat{K}_{IO} = [I_l \quad 0]K$.

Following the discussion on the previous section, a velocity-based model is readily obtained by

$$\begin{bmatrix} y_{k+1} \\ \Delta x_{k+1} \end{bmatrix} \approx \begin{bmatrix} I & \hat{A}(\rho_k) \\ 0 & A(\rho_k) \end{bmatrix} \begin{bmatrix} y_k \\ \Delta x_k \end{bmatrix} + \begin{bmatrix} \hat{B}(\rho_k) \\ B(\rho_k) \end{bmatrix} \Delta u_k \tag{6.18}$$

where $A(\rho), B(\rho)$ are given in (6.13) and

$$\hat{A}(\rho_k) = \hat{K}_{IO}\frac{\partial \Psi(x_k, u_k)}{\partial x} \quad \hat{B}(\rho_k) = \hat{K}_{IO}\frac{\partial \Psi(x_k, u_k)}{\partial u}.$$

Note that, having defined a state vector in the IO case, the same augmented state space prediction equation (6.14) can be used. However, this would entail carrying out unnecessary computations to predict future backward-shifted inputs and outputs.

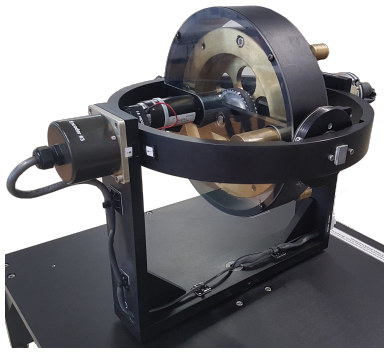
Having derived suitable qLPV models, a velocity-based state space qLMPC as presented in Chapter 4 can readily be applied.

6.4 Application Example: Data-Driven qLMPC of a Control Moment Gyroscope

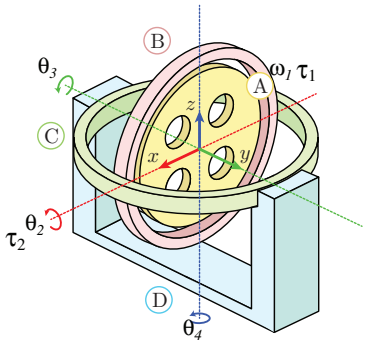
A Control Moment Gyroscope (CMG) is a mechanical device which consists of a flywheel mounted on a 3-degree-of-freedom gimbal. The flywheel and the first gimbal, bodies A and B in Figure 6.2b, are actuated whereas the two outermost gimbals (C and D) are not, they can however be controlled by exploiting the gyroscopic effect. The goal is thus to track reference trajectories for the outer two gimbals, using the torque of the flywheel and the torque of the motor actuating the innermost gimbal as control variables. The CMG used for the experiments presented in this section is the Model 750 from Educational Control Products (ECP) shown in Figure 6.2a.

Similar to the robot model presented in previous chapters, a model of the CMG is given by a second order vector differential equation [58]

$$M(\theta(t))\ddot{\theta}(t) + c(\theta(t), \dot{\theta}(t)) + \tau_f(\dot{\theta}(t)) = \tau(t) \tag{6.19}$$



(a) ECP Model 750 Control Moment Gyroscope



(b) CMG schematic

Figure 6.2: 3-DOF control moment gyroscope

where $\theta = \begin{bmatrix}\theta_1 & \theta_2 & \theta_3 & \theta_4\end{bmatrix}^\top$ and

$$M(\theta) = \begin{bmatrix} b_1 & 0 & b_1 c_2 & b_1 s_2 c_3 \\ 0 & b_3 & 0 & -b_3 s_3 \\ b_1 c_2 & 0 & b_2 s_2^2 + b_4 & -b_2 s_2 c_2 c_3 \\ b_1 s_2 c_3 & -b_3 s_3 & -b_2 s_2 c_2 c_3 & -b_2 s_2^2 c_3^2 + b_5 s_3^2 + b_6 \end{bmatrix}, \quad \tau_f(\dot{\theta}) = \begin{bmatrix} b_{13}\dot{q}_1 \\ b_{14}\dot{q}_2 \\ b_{15}\dot{q}_3 \\ b_{15}\dot{q}_4 \end{bmatrix}, \quad \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ 0 \\ 0 \end{bmatrix},$$

$$c(\theta,\dot{\theta}) = \begin{bmatrix} b_1\left(c_2c_3\dot{q}_2\dot{q}_4 - s_2\dot{q}_2\dot{q}_3 - s_2s_3\dot{q}_3\dot{q}_4\right) \\ b_1\left(s_2\dot{q}_1\dot{q}_3 - c_2c_3\dot{q}_1\dot{q}_4\right) + b_2\left(c_2c_3^2s_2\dot{q}_4^2 - c_2s_2\dot{q}_3^2\right) - b_8c_3\dot{q}_3\dot{q}_4 + b_7\left(1-2s_2^2\right)c_3\dot{q}_3\dot{q}_4 + 2b_9c_2^2c_3\dot{q}_3\dot{q}_4 \\ b_1\left(s_2s_3\dot{q}_1\dot{q}_4 - s_2\dot{q}_1\dot{q}_2\right) + \left(b_8+b_7\right)c_3\dot{q}_2\dot{q}_4 + b_{11}s_3c_3\dot{q}_4^2 + b_{10}\left(2c_2^2c_3\dot{q}_2\dot{q}_4 - 2s_2c_2\dot{q}_2\dot{q}_3 - s_3c_2^2c_3\dot{q}_4^2\right) \\ b_1\left(c_2c_3\dot{q}_1\dot{q}_2 - s_2s_3\dot{q}_1\dot{q}_3\right) + b_2s_2s_3c_2\dot{q}_3^2 - 2b_{11}s_3c_3\dot{q}_3\dot{q}_4 + 2b_{10}\left(c_2^2c_3\dot{q}_2\dot{q}_3 + s_2c_2c_3^2\dot{q}_2\dot{q}_4 + s_3c_2^2c_3\dot{q}_3\dot{q}_4\right) + b_{12}c_3\dot{q}_2\dot{q}_3 \end{bmatrix}$$

where b_i, $i = 1, ..., 15$ are constants and the short-hand notations $c_i = \cos(\theta_i)$ and $s_i = \sin(\theta_i)$, $i = 2, 3$ are used. Note that there is no gravity vector as due to its physical construction, this plant is not affected by gravity. A state vector can be defined as

$$x = [\theta_2 \quad \theta_3 \quad \theta_4 \quad \dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3 \quad \dot{\theta}_4]^\top,$$

where the angle θ_1 is disregarded since it has no effect on the dynamics and in normal operation is unbounded, given that the flywheel is perpetually in motion.

A comparison of IO-qLMPC and V-qLMPC using the full model (6.19) has been presented in [24]. In both cases, velocity-based linearization proves to be fundamental to obtain a quasi-LPV model, given the complexity of the nonlinear model. The resulting velocity models, although highly complex, lead to successful and indeed well-performing controllers. In the context of data-driven control, it would be desirable to forgo the need for such complex models without giving up performance; this issue is explored in what follows.

6.4.1 Selection of Basis Functions

Given that a model for the CMG is available, a first meaningful approach is to select a few of the nonlinearities directly from the equations of motion to be used as basis functions. Therefore, all Coriolis terms (given by the function $c(\theta, \dot{\theta})$) are considered i.e.

$$\begin{aligned}\Psi_1 = \big[&x^\top \;\; u^\top \;\; c_2c_3\dot\theta_2\dot\theta_4 \;\; s_2\dot\theta_2\dot\theta_3 \;\; s_2s_3\dot\theta_3\dot\theta_4 \;\; s_2\dot\theta_1\dot\theta_3 \;\; c_2c_3\dot\theta_1\dot\theta_4 \;\; c_2c_3^2s_2\dot\theta_4^2 \;\; c_2s_2\dot\theta_3^2 \;\; c_3\dot\theta_3\dot\theta_4 \\ &s_2^2c_3\dot\theta_3\dot\theta_4 \;\; c_2^2c_3\dot\theta_3\dot\theta_4 \;\; s_2s_3\dot\theta_1\dot\theta_4 \;\; s_2\dot\theta_1\dot\theta_2 \;\; c_3\dot\theta_2\dot\theta_4 \;\; s_3c_3\dot\theta_4^2 \;\; c_2^2c_3\dot\theta_2\dot\theta_4 \;\; s_2c_2\dot\theta_2\dot\theta_3 \;\; s_3c_2^2c_3\dot\theta_4^2 \\ &c_2c_3\dot\theta_1\dot\theta_2 \;\; s_2s_3\dot\theta_1\dot\theta_3 \;\; s_2s_3c_2\dot\theta_3^2 \;\; s_3c_3\dot\theta_3\dot\theta_4 \;\; c_2^2c_3\dot\theta_2\dot\theta_3 \;\; s_2c_2c_3^2\dot\theta_2\dot\theta_4 \;\; s_3c_2^2c_3\dot\theta_3\dot\theta_4 \;\; c_3\dot\theta_2\dot\theta_3\big]^\top\end{aligned} \tag{6.20}$$

This yields a truncated Koopman operator $K_1 \in \mathbb{R}^{34\times 34}$, where the subindex is used to denote its correspondence to Ψ_1. In order to explore the potential of the approach in cases in which the nonlinear model is not available, a second set of basis functions is proposed. These functions are derived using physical intuition: as it is expected that the contribution of the flywheel dominates the other Coriolis terms, only products of $\dot\theta_1$ with the rest of the velocities are considered, these products are then multiplied by trigonometric functions of the angles θ_2 and θ_3 (since, again from physical intuition it is clear that the rotations θ_1 and θ_4 do not have an effect on the dynamics) so that

$$\begin{aligned}\Psi_2 = \big[&x^\top \;\; u^\top \; c_2\dot\theta_1\dot\theta_2 \;\; c_2\dot\theta_1\dot\theta_3 \;\; c_2\dot\theta_1\dot\theta_4 \;\; c_3\dot\theta_1\dot\theta_2 \;\; c_3\dot\theta_1\dot\theta_3 \\ &c_3\dot\theta_1\dot\theta_4 \;\; s_2\dot\theta_1\dot\theta_2 \;\; s_2\dot\theta_1\dot\theta_3 \;\; s_2\dot\theta_1\dot\theta_4 \;\; s_3\dot\theta_1\dot\theta_2 \;\; s_3\dot\theta_1\dot\theta_3 \;\; s_3\dot\theta_1\dot\theta_4\big]^\top.\end{aligned} \tag{6.21}$$

yielding $K_2 \in \mathbb{R}^{21\times 21}$.

6.4.2 Computation and Update of the Koopman Operator

The Koopman operator is computed recursively using the procedure described in Section 6.2.1. Before starting the predictive controller, a short open-loop experiment[3] is performed using chirp test signals to bootstrap the Koopman algorithm and give a meaningful initial model to the MPC algorithm. The response of the open-loop experiment when using both basis functions Ψ_1, Ψ_2 can be seen in Figure 6.3, where the bottom plot shows when data is added (cf. Eq. (6.12)) to enrich the model.

For this training experiment, the parameter ν from (6.12) is set to $\nu = 0.001$ to encourage extracting information from it. When the controller is active, the factor is set to $\nu = 0.0025$ to avoid adding too many data points, thus rendering the model resistant to future updates.

6.4.3 Predictive Controller

A predictive controller using qLMPC (algorithm in Figure 3.1a), using a model in the form of (6.14) determined as described in Section 6.3 is used to control the CMG. For this, a sampling time of $T_s = 0.01$ s is used. This sampling time is also used for the sampling/update of the Koopman operator, so that both tasks (model update and control) are performed sequentially. For the predictive controller, a horizon $N = 30$ is chosen and the tuning parameters

[3]Before every experiment, both open- and closed-loop, a PI controller is used for 15 s to bring the flywheel up to speed, with brakes applied on all DOF to give an initial condition for the closed-loop experiments of $\omega_1(15) = 40$ rad/s, $\theta_i(15), \omega_i(15) = 0$, $i = 2, 3, 4$. At $t = 15$ s control authority is switched to the proposed controller.

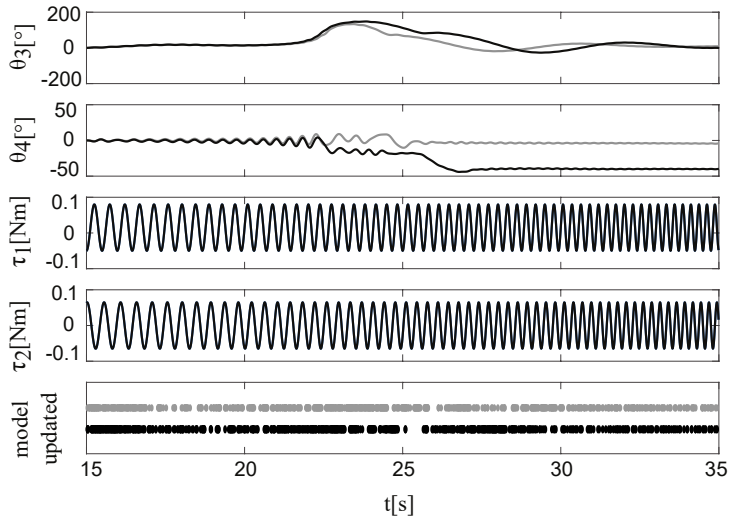


Figure 6.3: Open-loop training experiment

are $Q = \text{diag}(1, 120, 120, 0.01, 5, 2, 2, 0, 0, 0, 0, 0, 0, 0)$, $T = 10Q$, $R = \text{diag}(3000, 750)$ and the constraints on the inputs are $|\tau_1| < 0.5$Nm, $|\tau_2| < 2$Nm. For these experiments, no stabilizing terminal constraints are used.

