Systems Theory Project Report

Nonlinear Model Predictive Control Design

Elias Niepötter (3684096)

Abstract

This report presents a study on the design and implementation of finite horizon Nonlinear Model Predictive Control (NMPC) with terminal conditions. Furthermore, it details the process of synthesizing terminal conditions using sum-of-squares (SOS) methods to ensure stability guarantees for the closed-loop system. The implementation utilizes MATLAB, CasADi, and SOSOPT for numerical optimization and analysis. Results demonstrate the effectiveness of the proposed NMPC approach, showcasing stable and efficient control performance for the ball beam system.

Keywords: Model Predictive Control, Sum-of-Squares, Terminal Conditions

1 Introduction

The goal of the project is to utilize the methods of Sum of Squares (SOS) Programming to enhance the design of nonlinear Model Predictive Control (MPC). MPC is powerful control approach for nonlinear systems based on optimal control theory. Guaranteeing closed-loop stability of MPC is a crucial aspect of the design and fundamental for all control systems. There exists different approaches to guarantee closed-loop stability of MPC, from which some are presented in this report. Those presented rely on the conecpt of *terminal conditions*. These terminal conditions, consisting of a *terminal region* and *terminal costs*, have to fullfill certain properties in order to guarantee stability. The terminal region has to be an invariant set. It is a *region of attraction* of the closed-loop system. The terminal costs have to be a *Lyapunov function* for the closed-loop system. The synthesis of such terminal conditions fullfilling the said properties is done via SOS methods. The report first introduces the basics of MPC in 2. Continuing with the synthesis of terminal conditions for MPC, focussing on SOS methods in 3. Lastly, the methods presented are applied to the ball beam system. The results are shown and discussed in 4.

2 Model Predictive Control

Model Predicitve Control is a flexible and powerful control strategy. It naturally handles nonlinearities constraints as well as multiple inputs and outputs. The underlying idea is based on optimal control theory. MPC is characterized by two key features. First, an iterative online optimization is performed to compute the next control input. Second, at each instance of time an Optimal Control Problem (OCP) is solved, therefore the time horizon of the OCP moves forward. [4]
Figure 1 illustrates the two key feastures of MPC in discrete time. At time instance t_n the OCP for the next N time steps is solved. Given that the OCP is feasible, the first optimal control input u_0^* is applied to the system. The system evolves to the next time instance t_{n+1} and the process repeats.

2.1 Problem Formulation

The discrete time finite horizon optimal control problem can be formulated as follows:

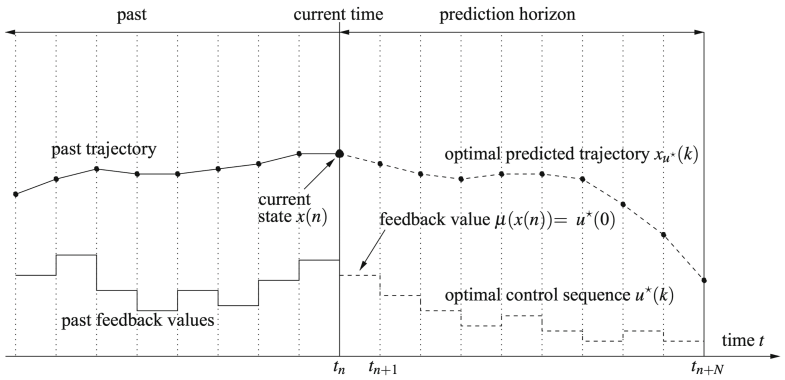

Figure 1: Moving horizon and optimization of an MPC in discrete time [4]

$$\begin{aligned} \min_{\nu} \quad & E\left(x_{k+N|k}\right) + \textstyle\sum_{i=k}^{k+N-1} F\left(x_{i|k}, u_{i|k}\right) \\ \text{s.t.} \quad & x_{i+1|k} = f\left(x_{i|k}, u_{i|k}\right), \quad x_{k|k} = x_k \\ & u_{i|k} \in \mathcal{U}, \quad i \in [k, k+N-1] \\ & x_{i|k} \in \mathcal{X}, \quad i \in [k, k+N-1] \\ & x_{k+N|k} \in \mathcal{E} \end{aligned} \tag{1}$$