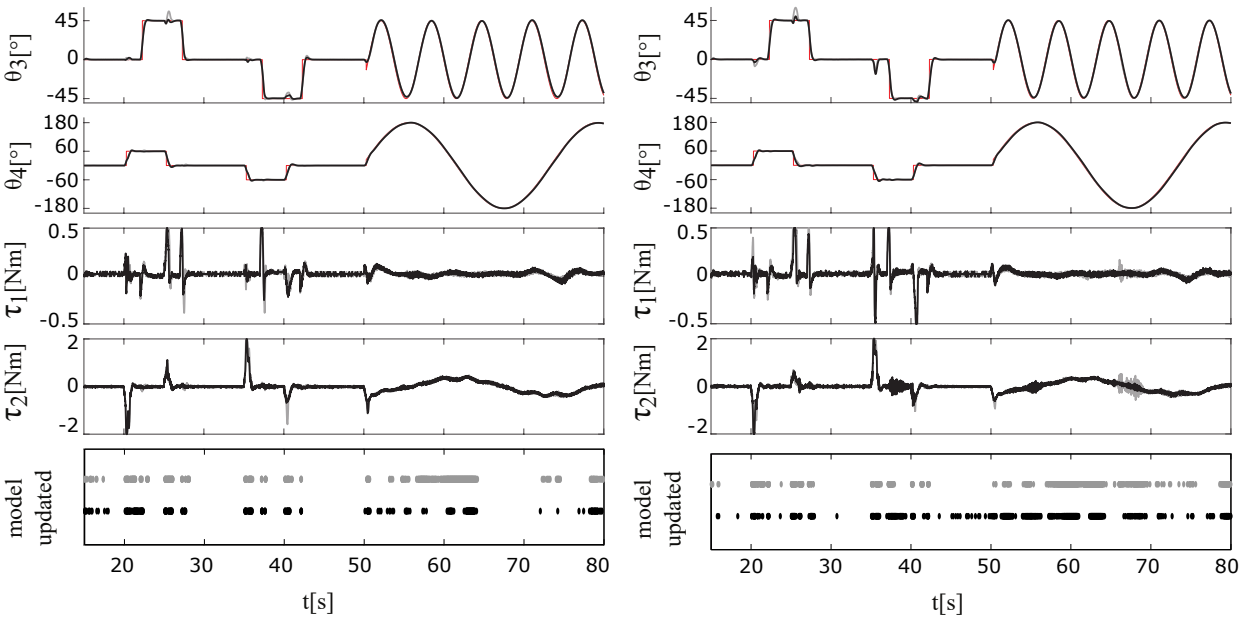
Closed-loop results are shown in Figures 6.4a and 6.4b, for the case where Ψ_1 and Ψ_2 are used, respectively. Each plot shows two experiments: one performed starting with the model after the training experiment, and a second one starting with the model after the first closed-loop experiment, in order to evaluate if performance improves after each iteration of a repetitive task. As expected, in both cases this is indeed the case, showing also that the second experiment needs to add fewer data points to the model, as there is less new information.

Comparing both controllers, the one based on K_1 has better performance, particularly regarding cross-coupling at $t = 27$ s and $t = 37$ s, which is again expected, given that the basis functions more closely resemble the equations of motion. It is however worth mentioning that both reference and tuning are selected to encourage an aggressive response as can be seen by the fact that inputs are driven into saturation, it is therefore remarkable that even with relatively simple basis functions, namely Ψ_2, the controller displays exceptional tracking performance.

6.4.4 Input-Output Controller

As mentioned earlier, assuming knowledge of the full state vector is in some cases unrealistic and one has to rely on observers, or numerical differentiation as performed above for all angular velocities. Alternatively, output feedback controllers can be realized via the input-output framework; indeed the effectiveness of the IO variant of Koopman-based qLMPC presented in Section 6.3 is examined in this section. To this end, define the output as

$$y(k) = [\theta_2(k) \quad \theta_3(k) \quad \theta_4(k) \quad \omega_1(k)]^\top$$



(a) Closed-loop experiment using K_1

(b) Closed-loop experiment using K_2

Figure 6.4: Closed-loop experiments. Reference (-----), first iteration (——), second iteration (——).

where $\omega_1 = \dot{\theta}_1$. Correspondingly, assuming second order dynamics the state vector can be defined as

$$x_k = \begin{bmatrix} y_k \\ y_{k-1} \end{bmatrix}.$$

The reason why q_2 and ω_1 were included in the output is twofold: firstly, in order to be able to capture the dynamics of the CMG, these variables play an important role; secondly, even though the objective is to track θ_3, θ_4, it is also desired to keep θ_2 and ω_1 close to their nominal values, to avoid reaching the physical limits of the plant.

The basis functions used are the discretized version of Ψ_2, where angular velocities are captured using finite differences by replacing $\omega_1(k)\dot{\theta}_i(k)$ with $[\omega_1(k)\theta_i(k) \quad \omega_1\theta_i(k-1)]$, $i = 2, 3$ resulting in

$$\begin{aligned}
\Psi_{2,IO} = \big[& x(k)^\top \quad u(k)^\top \\
& c_2\omega_1(k)\theta_2(k) \quad c_2\omega_1(k)\theta_2(k-1) \quad c_2\omega_1(k)\theta_3(k) \quad c_2\omega_1(k)\theta_3(k-1) \\
& c_2\omega_1(k)\theta_4(k) \quad c_2\omega_1(k)\theta_4(k-1) \quad c_3\omega_1(k)\theta_2(k) \quad c_3\omega_1(k)\theta_2(k-1) \\
& c_3\omega_1(k)\theta_3(k) \quad c_3\omega_1(k)\theta_3(k-1) \quad c_3\omega_1(k)\theta_4(k) \quad c_3\omega_1(k)\theta_4(k-1) \\
& s_2\omega_1(k)\theta_2(k) \quad s_2\omega_1(k)\theta_2(k-1) \quad s_2\omega_1(k)\theta_3(k) \quad s_2\omega_1(k)\theta_3(k-1) \\
& s_2\omega_1(k)\theta_4(k) \quad s_2\omega_1(k)\theta_4(k-1) \quad s_3\omega_1(k)\theta_2(k) \quad s_3\omega_1(k)\theta_2(k-1) \\
& s_3\omega_1(k)\theta_3(k) \quad s_3\omega_1(k)\theta_3(k-1) \quad s_3\omega_1(k)\theta_4(k) \quad s_3\omega_1(k)\theta_4(k-1) \big]^\top .
\end{aligned} \tag{6.22}$$

and correspondingly $K_{IO} \in \mathbb{R}^{33\times 33}$. Clearly a disadvantage of this approach is that it often leads to more basis functions. The tuning matrices corresponding to the outputs are discretized

so that $Q = \text{diag}(1, 120, 120, 0.01, 50000, 20000, 20000, 0, 0, 0, 0, 0)$ (using the approximation $\dot{\theta}^2 = \Delta\theta^2/T_s^2$ with $T_s = 0.01$). The same prediction horizon $N = 30$ is used in the IO case.

The result of the experiment is shown in Figure 6.5, where compared to its state-space counterpart in Figure 6.4b it can be seen that the performance is comparable, with relatively slower rise-time but noticeably less cross-coupling.

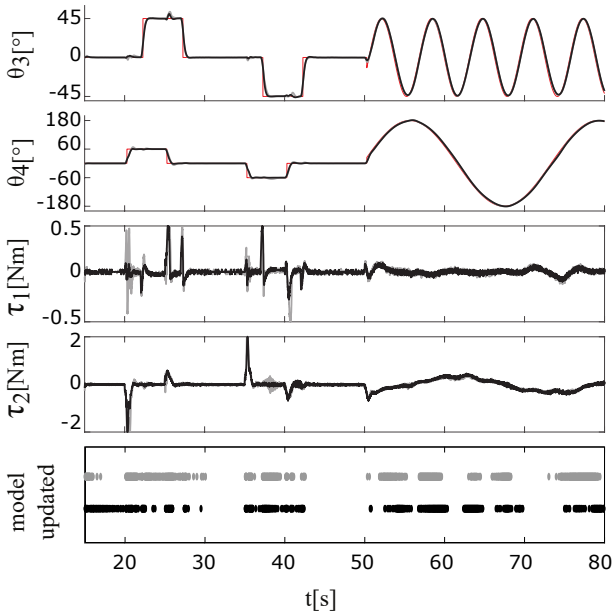


Figure 6.5: Closed-loop experiment using K_{IO}. Reference (- - - -), first iteration (——), second iteration (——).

6.5 Summary

The Koopman operator framework proves to be an effective and efficient way to discover the dynamics of an unknown system and simultaneously use the resulting model for data-driven control. This framework assumes little a priori knowledge of the underlying system in the form of basis functions; in the state space variant it also assumes that the state is measurable. In order to relax the latter assumption, the input-output framework can be used in a manner similar to how it is presented in Chapter 5 by defining a state vector containing past input and output data. The assumption is thus replaced by a milder assumption regarding system's order.

In both SS and IO variants, the velocity algorithm presented in Chapter 4 is used to obtain a qLPV model suitable for the qLMPC framework and with all the benefits of the velocity approach (i.e. integral action, parameterization-free tracking of unreachable set points) except for stability guarantees. Stability in this context is hard to establish since there is no guarantee that the basis functions correspond to the system, as they are design parameters. Nevertheless as was shown in the application example, even with few basis functions, outstanding closed-loop performance is attained. As mentioned in Remark 6.1, the motivation for using velocity linearization in lieu of the linear Koopman model is that the model's order is usually substantially lower, making online learning comparatively less computationally expensive.

Chapter 7

Stability Analysis of LPV MPC via Dissipativity

Throughout this work, as in most of the MPC literature, stability is established by imposing so-called *stabilizing constraints*. These constraints are fictitious in the sense that they are artificially imposed (i.e. they do not represent physical limitations of the system) for the sole purpose of establishing analysis results, and often take the form of terminal conditions, as in the present context, but in other cases trajectory constraints are imposed to force the system to behave in a certain way which lends itself to stability analysis. One such example is [96], where the closed-loop system is forced to behave passively, or its extension [130] where the condition is relaxed to a general dissipativity constraint. On this note, dissipativity has been proven to be strongly linked to optimizing controllers and their, under some conditions, optimal steady-state operation [89], although as mentioned in the latter work, finding these conditions is in general a difficult task. This chapter proposes an approach to establish stability of LPV predictive control laws a priori – that is, without the need to impose artificial stabilizing constraints – via a dissipation inequality. The presented analysis tool is given for systems which fulfill certain conditions to be mentioned in the first section. Indeed, under these specific conditions, stability can be guaranteed by existence of a storage function to be found via the solution to a convex optimization problem. The reader who is familiar with the framework of Integral Quadratic Constraints (IQC) will recognize many of the derivations in this chapter, particularly those paralleled to [108] where the relationship between IQCs and dissipativity is discussed; indeed, in a similar fashion, dissipativity in the context of this thesis is paralleled to a concept similar to IQCs deemed Parameter-Dependent Quadratic Constraints (PDQC).

To make this chapter self-contained, Section 7.1 recalls a result from the literature upon which the rest of the chapter is based. Section 7.2 extends this result to the LPV case and introduces the notion of parameter-dependent quadratic constraints. Section 7.3 develops the stability analysis tool and finally Section 7.4 illustrates the approach on a numerical example, ending the chapter with a summary in Section 7.5.

7.1 Quadratic Programming as a Sector-Bounded Nonlinearity

Before proceeding to the LPV case, a review of the foundation of the results to follow is given. This section strongly borrows from its eponym in [56], where the observation was first made that under some conditions, a quadratic program fulfills a sector condition in the sense of the following definition

Definition 7.1. [67] Assume $K_2 - K_1 \succ 0$ for a given $K_1, K_2 \in \mathbb{R}^{n\times n}$, the function $\phi : \mathbb{R}^n \to \mathbb{R}^n$ is said to belong to the sector $[K_1 \;\; K_2]$ if

$$(\phi(u) - K_1 u)^\top (\phi(u) - K_2 u) \leq 0 \quad \forall u \in \mathbb{R}^n \tag{7.1}$$

A geometric interpretation of (7.1) in the SISO case, where the condition reduces to $k_1 u^2 \leq u\phi(u) \leq k_2 u^2$ is shown in Figure 7.1. Indeed the nonlinearity belongs to the sector if it is always contained within the shaded region, bounded by the lines with slopes k_1, k_2.

Consider the quadratic program obtained from a dense formulation of LTI MPC as in (2.9), i.e.

$$\begin{aligned} \phi(g) = \arg\min_U \; & \frac{1}{2} U^\top H U - U^\top g \\ \text{subject to} \\ & AU \leq b \end{aligned} \tag{7.2}$$

where $\phi : \mathbb{R}^{Nm} \to \mathbb{R}^{Nm}$ is a function characterizing the QP; note that the function ϕ maps the gradient of the cost function into the optimal solution U^*. The Karush-Kuhn-Tucker conditions for the QP are given by

$$H\phi(g) + A^\top \lambda - g = 0 \tag{7.3a}$$
$$A\phi(g) + s = b \tag{7.3b}$$
$$s^\top \lambda = 0 \tag{7.3c}$$
$$s, \lambda \geq 0 \tag{7.3d}$$

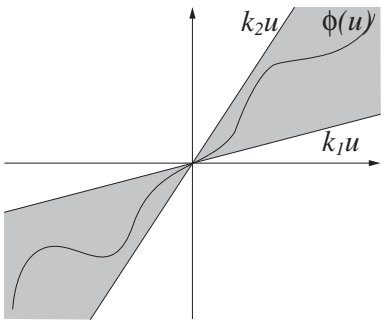


Figure 7.1: Sector bounded nonlinearity

Pre-multiplying the stationarity condition (7.3a) by $\phi(g)^\top$ yields

$$\phi(g)^\top H\phi(g) + \phi(g)^\top A^\top \lambda - \phi(g)^\top g = 0.$$

Substituting the slackness condition (7.3b) into the equation above gives

$$\phi(g)^\top H\phi(g) + (b^\top - s^\top)\lambda - \phi(g)^\top g = 0,$$

the complementarity condition (7.3c) can then be substituted to obtain

$$\phi(g)^\top H\phi(g) - \phi(g)^\top g = -b^\top \lambda.$$

Finally making the assumption that $b \geq 0$ the following condition is obtained

$$\phi(g)^\top (H\phi(g) - g) \leq 0. \tag{7.4}$$

It can be shown that (7.4) is equivalent to a sector condition and two linear transformations ([56]). Note that the relatively strong assumption that $b \geq 0$ was made. This is equivalent to assuming that 0 is always a feasible solution, which might preclude the optimization problem from including state constraints (cf. Eq. 2.9) and even non-symmetric input constraints.

7.2 Extension to LPV

This section extends the results above to the LPV case, where the optimization problem takes the form

$$\begin{aligned} \phi(g, \mathsf{P}) = \min_U \ & \frac{1}{2} U^\top H(\mathsf{P}) U + g(\mathsf{P})^\top U \\ & \text{subject to} \\ & AU \leq b \end{aligned} \tag{7.5}$$

clearly the dependence of H and g on P requires special attention since condition (7.4) is now parameter-dependent. To this end a characterization of Parameter-Dependent Quadratic Constraints (PDQC) is presented below.

7.2.1 Parameter-Dependent Quadratic Constraints

A promising and somewhat recent development which unifies classical and modern stability analysis results under the same umbrella is the Integral Quadratic Constraints (IQC) framework [84]. Whereas the problem at hand can be tackled under this framework, the structure of it is such that an equivalent result can be more easily obtained using dissipativity arguments. To this end and inspired by IQCs, the concept of parameter-dependent quadratic constraints is introduced

Definition 7.2. A bounded operator $\Delta(\rho) : \ell_{2e}^{l} \rightarrow \ell_{2e}^{m}$ is said to satisfy a parameter-dependent quadratic constraint (PDQC) defined by Π if $\forall \rho \in \mathcal{P}, \forall v \in \ell_2^l[0,\infty], w = \Delta(\rho, v)$

$$\begin{bmatrix} v \\ w \end{bmatrix}^* \Pi(\rho) \begin{bmatrix} v \\ w \end{bmatrix} \geq 0 \tag{7.6}$$

for a given compact set $\mathcal{P}$ of admissible parameters.

Borrowing from the IQC literature the notation $\Delta(\rho) \in PDQC(\Pi(\rho))$ is adopted to denote that the operator $\Delta(\rho)$ satisfies a PDQC as defined in Definition 7.2. The attentive reader may notice that the PDQC holds point wise in time, it therefore fulfills (if integrated) what some authors refer to as *hard IQC* ([108]) which facilitates the derivation of stability conditions.

Inspired by the J-spectral factorization frequently done in the context of IQCs [84], to express frequency domain IQCs in time domain, a factorization of the form $\Pi(\rho) = \Psi(\rho)^* M \Psi(\rho)$ can be performed to express the PDQC (7.6) as

$$z^\top M z \geq 0 \tag{7.7}$$

where z is the output of the bounded, linear parameter-varying operator $\Psi(\rho)$ and M is constant. This has the advantage that condition (7.7) is now parameter-independent. A graphical interpretation of the parameter-independent characterization of the quadratic constraint is given in Figure 7.2: the input and output of $\Delta(\rho)$ are filtered through $\Psi(\rho)$, yielding the equivalence condition of the following Lemma.

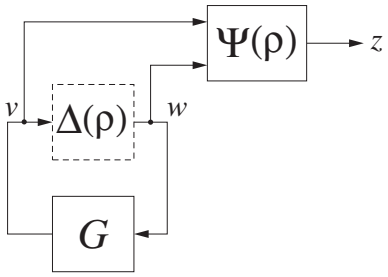


Figure 7.2: Graphical interpretation of the parameter-independent characterization of $\Delta(\rho)$

Lemma 7.1. Given a parameter-dependent bounded operator $\Delta(\rho)$ and a multiplier $\Pi(\rho)$, together with a factorization $\Pi(\rho) = \Psi(\rho)^* M \Psi(\rho)$ the following statements are equivalent:

i) $\Delta(\rho) \in \text{PDQC}(\Pi(\rho))$

ii) $z^\top M z \geq 0$

for $z = \Psi(\rho) \begin{bmatrix} v \\ \Delta(\rho, v) \end{bmatrix}$.

Proof. Obvious by substituting $\Pi(\rho) = \Psi(\rho)^* M \Psi(\rho)$ and $\Psi(\rho) \begin{bmatrix} v \\ \Delta(\rho, v) \end{bmatrix} = z$ in (7.6). □

Note that the input-output behavior of $\Delta(\rho)$ is described by the PDQC, as a consequence, the operator can be neglected on the subspace where the constraint is met (hence the dashed line in Figure 7.2).

Remark 7.1. *Lemma 7.1 establishes equivalence of the PDQC and its parameter independent characterization, however this does not imply that the statement is free of conservatism. Indeed, usually the set of operators that satisfy the PDQCs (or the IQCs) is larger than the set of operators being characterized [61] by the PDQC; in this case the set of operators being characterized is a singleton containing $\phi(g)$.*

7.2.2 Parameter-Dependent Predictions

As mentioned above, optimization problems in the form of (7.5) arise in MPC for LPV systems, in particular (but not exclusively) in qLMPC, treated throughout this thesis. In order to obtain a parameter-independent characterization of the PDQC arising from solving a parameter-dependent QP, a brief review of the discussion in Chapter 3 is made. Consider a quasi-LPV model

$$\bar{G}(\rho_k) = \begin{cases} x_{k+1} = & A(\rho_k)\, x_k + B(\rho_k)\, u_k \\ y_k = & I x_k, \end{cases} \tag{7.8}$$

future state trajectories are given by

$$X_k = \Phi(\mathsf{P}_k) x_k + \Gamma(\mathsf{P}_k) U_k, \tag{7.9}$$

where

$$X_k = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N} \end{bmatrix} \in \mathbb{R}^{Nn} \qquad U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ x_{k+N-1} \end{bmatrix} \in \mathbb{R}^{Nm} \qquad \mathsf{P}_k = \begin{bmatrix} \rho_k \\ \rho_{k+1} \\ \vdots \\ \rho_{k+N-1} \end{bmatrix} \in \mathbb{R}^{Nn_\rho}$$

$$\Phi(\mathsf{P}_k) = \begin{bmatrix} A(\rho_k) \\ A(\rho_{k+1})A(\rho_k) \\ \vdots \\ A(\rho_{k+N-1})A(\rho_{k+N-2})\ldots A(\rho_k) \end{bmatrix}$$

$$\Gamma(\mathsf{P}_k) = \begin{bmatrix} B(\rho_k) & 0 & \ldots & 0 \\ A(\rho_{k+1})B(\rho_k) & B(\rho_{k+1}) & \ldots & 0 \\ A(\rho_{k+2})A(\rho_{k+1})B(\rho_k) & A(\rho_{k+2})B(\rho_{k+1}) & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ A(\rho_{k+N-1})\ldots A(\rho_{k+1})B(\rho_k) & A(\rho_{k+N-1})\ldots A(\rho_{k+2})B(\rho_{k+1}) & \ldots & B(\rho_{k+N-1}) \end{bmatrix}.$$

Using these definitions, it was shown in Section 3.1 that the Hessian and gradient of the QP in the form (7.2) are given by

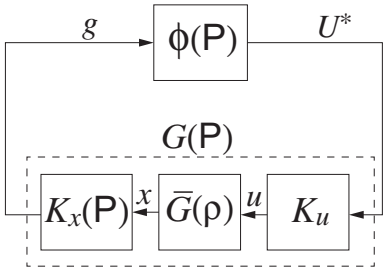


Figure 7.3: Parameter-dependent MPC loop

$$\begin{aligned} H(\mathsf{P}) &= 2\left(\hat{R} + \Gamma(\mathsf{P}_k)^\top \hat{Q}\Gamma(\mathsf{P}_k)\right) \\ g(\mathsf{P}, k) &= -2\Gamma(\mathsf{P}_k)^\top \hat{Q}\Phi(\mathsf{P}_k)x_k \end{aligned} \tag{7.10}$$

where $\hat{R} = I_N \otimes R$, $\hat{Q} = \mathrm{diag}(I_{N-1} \otimes Q, P)$. Note that only $g(\cdot)$ depends explicitly on k, given its dependence on x_k, however both H and g depend implicitly on time given their dependence on the time varying parameters ρ_{k+i}, $i = 0, 1, ..., N-1$. The MPC loop is thus closed by interconnecting the ϕ operator (defined in (7.5)) with the plant model $\bar{G}$ by defining the operators (see Figure 7.3)

$$\begin{aligned} K_u &= \begin{bmatrix} I_m & 0 & 0 & \dots 0 \end{bmatrix} \\ K_x &= -2\Gamma(\mathsf{P}_k)^\top \hat{Q}\Phi(\mathsf{P}_k) \end{aligned}$$

such that

$$G(\mathsf{P}_k) = K_x(\mathsf{P}_k)\bar{G}(\rho_k)K_u$$

where $\bar{G}$ is the model of the plant given in (7.8). Following the discussion above, in the LPV case $\phi(\mathsf{P}) \in \mathrm{PDQC}(\Pi(\mathsf{P}))$ where

$$\Pi(\mathsf{P}) = \begin{bmatrix} 0 & I \\ I & -2H(\mathsf{P}) \end{bmatrix} \tag{7.11}$$

represents the parameter-dependent equivalent to condition (7.4), in the sense of Definition 7.2.

Lifting parameter dependence

Both the model G and the quadratic constraint characterized by the multiplier[1] (7.11) depend on future parameter values, given their dependence on $\Phi(\mathsf{P}_k)$ and $\Gamma(\mathsf{P}_k)$. This can be regarded as dynamic dependence on parameters which is generally a problem when attempting to derive stability conditions. One possible way to handle this issue at the expense of conservatism is by

[1]The notion of multiplier is used in the sense analogous to IQCs, i.e. the coefficient matrix M in a quadratic form $x^\top Mx$

lifting the parameter space, i.e. define a new set of parameters such that

$$\mathsf{P} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_N \end{bmatrix}$$

and each $\rho_i \in \mathbb{R}^{n_\rho}$, $i = 1, 2, ..., N$ is independent of the other parameters, this is equivalent to assuming infinitely fast varying parameters. Along with the lifted parameters, the lifted compact set $\hat{\mathcal{P}} \subset \mathbb{R}^{Nn_\rho}$ is defined, such that

$$\rho_k \in \mathcal{P} \quad k = 0, 2, ..., N-1 \quad \Leftrightarrow \quad \mathsf{P} \in \hat{\mathcal{P}}.$$

7.3 Stability Analysis

In order to derive stability conditions for the closed-loop in Figure 7.3, an important stepping stone is the parameter-independent characterization (Lemma 7.1) of the quadratic constraint related to the multiplier (7.11). To this end, note that the parameter-dependent Hessian in Equation (7.10) can be straightforwardly factorized as

$$H(\mathsf{P}) = \begin{bmatrix} \Gamma(\mathsf{P}) \\ I \end{bmatrix}^\top \begin{bmatrix} 2\hat{Q} & 0 \\ 0 & 2\hat{R} \end{bmatrix} \begin{bmatrix} \Gamma(\mathsf{P}) \\ I \end{bmatrix}$$

such that the PDQC can equivalently be written as

$$\begin{bmatrix} * \end{bmatrix}^\top \underbrace{\left[\begin{array}{c|cc} 0 & 0 & I \\ \hline 0 & -4\hat{Q} & 0 \\ I & 0 & -4\hat{R} \end{array}\right]}_{M} \underbrace{\left[\begin{array}{cc} I & 0 \\ \hline 0 & \Gamma(\mathsf{P}) \\ 0 & I \end{array}\right]}_{\Psi(\mathsf{P})} \begin{bmatrix} v_k \\ w_k \end{bmatrix} \geq 0 \qquad \forall v_k, w_k = \phi(v_k), \forall \mathsf{P} \in \hat{\mathcal{P}}. \tag{7.12}$$

which in light of Lemma 7.1, letting $z = \Psi(\mathsf{P})[v_k^\top \;\; w_k^\top]^\top$ is equivalent to

$$z^\top \left[\begin{array}{c|cc} 0 & 0 & I \\ \hline 0 & -4\hat{Q} & 0 \\ I & 0 & -4\hat{R} \end{array}\right] z \geq 0. \tag{7.13}$$

Having derived the parameter independent constraint (7.13) and following the discussion in Section 7.2.1 the operator $\phi(\mathsf{P})$ can be *replaced* as shown in Figure 7.4a on the subspace defined by (7.12). The plant $G(\mathsf{P})$ can be augmented with the parameter-dependent filter $\Psi(\mathsf{P})$ to yield $G_\Psi(\mathsf{P})$; by performing a Linear Fractional Transformation (LFT) it is possible to express $G_\Psi(\mathsf{P})$ as the interconnection of a nominal, LTI plant G_Ψ^o and a scheduling block Θ as seen in Figure 7.4b. A state space model of the nominal augmented plant is given by

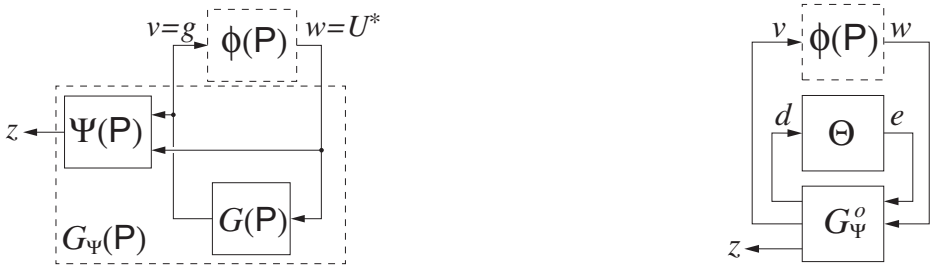


Figure 7.4: (a) Plant augmented with filter (b) LFT of augmented plant

$$G_{\Psi}^{o} \begin{cases} x_{k+1} = A^{o}x_k + B_w^o w_k + B_e^o e_k \\ z = C_z^o x_k + D_{zw}^o w_k + D_{ze}^o e_k \\ d = C_d^o x_k + D_{dw}^o w_k + D_{de}^o e_k \end{cases}$$

where the superscript ^{o} denotes the nominal LTI part of the plant after extracting parameter dependencies into the LFT block Θ. Note that the LFT is done for the augmented plant, so that parameter dependency of both $G(\rho)$ and $\Psi(\mathsf{P})$ is now extracted into the Θ-block. As the filter $\Psi(\mathsf{P})$ is static (i.e. it does not have dynamics) the order of the augmented plant is the same as the original one, albeit with more input-output channels.

7.3.1 Dissipation Inequality Formulation

The concept of dissipativity [123] extends the classical Lyapunov methods to systems with inputs. Roughly speaking, dissipativity characterizes the energy flow of a system with respect to a given supply. As such, it generalizes the concept of passivity by considering a richer family of supply functions.

Definition 7.3 (Dissipativity [123])**.** A dynamic system with input $u(t) \in \mathbb{R}^m$, output $y(t) \in \mathbb{R}^l$ and state $x(t) \in \mathbb{R}^n$ is said to be dissipative w.r.t a *supply* function $s : \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}$ if there exists a nonnegative *storage* function $V : \mathbb{R}^n \to \mathbb{R}$ such that for all $t_1, t_0 \in \mathbb{R}$, $x_0 \in \mathbb{R}^n$, $u \in \mathbb{R}^m$

$$V(x_1) - V(x_0) \leq \int_{t_0}^{t_1} s(u(t), y(t))dt$$

where x_1 is the state at t_1 reached from initial state x_0 at t_0 by applying the input trajectory $u(t)$.

Intuitively, the definition above can be understood as: *the change in energy stored in the system is less than the energy supplied to the system*, said otherwise, the system is *dissipating* part of the supplied energy. Clearly this concept generalizes the usual Lyapunov analysis in which an energy-like function is used similarly to the storage function, but without considering a supply rate.

The interest in using a standard dissipativity formulation, as opposed to the IQC formulation in [55], is that it offers the advantage that there are fewer assumptions about the system and the nonlinearity arising from the QP (in particular the homotopy condition disappears). The result, however, shares the same structure and conditions for stability can be efficiently evaluated by solving LMIs.

Before stating the main result of this chapter, a small detour is made to present a useful tool which enables the derivation of LMI conditions for stability, particularly when the system possesses a large number of scheduling parameters.

Lemma 7.2 (Full-block $\mathcal{S}$-procedure for LFTs [103][129])**.** Given parameters ρ confined to a compact set $\rho \in \mathcal{P}$, the matrix inequality

$$F(\rho)^\top Q F(\rho) \preceq 0 \quad \forall \rho \in \mathcal{P}$$

with $F(\rho) = \mathcal{F}_u\left(\begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}, \Theta(\rho)\right)$ holds if and only if there exists a real symmetric matrix M_Θ, s.t.

$$\begin{bmatrix} * \end{bmatrix}^\top Q \begin{bmatrix} F_{21} & F_{22} \end{bmatrix} + \begin{bmatrix} * \end{bmatrix}^\top M_\Theta \begin{bmatrix} F_{11} & F_{12} \\ I & 0 \end{bmatrix} \prec 0 \tag{7.14}$$

and

$$\begin{bmatrix} * \end{bmatrix}^\top M_\Theta \begin{bmatrix} I \\ \Theta(\rho) \end{bmatrix} \succeq 0 \quad \forall \rho \in \mathcal{P}. \tag{7.15}$$

Lemma 7.2 proves useful for deriving LMIs from a parameter-dependent (or uncertain) inequality for which an LFT can be derived (i.e. for systems with rational parameter dependence); unfortunately, its tractability suffers from the semi-infinite dimensional condition (7.15). Such a condition can be approximated by gridding, however, for a large number of parameters gridding might prove prohibitive. A simplification can be carried out by imposing a structure on the multiplier M_Θ which trivially fulfills (7.15), at the expense of conservatism. One such structure is the D/G-scalings presented next.

D/G-Scalings

The idea of D/G scalings is to impose structural conditions on the multiplier to improve tractability of the problem. These have been used in μ-analysis since the early 1990's, the scaled version presented here is due to [107]. Let

$$\Theta = \text{diag}(\rho_1 I_{r1}, \rho_2 I_{r2}, ..., \rho_{Nn_\rho} I_{rNn_\rho}), \quad \rho_i \in \mathbb{R}, \quad |\rho_i| \leq 1.$$

The set of $D/G - Scalings$ is defined as

$$\mathcal{M}_{D/G} = \left\{ \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^\top & -M_{11} \end{bmatrix} : M_{11} = M_{11}^\top \succ 0, M_{12} + M_{12}^\top = 0, \right.$$
$$\left. M_{11}\Theta = \Theta M_{11}, M_{12}\Theta = \Theta M_{12} \right\}.$$

By using a multiplier $M_\Theta \in \mathcal{M}_{D/G}$, condition (7.15) is trivially fulfilled and the LMI problem becomes parameter independent. The main result of this chapter is presented in the following theorem.

Theorem 7.1. System (7.8) in closed-loop with an MPC law in the form of (7.5) is quadratically stable if $\exists P = P^\top > 0$, $\tau > 0$ such that

$$\begin{bmatrix} A^{o\top}PA^o - P & A^{o\top}PB^o_w & A^{o\top}PB^o_e \\ B^{o\top}_w PA^o & B^{o\top}_w PB^o_w & B^{o\top}_w PB^o_e \\ B^{o\top}_e PA^o & B^{o\top}_e PB^o_w & B^{o\top}_e PB^o_e \end{bmatrix} + \tau\Pi_\phi + \Pi_\Theta \prec 0 \tag{7.16}$$

where

$$\Pi_\phi = \left[*\right]^\top \begin{bmatrix} 0 & 0 & I \\ 0 & -4\hat{Q} & 0 \\ I & 0 & -4\hat{R} \end{bmatrix} \begin{bmatrix} C^o_z & D^o_{zw} & D^o_{ze} \end{bmatrix},$$

$$\Pi_\Theta = \left[*\right]^\top M_\Theta \begin{bmatrix} C^o_d & D^o_{dw} & D^o_{de} \\ 0 & 0 & I \end{bmatrix}$$

and $M_\Theta \in \mathcal{M}_{D/G}$.

Proof. Define the quadratic storage function $V = x^\top Px$, pre- and postmultiplying equation (7.16) by $\begin{bmatrix} x^\top & w^\top & e^\top \end{bmatrix}$ and $\begin{bmatrix} x^\top & w^\top & e^\top \end{bmatrix}^\top$ respectively, yields the dissipation inequality

$$V_{k+1} - V_k + z_k^\top M z_k + \begin{bmatrix} d_k \\ e_k \end{bmatrix}^\top M_\Theta \begin{bmatrix} d_k \\ e_k \end{bmatrix} < 0.$$

where the supply function is defined as $\int_k^{k+1} s(u,y) = -z_k^\top M z_k - \begin{bmatrix} d_k \\ e_k \end{bmatrix}^\top M_\Theta \begin{bmatrix} d_k \\ e_k \end{bmatrix}$. Substituting $e_k = \Theta d_k$ yields

$$V_{k+1} - V_k + z_k^\top M z_k + d_k^\top \begin{bmatrix} I \\ \Theta \end{bmatrix}^\top M_\Theta \begin{bmatrix} I \\ \Theta \end{bmatrix} d_k < 0.$$

As $\phi \in \text{PDQC}(\Pi_\phi)$ condition (7.13) is met and given the structure of M_Θ, $d_k^\top \begin{bmatrix} I \\ \Theta \end{bmatrix}^\top M_\Theta \begin{bmatrix} I \\ \Theta \end{bmatrix} d_k \geq 0$, therefore

$$V_{k+1} - V_k < 0.$$

So the storage function defines a Lyapunov function and the origin of the state-space is (locally) asymptotically stable. □

Remark 7.2. *Although the LMI* (7.16) *is parameter independent, the admissible parameter set* $\hat{\mathcal{P}}$ *arises when computing the LFT of* $G_\Psi(\mathsf{P})$ *since an important assumption when using* D/G*-scalings is that* $|\rho_i| \leq 1$ *so the LFT needs to be scaled accordingly.*

The assumption made in Section 7.2.2 regarding the lifted parameters, makes the analysis from Section 7.3 also suitable for LPV systems with exogenous parameters. However, in this case the

prediction (and by extension the online optimization) would look different than that of equation (7.9), as in this case the future scheduling trajectory cannot be computed, so a worst case optimization is done instead [68][79]. In the nonlinear case (quasi-LPV), Theorem 7.1 is a local result in the sense that the admissible set $\mathcal{P}$, or more specifically its image in $\mathbb{R}^n$, represent a subset of the region of attraction. Although the fact that this represents a local result could be regarded as a limitation, it is in practice a feature, since inherited from the LTI result in [55], the result only applies to open-loop stable plants; in the nonlinear case the fact that it is a local result enables it to be applied to locally open-loop stable equilibria, as opposed to the much more restrictive set of globally open-loop stable systems.

7.4 Numerical Example

This section illustrates the approach presented in this chapter on a numerical simulation example. Consider the model given in Example 3.2,

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 4/3 + 0.2x_1(k) & -2/3 + 0.1x_2(k) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0 \end{bmatrix} u_k$$
$$y(k) = \begin{bmatrix} -2/3 & 1 \end{bmatrix}.$$

Recall that the origin of the system is locally open-loop stable but given the input constraint, cannot be globally stabilized. The system has unstable zero dynamics so it is expected that an aggressive predictive controller with a short horizon (without stabilizing terminal constraints) destabilizes the system by attempting to invert the zero [98]. An interesting question is therefore, how short can the horizon be such that the closed-loop is guaranteed to be stable? and within which region could stability be guaranteed? These questions can be answered using the analysis tool presented in this chapter.

In order to encourage zero inversion, the tuning parameters are chosen as $Q = C^\top C$, $R = 0.001$. Both the admissible parameter set and the horizon are heuristically found yielding values

$$\mathcal{P} \begin{cases} \rho_1 \in [-0.05 \quad 0.05] \\ \rho_2 \in [-0.1 \quad 0.1] \end{cases} \qquad N = 7$$

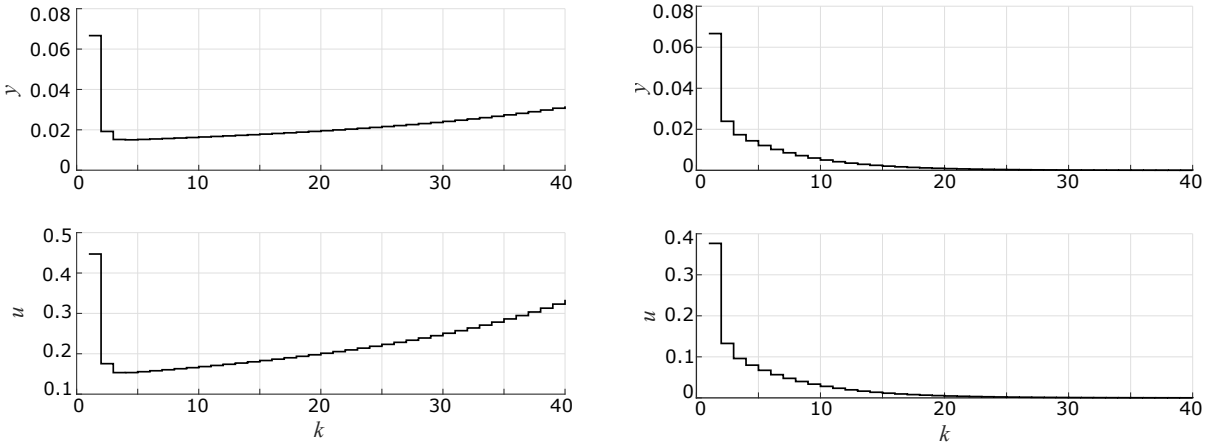
for which the solution to the LMI in Theorem 7.1 is[2]

$$P = \begin{bmatrix} 884.36 & -577.61 \\ -577.61 & 595.00 \end{bmatrix} \qquad \tau = 346.01.$$

The result can be interpreted as stating that within the region $[x_1 \;\; x_2]^\top = \rho \in \mathcal{P}$, a prediction horizon $N \geq 7$ guarantees that the closed-loop is stable. To get an idea of the conservatism of the approach, simulations can be carried out to examine if the result is far from reality,

[2] the multiplier $M_\Theta \in \mathbb{R}^{72\times 72}$ is not shown for space reasons

keeping in mind that simulations give a general idea of the stability of the closed-loop, yet stable closed-loop simulations do not, in any way, give stability guarantees. Simulations with different prediction horizons, initial conditions and saturation levels, reveal that for $N \leq 4$ the closed-loop is unstable (Figure 7.5), but every simulated scenario yields a stable response with $N \geq 5$. This suggests that for $N \geq 5$, the closed-loop is stable, which is not far from the lower bound $N = 7$ obtained by applying the methodology presented in this chapter.



(a) Closed-loop simulation with $N = 4$

(b) Closed-loop simulation with $N = 5$

Figure 7.5: Closed-loop simulations

Note that, apart from the assumption above Equation (7.1), i.e. $b \geq 0$, there is no other assumption made on the constraints. This assumption might prove restrictive if state constraints are to be considered; however, the assumption includes as a special case the practically relevant, and indeed ubiquitous, case of input constraints of the form $|u| \leq \bar{u}$ for any arbitrary $\bar{u} \geq 0$.

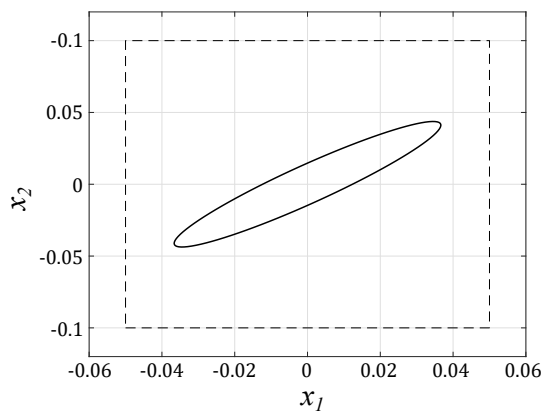
7.4.1 Comparison with Stabilizing Terminal Constraints

Even though being able to guarantee stability a priori is a strong motivation to consider this analysis tool, the result would not be attractive if it is considerably more conservative than what is considered *standard* in the MPC literature, that is, using stabilizing terminal constraints as presented in Chapters 3-5. In this section, a brief comparison is made in terms of the region of attraction and minimum prediction horizon. To this end, consider again the model from Example 3.2 with tuning matrices as selected above and admissible parameter region

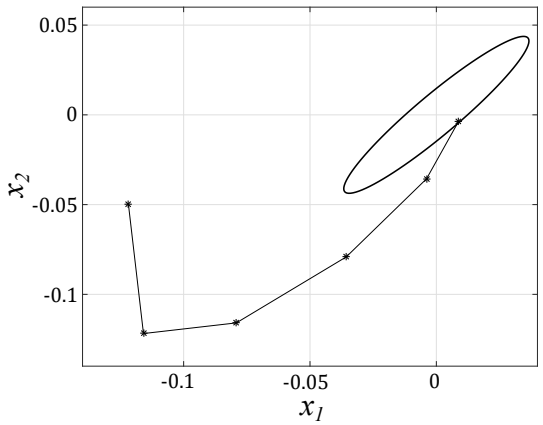
$$\mathcal{P}_2 \begin{cases} \rho_1 \in [-0.2 \quad 0.2] \\ \rho_2 \in [-0.2 \quad 0.2] \end{cases}$$

chosen as such to avoid limiting the size of the ellipsoid (recall Figure 3.4). Assuming a constraint $|u| \leq 0.1$ (necessary to compute the ellipsoidal terminal region), and for simplicity constant terminal ingredients, solution of the LMI problem in Theorem 3.1 yields the ellipsoid shown in Figure 7.6a; for comparison purposes, the region of attraction (corresponding to $\mathcal{P}$) from the approach in this chapter is also illustrated in the figure. Under these conditions, the gap between

both approaches in terms of feasibility and minimal horizon is small. Indeed, the optimization problem implementing stabilizing terminal constraints based on the computed ellipsoid with an initial condition $x(0) = [-0.05 \quad -0.1]$ becomes feasible for $N \geq 6$ (Figure 7.6b). It is worth mentioning that, although the result is comparable, the dissipativity-based approach gives a priori stability guarantees, meaning that the control law is simpler to implement as no additional constraints need to be introduced.



(a) Terminal constraint set (——), region of attraction from dissipativity approach (- - - - -)



(b) Predicted trajectory with stabilizing terminal constraint and $N = 6$

Figure 7.6: Comparison between dissipativity-based result and stabilizing terminal constraints

The approach presented throughout this chapter appears to be superior in this scenario. However, one disadvantage is that there is no way to include the input saturation level $\bar{u}$ in the analysis. As a consequence, if the saturation level is greater, say $\bar{u} = 1$, this approach seems to be overly conservative. Indeed, there would be no change in minimum horizon or region of attraction (as neither depend on the saturation level, and they hold for all possible saturation levels), but the ellipsoidal terminal region obtained with the standard approach would be much larger. On the other hand, a change in the opposite direction, i.e. $\bar{u} = 0.01$ would further benefit the dissipativity-based approach.

7.5 Summary

The analysis tool developed in this chapter provides the means to guarantee stability of nonlinear and parameter-dependent predictive control laws a priori, and without the need to implement stabilizing terminal constraints. This means that feasibility and performance are in no way affected by guaranteeing stability of the closed-loop. Analysis is based on the deemed parameter-dependent quadratic constraints, which, alternative to integral quadratic constraints, prove useful when dealing with parameter-dependent systems. The use of D/G-scalings and assuming infinitely fast changing parameters are the main sources of conservatism of the approach, yet as seen in the example section, conservatism could potentially be low, under some conditions. Important assumptions made are that the plant (more specifically the equilibrium of interest) is open-loop stable and that the constraints of the problem are such that the affine term of

the inequality constraint $Ax \leq b$ is piece-wise nonnegative, that is $b \geq 0$. This might prove prohibitive if there are state constraints.

In the example it was highlighted that under some conditions, the approach is superior to the standard stabilizing MPC (i.e. with stabilizing terminal constraints) but under other conditions it might prove overly conservative. As expected, there is no one superior approach that is better than others for all possible scenarios and having alternative tools at one's disposal seems to be the best bet. Indeed, although the dissipativity-based approach seems to be restrictive, should all conditions hold, it becomes extremely convenient, given that the control law is simple to implement, and both feasibility and stability are guaranteed.

Chapter 8

Conclusions and Outlook

8.1 Conclusions

The interest in fast Nonlinear Model Predictive Control stems from the desire to take advantage of the attractive features offered by MPC, particularly anticipative behavior and hard constraint satisfaction, to control complex nonlinear systems with fast dynamics, e.g. mechatronic, electrical systems, etc. However, there are two main challenges faced by predictive control laws, and they represent the main motivations for the proposed framework, these are:

1. Predictive controllers require the online solution of a constrained optimization problem within a sampling interval. In the nonlinear case, this optimization problem is non-convex and hard to solve.
2. Deriving tractable stability conditions for nonlinear systems is in general a difficult task, in the context of nonlinear MPC this is often done by linearizing the nonlinear system around the desired equilibrium and solving a convex optimization problem to derive the terminal ingredients.

To a somewhat lesser extent (at least in comparison to the other two), an additional challenge faced in nonlinear MPC is

3. NMPC requires an accurate yet as simple as possible model.

In order for predictions to be meaningful and in the interest of point number 2. above (i.e. to establish stability) an accurate model is needed; however, simpler models are preferred due to point number 1. above, i.e. complexity of the computations scale with model complexity. In practice a trade-off has to be made.

The quasi-Linear Model Predictive Control framework presented in this thesis addresses items 1.-3. in the following ways:

i. It provides an easy to implement, highly efficient algorithm for nonlinear Model Predictive Control by means of an iterative algorithm which exploits the similarity of two subsequent MPC optimization problems; this is achieved by freezing the parameter trajectory to the

one given by the (shifted) parameter trajectory from the previous time step (or iteration), thereby turning the nonlinear optimization problem linear, but time-varying and making it easy to solve.

ii. Conservatism of the stability conditions often encountered in MPC is substantially reduced by considering quasi-LPV models in lieu of a linearization around equilibrium points. Further reduction in conservatism can be achieved by using parameter dependent terminal ingredients.

iii. Quasi-LPV models enable exact representations of nonlinear systems in a simplified manner. Compared to Jacobian linearization, for example, quasi-LPV models are generally more accurate and simpler (as they are linear and not affine).

More specifically, there are several other important aspects that make qLMPC an attractive alternative to other efficient algorithms for implementation of nonlinear MPC laws. An extension to input-output models enables to straightforwardly implement a predictive control law which does not need observers. Stability conditions are adapted to the IO setting and the complexity of the resulting LMI problems is the same as in the state space case. Addressing item 3. above, if no model is available, IO-LPV system identifications techniques are simpler and similarly accurate than their state space counterparts and the IO-qLMPC framework allows to directly utilize the resulting model thus avoiding complications entailed by conversions between IO and SS models [118].

Back to item 2., the novel velocity algorithm for NMPC guarantees stability of the closed-loop without the need to compute terminal ingredients, thereby simplifying design. This is achieved by using terminal equality constraints in the velocity space, which have the advantage (compared to using terminal equality constraints in standard MPC) that no parameterization of equilibrium steady states as a function of the desired set point is necessary (which in the nonlinear case can potentially be a complex task). Furthermore, for the same reason, intermediate way points need not be explicitly computed as all equilibria are mapped to the origin. Under mild assumptions (the system having a continuous locus of equilibria), recursive feasibility of the control law is guaranteed even for short horizons. Short horizons could admittedly have an adverse effect on closed-loop performance since the terminal velocity constraints would to an extent preclude fast trajectories. The velocity algorithm is built upon the use of velocity-based linearization, the resulting model is such that a quasi-LPV representation follows straightforwardly thereby making qLMPC a prime candidate to be used together with the proposed velocity algorithm.

Under somewhat more restrictive conditions (equilibrium of interest being open-loop stable and $u = 0$ being always feasible) stability can be established a priori using the stability analysis tool developed in Chapter 7. In this case, no artificially imposed stabilizing constraints are necessary.

Finally item 3. can be addressed by data-driven techniques in conjunction with predictive control schemes. Indeed, in this thesis a Koopman operator-based qLMPC is proposed in which the dynamics of the system are discovered in real-time and the resulting model is used in a velocity-based qLMPC. In most practical applications a short training experiment would likely be needed to bootstrap the predictive controller (i.e. to give an initial model to work with).

8.2 Outlook

This thesis proposes a framework for nonlinear model predictive control and discusses several aspects regarding modelling, computational complexity, stability and feasibility. Nevertheless there is still much to be further explored within this framework, and several open issues are still to be addressed. The most important one is a thorough discussion of convergence of the qLMPC iterative algorithm. Indeed, a parallelism to SQP methods is established in Appendix A and standard local convergence results from Newton-type methods are discussed, particularly in the velocity case where the system dynamics are linearized, making the resulting equality constraint similar to that encountered in Newton-SQP. However, these conditions are local in nature and are based on the assumption that the initial solution estimate is sufficiently close to the optimal. In the qLMPC case, an important step yet to be taken is to evaluate whether (or under which conditions) the suggested initial parameter trajectory, $\mathsf{P} = \mathbf{1} \otimes H[x_k^\top \quad u_k^\top]$, is appropriate; alternatively, an investigation on how to make the initial-guess parameter trajectory more suitable can be carried out by e.g. optimal feed-forward control trajectories generated offline.

The tracking case is treated in this thesis, however the stability analysis is limited to piece-wise constant reference trajectories. The current methodologies can be extended to consider smooth trajectories but the stability analysis would require a fundamental rework, as the methods presented here assume that the reference is a steady state. One possibility is to consider periodic reference signals along the lines of [76], where an artificial reference is introduced similar to the authors' piece-wise constant reference tracking scenario [45]. The latter was used as a basis for the velocity algorithm proposed in this work, unfortunately the advantages of the velocity algorithm regarding artificial set points cannot be used in the periodic reference case as the target reference is not a steady state and the velocities at the target reference are therefore non-zero.

In the input-output framework, the presented approach deals only with constant terminal ingredients. Extension to parameter dependent ingredients follows straight-forwardly from an analysis stand-point (along the lines of the state space presentation) however the tracking scenario in the IO framework requires gridding over both the parameter and the output spaces and offline computations could quickly become much more complex in the parameter dependent case. On this note, Linear Fractional Representations (LFR) coupled with the Full-Block S-Procedure [103] and D/G scalings can be used to derive parameter independent conditions, making the problem easier to solve.

The dissipativity-based stability analysis result can potentially be overly conservative, as discussed in the numerical example. Conservatism could be partially addressed by considering bounded parameter rates on the parameters, avoiding the parameter *lifting* that is carried out. The number of parameters would then be substantially reduced and the need for D/G scalings could potentially be forgone.

The data-driven Koopman operator-based approach assumes that the dynamics of the system are time-invariant, so that new information is always added to the model but old information is never forgotten. If the system is time-varying, the presented approach would likely not settle and new data would be added perpetually, making the learning algorithm resistant to change (as every new data point would be weighted by a number approaching 0). This stems from the

recursive computation of the approximated Koopman operator (Section 6.2.1) as otherwise a full dictionary of data would need to be stored and memory management as well as computational complexity could be prohibitive. The computation of the Koopman operator could be carried out differently to have a trade-off between complexity and adaptability to time-varying conditions. In addition, the assumption that suitable basis functions are available could be relaxed by making use of the kernel-based Koopman operator [125].

8.2.1 Applications

In addition to the applications reported in this work, there have been several publications by external research groups where the framework and methods presented in this dissertation are used directly or with slight modifications; some examples are listed below. In [7], qLPMC is proposed for autonomous vehicles in a cascaded structure, in which the predictive controller is used for kinematic control and a separate LPV controller for dynamic control. Autonomous driving is a particularly interesting application, given that the usual control structure includes a trajectory planning stage; this stage readily provides a good initial guess for the parameter trajectory, making qLMPC a strong candidate. In [82], predictive control is combined with iterative learning control in a case study of an industrial reactive batch distillation column. The authors examine three different methods to estimate the parameter trajectory along the prediction horizon, including the one used on qLMPC, and come to the conclusion that the qLMPC prediction leads to better performance, when compared to the other two methods. Similarly, [46] also combines qLMPC with learning algorithms, however this approach uses a Guassian process to learn the policy generated by qLMPC, and applying it as an explicit MPC law. In [88], qLMPC is applied in a high-fidelity simulation of a wind turbine to maximize power output and avoid vibrations at the turbine tower natural frequency, thereby minimizing fatigue load. Range maximization of electric vehicles by active cell balancing is proposed in [57]; in this work, qLMPC is used to calculate the balancing currents from cell to cell. The tutorial [86] compares different qLPV-MPC approaches, including qLMPC, and tests them in two simulation studies: a semi-active suspension equipped with an electrorheological damper and a quadruple tank system, in both cases qLMPC displays excellent performance with real-time-capable computational complexity. Similarly, [87] compares several methods on an Active Magnetic Bearing System, where it is seen that performance is comparable to NMPC, but at a lower implementation cost. An extension of qLMPC to economic MPC is presented in [64], showing that qLMPC can also be used to optimize economic costs directly, by adapting the stability analysis accordingly.

Appendix A

Convergence Analysis

This chapter gives an overview of the local convergence characteristics of the qLMPC iterative algorithm. Analysis is based on drawing a parallelism with Newton-type Sequential Quadratic Programming (SQP) in order to apply its well-known convergence results to the qLMPC algorithm. To this end, a summary of the Newton-type SQP method and its convergence conditions is presented in what follows.

The following section is a summary of [18].

A.1 Sequential Quadratic Programming

Consider the nonlinear optimization problem

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & h(x) = 0 \\ & g(x) \leq 0 \end{aligned} \tag{A.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^n \to \mathbb{R}^p$. In the context of nonlinear predictive control (as presented here), $f(x)$ is the quadratic cost function, $h(x)$ represents the nonlinear system dynamics and $g(x)$ reflects the state and input constraints. The idea of SQP is to approximate (A.1) by a quadratic programming *subproblem* at an estimated solution x^l. This is done by using a second order approximation of $f(x)$ at x^l and a first order approximation of $h(x)$ and $g(x)$ at x^l, and using the solution of this subproblem to construct a better estimated solution x^{l+1}. If this general idea sounds familiar is because qLMPC can indeed be categorized as an SQP method.

Before discussing the quadratic subproblems in more detail, important definitions and assumptions are stated.

The following starting assumption is made:

A0: The functions $f(x), h(x), g(x)$ are three times continuously differentiable.

Definition A.1 (Lagrangian function)**.** The Lagrangian function of problem (A.1) is given by

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^\top h(x) + \mu^\top g(x) \tag{A.2}$$

where $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^p_{\geq 0}$ are the so called Lagrange multipliers corresponding to the equality, and inequality constraints respectively.

Definition A.2 (Set of active constraints)**.** The set of active inequality constraints at x is defined as those that are equalities at x, i.e. the index set

$$\mathcal{I}(x) = \{i = 1, ..., p \ : \ g_i(x) = 0\} .$$

A matrix with the gradients of the active constraints $G(x)$ can be constructed as

$$G(x) = \begin{bmatrix} \nabla h_1(x) & ... & \nabla h_m(x) & \nabla g_i(x) \end{bmatrix}, \quad i \in \mathcal{I}(x).$$

This matrix will be used to describe assumptions on the constrained optimization problem in what follows.

In the rest of this chapter, a local *solution* of the optimization problem (A.1) is denoted by x^* and its corresponding Lagrange multipliers by λ^*, μ^*, respectively. The following assumptions are made for each solution:

A1: The first order necessary conditions hold, i.e., there exist optimal multipliers λ^*, $\mu^* \geq 0$ such that

$$\nabla \mathcal{L}(x^*, \lambda^*, \mu^*) = \nabla f(x^*) + \nabla h(x^*)^\top \lambda^* + \nabla g(x^*)^\top \mu^* = 0.$$

A2: The columns of $G(x^*)$ are linearly independent.

A3: Strict complementary slackness holds, i.e.,

$$g_i(x^*)\mu_i^* = 0$$

for $i = 1, ..., p$ and if $g_i(x^*) = 0$ then $\mu_i^* > 0$.

A4: The Hessian of the Lagrangian is positive definite on the null space of $G(x^*)^\top$, i.e.

$$d^\top H\mathcal{L}(x^*, \lambda^*, \mu^*)d > 0$$

$\forall d \neq 0$ such that $G(x^*)^\top d = 0$.

A.1.1 The Quadratic Subproblem

As mentioned above, the philosophy of SQP is to find the solution to the nonlinear optimization problem (A.1) by a sequence of quadratic subproblems obtained from a current solution estimate

x^l. The main reason to consider QP is that they can be efficiently solved but can also to some extent reflect the nonlinearity of the original problem (because of the quadratic cost function). In the context of MPC, SQP is a natural candidate since the objective function $f(x)$ is usually quadratic to begin with. The choice of quadratic subproblem is of utmost importance, the following is the first obvious choice

$$\begin{aligned} \min_{d_x} \quad & \frac{1}{2} d_x^\top B_l d_x + \nabla f(x^l)^\top d_x \\ \text{s.t.} \quad & h(x^l) + \nabla h(x^l) d_x = 0 \\ & g(x^l) + \nabla g(x^l) d_x \leq 0 \end{aligned} \tag{A.3}$$

where $d_x = x - x^l$ and $B_l = Hf(x^l)$ is the Hessian of the original objective function. For general nonlinear optimization problems, where nonlinear constraints are present and the second order approximation of the objective function might not be positive definite, this choice might be inappropriate (see convergence conditions below). Although in MPC, the quadratic cost function can in general be made positive definite by an appropriate choice of Q and R (namely both being positive definite), it is worth reviewing a more generally suitable quadratic subproblem. Consider the quadratic subproblem arising from using a second order approximation of the Lagrangian as the objective function, i.e.

$$\begin{aligned} \min_{d_x} \quad & \frac{1}{2} d_x^\top B_l d_x + \nabla \mathcal{L}(x^l, \lambda^l, \mu^l)^\top d_x \\ \text{s.t.} \quad & h(x^l) + \nabla h(x^l) d_x = 0 \\ & g(x^l) + \nabla g(x^l) d_x \leq 0 \end{aligned} \tag{A.4}$$

where λ^l, μ^l are the Lagrange multiplier iterates associated with the quadratic subproblem (A.4) and $B_l = H\mathcal{L}(x^l, \lambda^l, \mu^l)$. This choice is justified by the fact that conditions A1-A4 imply that x^* is a local minimum of a (A.1) if the objective $f(x)$ is substituted with $\mathcal{L}(x, \lambda^*, \mu^*)$; a good motivation for this quadratic subproblem is that when applied to the so-called KKT system (the system composed of the first order necessary condition A1 and the equality constraints, including the active inequality constraints) it generates iterates identical to those of Newton's method, so that convergence properties from this method are inherited.

The most widely encountered quadratic subproblem is that in (A.3), taking $B_l = H\mathcal{L}(x^l, \lambda^l, \mu^l)$; note that in contrast to (A.4), the gradient of the objective is given by the gradient of the original $f(x)$; this is motivated by the fact that if only equality constraints are present, the linearized equality constraint forces $\nabla h(x^l) d_x$ to be a constant, so that the objective function from (A.4) reduces to the one in (A.3) if $B_l = H\mathcal{L}(x^l, \lambda^l, \mu^l)$. In the inequality constrained case, these problems are only equivalent if the multiplier estimate μ^l is zero for all inactive linearized constraints. However, this quadratic subproblem is indeed equivalent to (A.4) for the slack variable formulation of (A.1) (see [18] for more details). In conclusion, the quadratic subproblem (A.3) with $B_l = H\mathcal{L}(x^l, \lambda^l, \mu^l)$ is an appropriate quadratic subproblem for (A.1).

Basic Newton SQP Algorithm

Algorithm 1 Sequential Quadratic Programming

Initialization: x^0, λ^0, μ^0

1: **repeat**
2: $l \leftarrow 0$
3: Compute B^l
4: Solve (A.3) to obtain (d_x, λ, μ)
5: Set $x^{l+1} = x^l + d_x$
6: **until** convergence

A.1.2 Local Convergence

This section establishes conditions for which the solution estimate iterates converge to a solution given that the starting data $x^0, \lambda^0, \mu^0, B_0$ are sufficiently close to the corresponding data at a solution x^*. An important assumption is that active inequality constraints at x^* are known. This is justified because if x^l is close to x^* then the QP problem (A.3) will have the same active constraints at x^l as the nonlinear problem (A.1) at x^*. The motivation for this is that those constraints which are inactive for (A.1) at x^* can be ignored for the QP subproblem, and those that are active can be treated as equality constraints. Hence for the purpose of local convergence analysis, the much simpler case of equality-constrained QP subproblems can be considered without loss of generality.

For reference, the equality-constrained subproblem is rewritten so that no inequality constraints are explicitly included

$$\begin{aligned} \min_{d_x} \quad & \frac{1}{2} d_x^\top B_l d_x + \nabla f(x^l)^\top d_x \\ \text{s.t.} \quad & h(x^l) + \nabla h(x^l) d_x = 0 \end{aligned} \tag{A.5}$$

Local convergence follows from the use of Newton's method on the system of equations obtained from the first order necessary conditions A1, and the constraint equation:

$$\Psi(x, \lambda) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda) \\ h(x) \end{bmatrix} = 0$$

It follows from assumptions A1-A4 that, at the solution (x^*, λ^*) the Jacobian of this system of equations,

$$J(x^*, \lambda^*) = \begin{bmatrix} H\mathcal{L}(x^*, \lambda^*) & \nabla h(x^*) \\ \nabla h(x^*)^\top & 0 \end{bmatrix},$$

is nonsingular. So Newton iterates given by the solution to

$$J(x^l, \lambda^l) \begin{bmatrix} d_x \\ d_\lambda \end{bmatrix} = -\Psi(x^l, \lambda^l)$$

converge to (x^*, λ^*) if (x^0, λ^0) is sufficiently close.

Conditions for Local Convergence

In what follows, the properties of the approximations B_l sufficient to guarantee local (Newton-like) convergence of SQP are discussed. The following assumption is made

A5: The matrix $H\mathcal{L}(x^*, \lambda^*, \mu^*)$ is nonsingular.

The following conditions guarantee local convergence of the Newton SQP algorithm.

C1: The matrices B_l are uniformly positive definite on the null spaces of $\nabla h(x_l)^\top$, such that $\forall l$

$$d^\top B_l d \geq \beta_1 \|d\|^2 \quad \forall d \ : \ \nabla h(x^l)^\top d = 0.$$

C2: The sequence $\{B_l\}$ is uniformly bounded, i.e. $\exists \beta_2 > 0$ such that $\forall l$

$$\|B_l\| \leq \beta_2$$

C3: The matrices $\{B_l\}$ have uniformly bounded inverses, i.e. $\exists \beta_3 > 0$ such that $\forall l$ B_l exists and

$$\|B_l^{-1}\| \leq \beta_3$$

A.1.3 Newton SQP for Nonlinear MPC

This section gives a rather brief overview of how a typical nonlinear MPC problem can be solved using SQP, for a more in-depth review see [44]. The purpose is to later show that the qLMPC algorithm under some conditions resembles this formulation and as such convergence analysis can potentially be carried out along the lines of SQP.

Consider the nonlinear system governed by the discrete-time model

$$x_{k+1} = f(x_k, u_k)$$

and a typical MPC optimization problem

$$\min_Z \frac{1}{2} Z^\top Q Z$$
$$\text{subject to}$$
$$h(Z) = 0$$

where

$$Q = \text{diag}(Q, R, Q, R, Q..., R, Q),$$
$$Z = [x_0^\top \ u_0^\top \ x_1^\top \ u_1^\top \ x_2^\top \ \dots \ u_{N-1}^\top \ x_N^\top]^\top,$$
$$h(Z) = [(x_0 - x_k)^\top \ (f(x_0, u_0) - x_1)^\top \ (f(x_1, u_1) - x_2)^\top \ \dots \ (f(x_{N-1}, u_{N-1}) - x_N)^\top]^\top.$$

In MPC, inequality constraints are indeed practically relevant, unfortunately convergence analysis in this case is much more cumbersome. Similar to the derivations above, one can assume

only equality constrained problems if the assumption is made that active inequality constraints at x^* are known, so that these are treated as equalities, while the rest are ignored.

A QP subproblem can be derived in the form of (A.3) with $B_l = H\mathcal{L}(\cdot)$ as

$$\begin{aligned} &\min_{d_Z} \frac{1}{2} d_Z^\top (Q + Hh(x^l)) d_Z \\ &\text{subject to} \\ &\quad h(Z^l) + \nabla h(Z^l) d_Z = 0 \end{aligned} \tag{A.6}$$

The equality constraints can also be written as

$$\begin{bmatrix} I & & & & & & & \\ \nabla_x f|_0 & \nabla_u f|_0 & -I & & & & & \\ & & \nabla_x f|_1 & \nabla_u f|_1 & -I & & & \\ & & & & \nabla_x f|_2 & \nabla_u f|_2 & -I & \\ & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} d_{x0} \\ d_{u0} \\ d_{x1} \\ d_{u1} \\ d_{x2} \\ d_{u2} \\ d_{u3} \\ \vdots \end{bmatrix} = \begin{bmatrix} x_k - x_0^l \\ f(x_0^l, u_0^l) - x_1^l \\ f(x_1^l, u_1^l) - x_2^l \\ f(x_2^l, u_2^l) - x_3^l \\ \vdots \end{bmatrix} \tag{A.7}$$

where $\nabla_x f|_i = \nabla_x f(x_i^l, u_i^l)$, and similarly for $\nabla_u f|_i$.

A.2 qLMPC

Having reviewed the derivation and convergence properties of Newton-type SQP, a parallelism to qLMPC under certain conditions can be established. Assume, as before, a nonlinear system governed by the difference equation

$$x_{k+1} = f(x_k, u_k).$$

A.2.1 Incremental model

The velocity-based qLMPC framework from Chapter 4 is to be used; to this end, a quasi-LPV model is derived by applying the Mutlivariable Mean Value Theorem (see Section 4.2.1) yielding the state-dependent (quasi-LPV) model

$$\Delta x_{k+1} = \nabla_x f(x_k, u_k) \Delta x_k + \nabla_u f(x_k, u_k) \Delta u_k \tag{A.8}$$

The MPC optimization problem can thus be written as

$$\begin{aligned} &\min_{Z} \frac{1}{2} Z^\top Q Z \\ &\text{subject to} \\ &\quad x_0 = x_k \\ &\quad \tilde{h}(Z^l) Z = 0 \end{aligned} \tag{A.9}$$

where

$$Q = \text{diag}(Q, R, Q, R, ..., R, Q),$$
$$Z = [\Delta x_0^\top \ \Delta u_0^\top \ \Delta x_1^\top \ \Delta u_1^\top \ \Delta x_2^\top \ ... \ \Delta u_{N-1}^\top \ \Delta x_N^\top]^\top.$$

As before, the equality constraint can be expanded to yield

$$\begin{bmatrix} I & & & & & & & \\ \nabla_x f|_0 & \nabla_u f|_0 & -I & & & & & \\ & & \nabla_x f|_1 & \nabla_u f|_1 & -I & & & \\ & & & & \nabla_x f|_2 & \nabla_u f|_2 & -I & \\ & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \Delta x_0 \\ \Delta u_0 \\ \Delta x_1 \\ \Delta u_1 \\ \Delta x_2 \\ \Delta u_2 \\ \Delta u_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} x_k - x_{k-1} \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{A.10}$$

revealing that is it almost identical to that of the SQP in (A.7); in particular, the constraints span the same space.

A.2.2 Newton's Method on qLMPC

So far an equivalence of the constraints present in Newton-SQP and qLMPC has been established; however, an important difference remains, namely that in the QP subproblems of Newton SQP, the Hessian of the Lagrangian is used, as opposed to the Hessian of the objective function from the original nonlinear optimization problem. This stems from the fact that, as reviewed in Section A.1.2, in order to express the nonlinear system of equations obtained from the first order necessary conditions in a linear fashion (to be solved efficiently), the *derivative* (Jacobian) of the first order necessary conditions is used. In the qLMPC case, however, a slightly modified Newton-like method can be applied in which it is not necessary to consider the derivative of the first order necessary conditions, but rather, as in the case of linear constraints, the first order necessary conditions themselves.

Consider the system of equations arising from the first order necessary condition for the optimization problem (A.9) and the constraint equations

$$\Psi(x, \lambda) = \begin{bmatrix} \nabla \mathcal{L}(x, \lambda) \\ \tilde{h}(Z^l) Z \end{bmatrix} = 0$$

given the (quasi-) linearity of the equality constraints, this can be easily expressed as the KKT system

$$\begin{bmatrix} \mathcal{Q} & \tilde{h}(Z^l) \\ \tilde{h}(Z^l) & 0 \end{bmatrix} \begin{bmatrix} Z \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

so that it can be solved without requiring a computation of the Hessian of the Lagrangian.

Finally, regarding convergence, it is easy to see that condition C1 will hold for qLMPC if it holds for Newton SQP, since the nullspace $\nabla h(x^l)^\top$ is identical, and the Hessian of A.9 can be made positive definite by choosing positive definite Q, R. Conditions C2, C3 are also trivially fulfilled for this choice of Q, R.

A.2.3 Augmented model

In the previous section an incremental qLMPC model derived from velocity-based linearization was considered in order to immediately establish an analogy to Newton SQP. However, the *purely* incremental model is not practical as it would not be able to track or regulate around a desired set point, but rather only stabilize the system. In this section, the discussion is extended to the augmented model as presented in Chapter 4. Consider the augmented state-dependent (quasi-LPV) system obtained from applying velocity-based linearization to the nonlinear model and subsequently augmenting the model.

$$\begin{bmatrix} x_{k+1} \\ \Delta x_{k+1} \end{bmatrix} = \begin{bmatrix} I & \nabla_x f(x_k, u_k) \\ 0 & \nabla_x f(x_k, u_k) \end{bmatrix} \begin{bmatrix} x_k \\ \Delta x_k \end{bmatrix} + \begin{bmatrix} \nabla_u f(x_k, u_k) \\ \nabla_u f(x_k, u_k) \end{bmatrix} \Delta u_k \tag{A.11}$$

where the first block-row equation is equivalent to $x_{k+1} = x_k + \Delta x_{k+1}$ and is analogous to the Newton step $x^{l+1} = x^l + d_x$ (line 5: in Algorithm 1). The optimization problem is given by (A.9) where

$$\mathcal{Q} = \text{diag}(Q_1, Q_2, R, Q_1, Q_2, R, ..., R, Q_1, Q_2),$$

$$Z = [x_0^\top \ \Delta x_0^\top \ \Delta u_0^\top \ x_1^\top \ \Delta x_1^\top \ \Delta u_1^\top \ ... \ \Delta u_{N-1}^\top \ x_N^\top \ \Delta x_N^\top]^\top,$$

and the equality constraint is now given by

$$\begin{bmatrix} I & & & & & & & & & \\ I & \nabla_x f & \nabla_u f & -I & & & & & & \\ & \nabla_x f & \nabla_u f & 0 & -I & & & & & \\ & & & I & \nabla_x f & \nabla_u f & -I & & & \\ & & & 0 & \nabla_x f & \nabla_u f & 0 & -I & & \\ & & & & & & & & \ddots & \end{bmatrix} \begin{bmatrix} x_0 \\ \Delta x_0 \\ \Delta u_0 \\ x_1 \\ \Delta x_1 \\ \Delta u_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} x_k \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}. \tag{A.12}$$

In order to establish an analogy to the Newton-SQP along the lines as before, the constraint can be brought to the same form of (A.7). One way to do this is by substituting the additional constraints into the cost function. To this end, one can perform simple algebraic manipulation

to express the constraints involving the states x_i in terms of the incremental states Δx_i. This follows intuitively by using the identity

$$x_i = x_0 + \sum_{j=1}^{i} \Delta x_j \tag{A.13}$$

(this expression can also be derived from the odd row-block equations in A.12). The resulting equality constraints have indeed the same structure as (A.7), (A.10), but the cost function was modified in the process, and as such convergence cannot be immediately established before analyzing the resulting objective function. Let us rewrite the cost function in a way which lends itself to a simple substitution of the equality constraints mentioned above:

$$\frac{1}{2}Z^\top QZ = \frac{1}{2}\left([*]^\top \mathrm{diag}(Q_1Q_2, R, Q_2, R...) \begin{bmatrix} x_0 \\ \Delta x_0 \\ \Delta u_0 \\ \Delta x_1 \\ \Delta u_1 \\ \vdots \end{bmatrix} + [*]^\top \mathrm{diag}_{N-1}(Q_1) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} \right)$$

the second term on the right hand side can then be rewritten using the identity (A.13) as

$$[*]^\top \mathrm{diag}_N(Q_1) \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix} = [*]^\top \underbrace{\begin{bmatrix} NQ_1 & NQ_1 & (N-1)Q_1 & (N-2)Q_1 & \dots \\ NQ_1 & NQ_1 & (N-1)Q_1 & (N-2)Q_1 & \dots \\ (N-1)Q_1 & (N-1)Q_1 & (N-1)Q_1 & (N-2)Q_1 & \dots \\ (N-2)Q_1 & (N-2)Q_1 & (N-2)Q_1 & (N-2)Q_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}_{\tilde{Q}} \begin{bmatrix} x_0 \\ \Delta x_1 \\ \Delta x_2 \\ \vdots \end{bmatrix}.$$

where, provided Q is positive definite, $\tilde{Q}$ is positive definite. So that the Hessian of the objective function, given by

$$\mathrm{diag}(Q_1, Q_2, R, Q_2, R, ...) + \tilde{Q}$$

still fulfills the conditions C1-C3 and therefore local convergence can be established.

Bibliography

[1] H. S. Abbas, R. Tóth, N. Meskin, J. Mohammadpour, and J. Hanema. "A Robust MPC for Input-Output LPV Models". In: *IEEE Transactions on Automatic Control* 61.12 (2016), pp. 4183–4188. DOI: 10.1109/TAC.2016.2553143.

[2] H. S. Abbas, R. Tóth, N. Meskin, J. Mohammadpour, and J. Hanema. "An MPC approach for LPV systems in input-output form". In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015, pp. 91–96. DOI: 10.1109/CDC.2015.7402091.

[4] HS Abbas, J Hanema, R Tóth, J Mohammadpour, and N Meskin. "An improved robust model predictive control for linear parameter-varying input-output models". In: *International Journal of Robust and Nonlinear Control* 28.3 (2018), pp. 859–880.

[5] Ian Abraham, Gerardo De La Torre, and Todd D Murphey. "Model-based control using Koopman operators". In: *arXiv preprint arXiv:1709.01568* (2017).

[6] Bernt M. Åkesson, Hannu T. Toivonen, Jonas B. Waller, and Rasmus H. Nyström. "Neural network approximation of a nonlinear model predictive controller applied to a pH neutralization process". In: *Computers and Chemical Engineering* 29.2 (2005), pp. 323–335. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2004.09.023. URL: http://www.sciencedirect.com/science/article/pii/S009813540400287X.

[7] Eugenio Alcalá, Vicenç Puig, and Joseba Quevedo. "LPV-MPC Control for Autonomous Vehicles". In: *IFAC-PapersOnLine* 52.28 (2019). 3rd IFAC Workshop on Linear Parameter Varying Systems LPVS 2019, pp. 106–113. ISSN: 2405-8963.

[8] Mukhtar Ali, Hossam S Abbas, and Herbert Werner. "MIMO controller synthesis for LTI and LPV systems using input-output models". In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 11338–11343.

[9] Mukhtar Ali, Hossam Abbas, and Herbert Werner. "Controller synthesis for input-output LPV models". In: *49th IEEE Conference on Decision and Control (CDC)*. IEEE. 2010, pp. 7694–7699.

[10] David Angeli, Alessandro Casavola, and Edoardo Mosca. "Constrained predictive control of nonlinear plants via polytopic linear system embedding". In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 10.13 (2000), pp. 1091–1103.

[11] P. Apkarian and P. Gahinet. "A convex characterization of gain-scheduled H/sub /spl infin// controllers". In: *IEEE Transactions on Automatic Control* 40.5 (1995), pp. 853–864. DOI: 10.1109/9.384219.

[12] C. Ariño, E. Pérez, A. Querol, and A. Sala. "Model Predictive Control for discrete fuzzy systems via iterative quadratic programming". In: *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. July 2014, pp. 2288–2293. DOI: 10.1109/FUZZ-IEEE.2014.6891633.

[13] V. Bachtiar, T. Mühlpfordt, W.H. Moase, T. Faulwasser, R. Findeisen, and C. Manzie. "Nonlinear model predictive missile control with a stabilising terminal constraint". In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 457–462. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.02122. URL: http://www.sciencedirect.com/science/article/pii/S1474667016416563.

[14] Bassam Bamieh and Laura Giarré. "Identification of linear parameter varying models". In: *International Journal of Robust and Nonlinear Control* 12.9 (July 2002), pp. 841–853. ISSN: 1049-8923. DOI: 10.1002/rnc.706. URL: http://dx.doi.org/10.1002/rnc.706.

[15] A. Bemporad and M. Morari. "Robust Model Predictive Control: A Survey". In: *Lecture Notes in Control and Information Sciences* 245 (1999), pp. 207–226.

[16] G. Betti, M. Farina, and R. Scattolini. "An MPC algorithm for offset-free tracking of constant reference signals". In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. 2012, pp. 5182–5187. ISBN: 0191-2216. DOI: 10.1109/CDC.2012.6426758.

[17] Hans Georg Bock and Karl-Josef Plitt. "A multiple shooting algorithm for direct solution of optimal control problems". In: *IFAC Proceedings Volumes* 17.2 (1984), pp. 1603–1608.

[18] Paul T Boggs and Jon W Tolle. "Sequential quadratic programming". In: *Acta numerica* 4 (1995), pp. 1–51.

[19] Sudchai Boonto. "Identification of Linear Parameter-Varying Input-Output Models". PhD thesis. Hamburg University of Technology, 2011.

[20] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Vol. 15. Studies in Applied Mathematics. Philadelphia, PA, June 1994. ISBN: 0-89871-334-X.

[21] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. USA: Cambridge University Press, 2004. ISBN: 0521833787.

[22] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. "Sparse identification of nonlinear dynamics with control (SINDYc)". In: *IFAC-PapersOnLine* 49.18 (2016), pp. 710–715.

[23] Marko Budišić, Ryan Mohr, and Igor Mezić. "Applied Koopmanism". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22.4 (2012), p. 047510.

[24] Horacio M Calderón, Pablo SG Cisneros, and Herbert Werner. "qLPV Predictive Control-A Benchmark Study on State Space vs Input-Output Approach". In: *IFAC-PapersOnLine* 52.28 (2019), pp. 146–151.

[25] Mark Cannon, Johannes Buerger, Basil Kouvaritakis, and Saša Rakovic. "Robust tubes in nonlinear model predictive control". In: *IEEE Transactions on Automatic Control* 56.8 (2011), pp. 1942–1947.

[26] Alessandro Casavola, Domenico Famularo, and Giuseppe Franze. “A feedback min-max MPC algorithm for LPV systems subject to bounded rates of change of parameters”. In: *IEEE Transactions on Automatic Control* 47.7 (2002), pp. 1147–1153.

[27] Alessandro Casavola, Domenico Famularo, Giuseppe Franzè, and Emanuele Garone. “A fast ellipsoidal MPC scheme for discrete-time polytopic linear parameter varying systems”. In: *Automatica* 48.10 (2012), pp. 2620–2626.

[28] L. Chisci, P. Falugi, and G. Zappa. “Predictive tracking control of constrained nonlinear systems”. In: *Proc.-Control Theory and Applications* 152.3 (May 2005).

[29] Luigi Chisci, Paola Falugi, and Giovanni Zappa. “Gain-scheduling MPC of nonlinear systems”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 13.3-4 (2003), pp. 295–308.

[33] Pablo S. G. Cisneros, Sophia Voss, and Herbert Werner. “Efficient Nonlinear Model Predictive Control via quasi-LPV Representation”. In: *55th IEEE Conference in Decision and Control* (Dec. 2016).

[34] Pablo S. G. Cisneros and Herbert Werner. “A Dissipativity Formulation for Stability Analysis of Nonlinear and Parameter Dependent MPC”. In: *American Control Conference*. IEEE. June 2018.

[35] Pablo S. G. Cisneros and Herbert Werner. “A Velocity Algorithm for Nonlinear Model Predictive Control”. In: *IEEE Transactions on Control Systems Technology* (2020).

[37] Pablo S. G. Cisneros and Herbert Werner. “Parameter Dependent Stability Conditions for Quasi-LPV Model Predictive Control”. In: *American Control Conference* (May 2017).

[38] Pablo S. G. Cisneros and Herbert Werner. “Stabilizing Model Predictive Control for Nonlinear Systems in Input-Output quasi-LPV Form”. In: *American Control Conference*. IEEE. July 2019.

[40] D. W. Clarke, C. Mohtadi, and P. S. Tuffs. “Generalized Predictive Control-Part I. The Basic Algorithm”. In: *Automatica* 23.2 (1987), pp. 137–148.

[42] C.R. Cutler and B.C. Ramaker. “Dynamic Matrix Control - A computer Control Algorithm”. In: *Automatic Control Conference, San Francisco, CA* (1980).

[43] Jonathan A. DeCastro. “Rate-Based Model Predictive Control of Turbofan Engine Clearance”. In: *Journal of Propulsion and Power* 23.4 (July 2007), pp. 804–813. ISSN: 1533-3876. DOI: 10.2514/1.25846. URL: http://dx.doi.org/10.2514/1.25846.

[44] Moritz Diehl, Hans Georg Bock, and Johannes P Schlöder. “A real-time iteration scheme for nonlinear optimization in optimal feedback control”. In: *SIAM Journal on control and optimization* 43.5 (2005), pp. 1714–1736.

[45] A. Ferramosca, D. Limon, I. Alvarado, T. Alamo, and E.F. Camacho. “MPC for tracking of constrained nonlinear systems”. In: *48th IEEE Conference on Decision and Control* (Dec. 2009).

[46] D. Gángó, T. Péni, and R. Tóth. “Learning Based Approximate Model Predictive Control for Nonlinear Systems”. In: *IFAC-PapersOnLine* 52.28 (2019). 3rd IFAC Workshop on Linear Parameter Varying Systems LPVS 2019, pp. 152–157.

[47] Wilfried Gilbert, Didier Henrion, Jacques Bernussou, and David Boyer. “Polynomial LPV synthesis applied to turbofan engines”. In: *Control Engineering Practice* 18.9 (2010), pp. 1077–1083.

[48] Pablo S González Cisneros and Herbert Werner. “Nonlinear model predictive control for models in quasi-linear parameter varying form”. In: *International Journal of Robust and Nonlinear Control* (2020).

[49] Ramón Gonzalez, Mirko Fiacchini, Teodoro Alamo, José Luis Guzmán, and Francisco Rodrıguez. “Online robust tube-based MPC for time-varying systems: A practical approach”. In: *International Journal of Control* 84.6 (2011), pp. 1157–1170.

[50] Jurre Hanema, Mircea Lazar, and Roland Tóth. “Stabilizing tube-based model predictive control: Terminal set and cost construction for LPV systems”. In: *Automatica* 85 (2017), pp. 137–144.

[51] Jurre Hanema, Roland Tóth, and Mircea Lazar. “Stabilizing non-linear MPC using linear parameter-varying representations”. In: *2017 IEEE 56th Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 3582–3587.

[52] Jurre Hanema, Roland Tóth, and Mircea Lazar. “Tube-based anticipative model predictive control for linear parameter-varying systems”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 1458–1463.

[53] Jurre Hanema, Roland Toth, Mircea Lazar, and Hossam S Abbas. “MPC for linear parameter-varying systems in input-output representation”. In: *2016 IEEE International Symposium on Intelligent Control (ISIC)*. IEEE. 2016, pp. 1–6.

[54] Seyed Mahdi Hashemi, Hossam Seddik Abbas, and Herbert Werner. “Low-complexity linear parameter-varying modeling and control of a robotic manipulator”. In: *Control Engineering Practice* 20.3 (2012), pp. 248–257.

[55] WP Heath, G Li, AG Wills, and B Lennox. “The robustness of input constrained model predictive control to infinity-norm bound model uncertainty”. In: *IFAC Proceedings Volumes* 39.9 (2006), pp. 495–500.

[56] WP Heath, Adrian G Wills, and JAG Akkermans. “A sufficient condition for the stability of optimizing controllers with saturating actuators”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 15.12 (2005), pp. 515–529.

[57] F. S. J. Hoekstra, H. J. Bergveld, and M. C. F. Donkers. “Range Maximisation of Electric Vehicles through Active Cell Balancing using Reachability Analysis”. In: *American Control Conference*. 2019, pp. 1567–1572.

[58] Christian Hoffmann and Herbert Werner. “LFT-LPV modeling and control of a control moment gyroscope”. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE. 2015, pp. 5328–5333.

[59] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. “ACADO toolkit—An open-source framework for automatic control and dynamic optimization”. In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312.

[60] Rolf Isermann. *Digital Control Systems*. Springer Berlin Heidelberg, 1981. ISBN: 9783662023198. DOI: 10.1007/978-3-662-02319-8. URL: http://dx.doi.org/10.1007/978-3-662-02319-8.

[61] Ulf Jönsson. "Lecture notes on integral quadratic constraints". In: (2001).

[62] Isaac Kaminer, Antonio M. Pascoal, Pramod P. Khargonekar, and Edward E. Coleman. "A velocity algorithm for the implementation of gain-scheduled controllers". In: *Automatica* 31.8 (1995), pp. 1185–1191. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(95)00026-S. URL: http://www.sciencedirect.com/science/article/pii/000510989500026S.

[63] Bartosz Käpernick and Knut Graichen. "The gradient based nonlinear model predictive control software GRAMPC". In: *2014 European Control Conference (ECC)*. IEEE. 2014, pp. 1170–1175.

[64] Fatemeh Karimi-Pour, Vicenç Puig, and Carlos Ocampo-Martinez. "Economic model predictive control of nonlinear systems using a linear parameter varying approach". In: *International Journal of Robust and Nonlinear Control* (2021).

[65] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM. 1984, pp. 302–311.

[66] S. S. Keerthi and E. G. Gilbert. "Optimal, infinite horizon feedback laws for a general class of constrained discrete time systems: Stability and moving-horizon approximations." In: *Journal of Optimization Theory and Application* 57 (1988), pp. 265–293.

[67] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*. Vol. 3. Prentice hall Upper Saddle River, NJ, 2002.

[68] T. H. Kim, J. H. Park, and T. Sugie. "Output-feedback Model Predictive Control for LPV Systems with Input Saturation based on Quasi-Min-Max Algorithm". In: *IEEE Conference on Decision and Control* 45 (Dec. 2006), pp. 1454–1459.

[69] Tae-Hyoung Kim, Jee-Hun Park, and Toshiharu Sugie. "Output-feedback model predictive control for LPV systems with input saturation based on quasi-min-max algorithm". In: *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE. 2006, pp. 1454–1459.

[71] Mayuresh V Kothare, Venkataramanan Balakrishnan, and Manfred Morari. "Robust constrained model predictive control using linear matrix inequalities". In: *Automatica* 32.10 (1996), pp. 1361–1379.

[72] Young Il Lee, Basil Kouvaritakis, and Mark Cannon. "Constrained receding horizon predictive control for nonlinear systems". In: *Automatica* 38.12 (2002), pp. 2093–2102.

[73] D. J. Leith and W. E. Leithead. "Gain-Scheduled and nonlinear systems: dynamic analysis by velocity-based linearization families". In: *International Journal of Control* 70.2 (1998), pp. 289–317.

[74] Wei C Li and Lorenz T Biegler. "A multistep, Newton-type control strategy for constrained, nonlinear processes". In: *1989 American Control Conference*. IEEE. 1989, pp. 1526–1527.

[75] D. Limon, I. Alvarado, T. Alamo, and E.F. Camacho. "MPC for tracking piecewise constant references for constrained linear systems". In: *Automatica* 44.9 (2008), pp. 2382–2387. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2008.01.023. URL: http://www.sciencedirect.com/science/article/pii/S0005109808001106.

[76] Daniel Limon, Teodoro Alamo, D Muñoz de la Peña, Melanie Nicole Zeilinger, CN Jones, and Mario Pereira. "MPC for tracking periodic reference signals". In: *IFAC Proceedings Volumes* 45.17 (2012), pp. 490–495.

[77] J. Löfberg. "Linear Model Predictive Control: Stability and Robustness". PhD thesis. Linköping Studies in Science and Technology, Linköping University, 2001.

[78] Johan Löfberg. *YALMIP*. June 2019. URL: https://yalmip.github.io/command/geomean/.

[79] Y. Lu and Y. Arkun. "A Quasi-min-max MPC Algorithm for Linear Parameter-Varying Systems with Bounded Rate of Change of Parameters". In: *American Control Conference* (June 2000), pp. 3234–3238.

[80] Yaohui Lu and Yaman Arkun. "A scheduling quasi–min-max model predictive control algorithm for nonlinear systems". In: *Journal of Process Control* 12.5 (2002), pp. 589–604.

[81] Yaohui Lu and Yaman Arkun. "Quasi-min-max MPC algorithms for LPV systems". In: *Automatica* 36.4 (2000), pp. 527–540.

[82] Alejandro Marquez-Ruiz, Marco Loonen, M. Bahadır Saltık, and Leyla Özkan. "Model Learning Predictive Control for Batch Processes: A Reactive Batch Distillation Column Case Study". In: *Industrial Engineering Chemistry Research* 58.30 (2019).

[83] David Q Mayne and Hannah Michalska. "Receding horizon control of nonlinear systems". In: *IEEE Transactions on automatic control* 35.7 (1990), pp. 814–824.

[84] Alexandre Megretski and Anders Rantzer. "System analysis via integral quadratic constraints". In: *IEEE Transactions on Automatic Control* 42.6 (1997), pp. 819–830.

[85] H. Michalska and D. Q. Mayne. "Robust receding horizon control of constrained nonlinear systems." In: *IEEE Transactions on Automatic Control* (1993), pp. 1623–1632.

[86] Marcelo Menezes Morato, Gia Quoc Bao Tran, Guilherme dos Reis, Julio Normey-Rico, and Olivier Sename. "NMPC Through qLPV Embedding: A Tutorial Review of Different Approaches". In: *7th IFAC Conference on Nonlinear Model Predictive Control*. 2021.

[87] Abdelrahman Morsi, Hossam Seddik Abbas, Sabah Mohamed Ahmed, and Abdelfatah Mahmoud Mohamed. "Model Predictive Control Based on Linear Parameter-Varying Models of Active Magnetic Bearing Systems". In: *IEEE Access* 9 (2021), pp. 23633–23647.

[88] Sebastiaan Paul Mulders, Tobias Gybel Hovgaard, Jacob Deleuran Grunnet, and Jan-Willem van Wingerden. "Preventing wind turbine tower natural frequency excitation with a quasi-LPV model predictive control scheme". In: *Wind Energy* 23.3 (2020), pp. 627–644.

[89] Matthias A Müller, Lars Grüne, and Frank Allgöwer. "On the role of dissipativity in economic model predictive control". In: *IFAC-PapersOnLine* 48.23 (2015), pp. 110–116.

[90] Toshiyuki Ohtsuka. "A continuation/GMRES method for fast computation of nonlinear receding horizon control". In: *Automatica* 40.4 (2004), pp. 563–574.

[91] Mauricio C. de Oliveira and Robert E. Skelton. "Stability tests for constrained linear systems". In: *Perspectives in robust control*. Ed. by S. O. Reza Moheimani. London: Springer London, 2001, pp. 241–257. ISBN: 978-1-84628-576-9.

[92] Gabriele Pannocchia, Marco Gabiccini, and Alessio Artoni. "Offset-free MPC explained: novelties, subtleties, and applications". In: *IFAC-PapersOnLine* 48.23 (2015). 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015, pp. 342–351. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.11.304. URL: http://www.sciencedirect.com/science/article/pii/S2405896315025884.

[93] Gabriele Pannocchia, James B. Rawlings, and Stephen J. Wright. "Conditions under which suboptimal nonlinear MPC is inherently robust". In: *Systems Control Letters* 60.9 (2011), pp. 747–755. ISSN: 0167-6911. DOI: https://doi.org/10.1016/j.sysconle.2011.05.013.

[94] Bert Pluymers, JA Rossiter, JAK Suykens, and Bart De Moor. "Interpolation based MPC for LPV systems using polyhedral invariant sets". In: *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, pp. 810–815.

[95] Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. "Generalizing Koopman theory to allow for inputs and control". In: *SIAM Journal on Applied Dynamical Systems* 17.1 (2018), pp. 909–930.

[96] Tobias Raff, Christian Ebenbauer, and Prank Allgöwer. "Nonlinear model predictive control: A passivity-based approach". In: *Assessment and future directions of nonlinear model predictive control*. Springer, 2007, pp. 151–162.

[97] J. B. Rawlings, D. Bonne, J. B. Jorgensen, A. N. Venkat, and S. B. Jorgensen. "Unreachable Setpoints in Model Predictive Control". In: *IEEE Transactions on Automatic Control* 53.9 (2008), pp. 2209–2215. DOI: 10.1109/TAC.2008.928125.

[98] J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

[99] James B Rawlings, David Angeli, and Cuyler N Bates. "Fundamentals of economic model predictive control". In: *2012 IEEE 51st IEEE conference on decision and control (CDC)*. IEEE. 2012, pp. 3851–3861.

[100] J. Richalet, A. Rault, J.L. Testud, and J. Papon. "Model Predictive Heuristic Control: Application to Industrial Processes". In: *Automatica* 14.5 (1978), pp. 413–428.

[101] J. A. Rossiter. *Model-Based Predictive Control: A Practical Approach*. CRC Press, 2003.

[102] Wilson J. Rugh and Jeff S. Shamma. "Research on gain scheduling". In: *Automatica* 36.10 (2000), pp. 1401–1425. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(00)00058-3. URL: http://www.sciencedirect.com/science/article/pii/S0005109800000583.

[103] Carsten W Scherer. "LPV control and full block multipliers". In: *Automatica* 37.3 (2001), pp. 361–375.

[104] Peter J Schmid. "Dynamic mode decomposition of numerical and experimental data". In: *Journal of fluid mechanics* 656 (2010), pp. 5–28.

[105] Erik Schulz, Ashish Bussa, and Herbert Werner. "Identification of linear parameter-varying systems via IO and subspace identification-a comparison". In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 7147–7152.

[106] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. "Suboptimal model predictive control (feasibility implies stability)". In: *IEEE Transactions on Automatic Control* 44.3 (1999), pp. 648–654. DOI: 10.1109/9.751369.

[107] Gerard Scorletti and Laurent El Ghaoui. "Improved LMI conditions for gain scheduling and related control problems". In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 8.10 (1998), pp. 845–877.

[108] Peter Seiler, Andrew Packard, and Gary J Balas. "A dissipation inequality formulation for stability analysis with integral quadratic constraints". In: *49th IEEE Conference on Decision and Control (CDC)*. IEEE. 2010, pp. 2304–2309.

[109] Jeff S Shamma. "Analysis and design of gain scheduled control systems". PhD thesis. Massachusetts Institute of Technology, 1988.

[110] Jeff S. Shamma and Michael Athans. "Guaranteed properties of gain scheduled control for linear parameter-varying plants". In: *Automatica* 27.3 (1991), pp. 559–564. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(91)90116-J. URL: http://www.sciencedirect.com/science/article/pii/000510989190116J.

[111] D. Simon, J. Löfberg, and T. Glad. "Reference Tracking MPC Using Dynamic Terminal Set Transformation". In: *IEEE Transactions on Automatic Control* 59.10 (2014), pp. 2790–2795. DOI: 10.1109/TAC.2014.2313767.

[112] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[113] Mark W Spong and Daniel J Block. "The pendubot: A mechatronic system for control research and education". In: *Proceedings of 1995 34th IEEE Conference on Decision and Control*. Vol. 1. IEEE. 1995, pp. 555–556.

[114] Yang Su, Kok Kiong Tan, and Tong Heng Lee. "Tube based quasi-min-max output feedback MPC for LPV systems". In: *IFAC Proceedings Volumes* 45.15 (2012), pp. 186–191.

[115] H Suzukia and Toshiharu Sugie. "MPC for LPV systems with bounded parameter variation using ellipsoidal set prediction". In: *2006 American Control Conference*. IEEE. 2006, 6–pp.

[116] Hannu T. Toivonen. "State-dependent parameter models of non-linear sampled-data systems: a velocity-based linearization approach". In: *International Journal of Control* 76.18 (Dec. 2003), pp. 1823–1832. ISSN: 1366-5820. DOI: 10.1080/00207170310001637002. URL: http://dx.doi.org/10.1080/00207170310001637002.

[117] Roland Tóth. "Modeling and Identification of Linear Parameter-Varying Systems, an Orthonormal Basis Function Approach". PhD thesis. TU Delft, 2008.

[118] Roland Tóth, Hossam Seddik Abbas, and Herbert Werner. "On the state-space realization of LPV input-output models: Practical approaches". In: *IEEE transactions on control systems technology* 20.1 (2011), pp. 139–153.

[119] Andreas Wachter. "An interior point algorithm for large-scale nonlinear optimization with applications in process engineering." In: (2003).

[120] Nobutaka Wada, Koji Saito, and Masami Saeki. "Model predictive control for linear parameter varying systems using parameter dependent Lyapunov function". In: *The 2004 47th Midwest Symposium on Circuits and Systems, 2004. MWSCAS'04.* Vol. 3. IEEE. 2004, pp. iii–133.

[121] Zhaoyang Wan and Mayuresh V Kothare. "An efficient off-line formulation of robust model predictive control using linear matrix inequalities". In: *Automatica* 39.5 (2003), pp. 837–846.

[122] Liuping Wang. "A Tutorial on Model Predictive Control: Using a Linear Velocity-Form Model". In: *Developments in Chemical Engineering and Mineral Processing* 12.5-6 (May 2008), pp. 573–614. ISSN: 0969-1855. DOI: 10.1002/apj.5500120511. URL: http://dx.doi.org/10.1002/apj.5500120511.

[123] Jan C Willems. "Dissipative dynamical systems part I: General theory". In: *Archive for rational mechanics and analysis* 45.5 (1972), pp. 321–351.

[124] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. "A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition". In: *Journal of Nonlinear Science* 25.6 (2015), pp. 1307–1346.

[125] MO Williams, CW Rowley, and IG Kevrekidis. "A kernel-based method for data-driven Koopman spectral analysis". In: *Journal of Computational Dynamics* 2.2 (2015), pp. 247–265.

[126] S. Wollnack, H. S. Abbas, H. Werner, and R. Tóth. "Fixed-structure LPV controller synthesis based on implicit input output representations". In: *52nd IEEE Conference on Decision and Control*. 2013, pp. 2103–2108. ISBN: 0191-2216. DOI: 10.1109/CDC.2013.6760192.

[127] S. Wollnack and H. Werner. "LPV-IO controller design: An LMI approach". In: *American Control Conference*. 2016, pp. 4617–4622. ISBN: 2378-5861. DOI: 10.1109/ACC.2016.7526080.

[128] F. Wu. "Control of Linear Parameter Varying Systems". PhD thesis. University of California, Berkeley, 1995.

[129] Fen Wu and Ke Dong. "Gain-scheduling control of LFT systems using parameter-dependent Lyapunov functions". In: *Automatica* 42.1 (2006), pp. 39–50.

[130] Yang Yan and Panos Antsaklis. "Stabilizing nonlinear model predictive control scheme based on passivity and dissipativity". In: *American Control Conference*. IEEE. 2016, pp. 4476–4481.

[131] Shuyou Yu, Christoph Böhm, Hong Chen, and Frank Allgöwer. “Model predictive control of constrained LPV systems”. In: *International Journal of Control* 85.6 (2012), pp. 671–683.

[132] A. Zemouche, M. Boutayeb, and G. I. Bara. “Observer Design for Nonlinear Systems: An Approach Based on the Differential Mean Value Theorem.” In: *Proceedings of the 44th IEEE Conference on Decision and Control*. 2005, pp. 6353–6358. ISBN: 0191-2216. DOI: 10.1109/CDC.2005.1583180.

[133] Kemin Zhou, Gregory Salomon, and Eva Wu. “Balanced realization and model reduction for unstable systems”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 9.3 (1999), pp. 183–198.

List of Publications

Journal Publications

Pablo S González Cisneros and Herbert Werner. "Nonlinear model predictive control for models in quasi-linear parameter varying form". In: *International Journal of Robust and Nonlinear Control* (2020)

Pablo S. G. Cisneros and Herbert Werner. "A Velocity Algorithm for Nonlinear Model Predictive Control". In: *IEEE Transactions on Control Systems Technology* (2020)

Conference Publications

Pablo S. G. Cisneros, C Hoffmann, M Bartels, and Herbert Werner. "Linear parameter-varying controller design for a nonlinear quad-rotor helicopter model for high speed trajectory tracking". In: *American Control Conference*. IEEE. July 2016

Pablo S. G. Cisneros, Sophia Voss, and Herbert Werner. "Efficient Nonlinear Model Predictive Control via quasi-LPV Representation". In: *55th IEEE Conference in Decision and Control* (Dec. 2016)

Pablo S. G. Cisneros and Herbert Werner. "Parameter Dependent Stability Conditions for Quasi-LPV Model Predictive Control". In: *American Control Conference* (May 2017)

Pablo S. G. Cisneros and Herbert Werner. "Fast Nonlinear MPC for Reference Tracking Subject to Nonlinear Constraints via Quasi-LPV Representations". In: *20th IFAC World Congress* (July 2017)

Pablo S. G. Cisneros and Herbert Werner. "A Dissipativity Formulation for Stability Analysis of Nonlinear and Parameter Dependent MPC". in: *American Control Conference*. IEEE. June 2018

Pablo S. G. Cisneros, Aadithyan Sridharan, and Herbert Werner. "Constrained Predictive Control of a Robotic Manipulator using quasi-LPV Representations". In: *IFAC-PapersOnLine* 51.26 (2018), pp. 118–123

Pablo S. G. Cisneros and Herbert Werner. "Stabilizing Model Predictive Control for Nonlinear Systems in Input-Output quasi-LPV Form". In: *American Control Conference*. IEEE. July 2019

Pablo S. G. Cisneros and Herbert Werner. "Wide Range Stabilization of a Pendubot using quasi-LPV Predictive Control". In: *IFAC-PapersOnLine* 52.28 (2019). 3rd IFAC

Workshop on Linear Parameter Varying Systems LPVS 2019, pp. 164–169

Pablo S. G. Cisneros, Adwait Datar, Patrick Göttsch, and Herbert Werner. "Data-Driven quasi-LPV Model Predictive Control Using Koopman Operator Techniques". In: *21st IFAC World Congress* (July 2020)

Co-authored Publications

Patrick J. W. Koelewijn, Pablo S. G. Cisneros, Herbert Werner, and Roland Tóth. "LPV Control of a Gyroscope with Inverted Pendulum Attachment". In: *IFAC-PapersOnLine* 51.26 (2018), pp. 49–54

Andreas Cloppenborg, Carlos Rojas, Pablo Gonzalez Cisneros, Joachim Horn, and Herbert Werner. "Nonlinear Model Predictive Control of a PEM Fuel Cell with an integrated Li-ion Battery using qLPV". in: *15th IEEE International Conference on Control and Automation* (2019)

Horacio M Calderón, Pablo SG Cisneros, and Herbert Werner. "qLPV Predictive Control-A Benchmark Study on State Space vs Input-Output Approach". In: *IFAC-PapersOnLine* 52.28 (2019), pp. 146–151

Hossam S Abbas, Pablo SG Cisneros, Georg Männel, Philipp Rostalski, and Herbert Werner. "Practical Model Predictive Control for a Class of Nonlinear Systems Using Linear Parameter-Varying Representations". In: *IEEE Access* (2021)