where

$$J\left(\nu, x_k\right) = \underbrace{E\left(x_{k+N|k}\right)}_{\text{terminal costs}} + \underbrace{\sum_{i=k}^{k+N-1} F\left(x_{i|k}, u_{i|k}\right)}_{\text{stage costs}} \tag{2}$$

with $\mathcal{U}$ input constraints, $\mathcal{X}$ state constraints and $\mathcal{E}$ terminal region/set as the constraining sets. The OCP is solved for the prediction horizon N, the initial condition x_k at the current time step t_k, the dynamics of the system given by $f\left(x_{i|k}, u_{i|k}\right)$ and the terminal state $x_{k+N|k}$. The cost function (2) consists of the terminal costs at last time instance t_N of the optimization and the stage costs for all other time instances starting from the initial condition. The optimal control sequence is defined as

$$\nu^* = [u^*_{k|k}, \ldots, u^*_{k+N-1|k}]^T. \tag{3}$$

The control input μ at each discrete time step t_k is the first element of the optimal control sequence ν^*

$$\mu(x(t_k)) \coloneqq \nu^*(0) \tag{4}$$

See [4] and [7] for details on the formulation of the OCP resp. the MPC formulation.

2.2 Terminal Conditions for MPCs

The shown MPC includes terminal conditions. These conditions are part of the design of the MPC and are used to guarantee certain stability properties of the closed loop system. The followong Theorem connects terminal conditions and stability of the closed loop system. [7]

Theorem 1

The closed loop system is asymptotically stable if the optimal control problem is feasible at the first time instant and the following assumptions are satisfied for a terminal cost E, a terminal region $\mathcal{E}$, and a locally stabilizing control law $u_k = \varphi(x_k)$:
[A1] $E(x_k) > 0, \forall x_k \in \mathbb{R}^n \backslash \{0\}$
[A2] $\mathcal{E} \subseteq \mathcal{X}, 0 \in \mathcal{E}$
[A3] $\varphi(x_k) \in \mathcal{U}, \forall x_k \in \mathcal{E}$
[A4] $f(x_k, \varphi(x_k)) \in \mathcal{E}, \forall x_k \in \mathcal{E}$
[A5] $E(f(x_k, \varphi(x_k))) - E(x_k) \leq -F(x_k, \varphi(x_k)), \forall x_k \in \mathcal{E}.$

Several approaches based on Theorem 1 exist to ensure closed loop stability. Those investigated in this work are the following:

Zero Terminal State Constraint
The idea is to use $E(x_k) = 0$, $\mathcal{E} = \{0\}$ and $\varphi(x_k) = 0$. No linear controller is used. The stability of the MPC is guaranteed through $x_{k+N|k} = 0$. When assuming an equlibrium at the origin of the system, this ensures stability at the last step of the OCP.

Terminal Region
No terminal costs are used, but a terminal region $\mathcal{E} = \{x_k \in \mathbb{R}^n \mid x_k^T P x_k \leq \alpha\}$ is choosen as well as a locally stabilizing linear controller $\varphi(x_k) = Kx_k$. This approach is also *dual mode control*.

Terminal Region and Terminal Cost
The so-called quasi-infinite horizon MPC uses a terminal region $\mathcal{E} = \{x_k \in \mathbb{R}^n \mid x_k^T P x_k \leq \alpha\}$ and a terminal cost $E(x_k) = x_k^T P x_k$ as well as locally stabilizing linear controller $\varphi(x_k) = Kx_k$. The terminal cost E, terminal region $\mathcal{E}$ and locally stabilizing linear controller $\varphi(x_k)$ are calculated off-line. The procedure is described in section 3.

2.3 Variations of MPC

There are several variations of the MPC formulation. Two important distinctions have to be made regarding the length of the prediction horiton. One differentiate between *finite* and *infinite* horizon optimal control. presented above is the finite horizon MPC. The infinite horizon MPC, as the name implies, has an infinite prediction horizon $N \to \infty$. It can be shown that the inifinte horizon MPC asymptotically stabilizes the system, without need of any terminal conditions. Hence the name *quasi-infinite horizon* is motivated by this property. Furthermore, it should be differentiated between *stabilizing* and *tracking* MPC. The stabilizing MPC is used to stabilize the system around an equilibrium point. The tracking MPC is used to follow a reference trajectory as good as possible. For the stabilizing MPC the reference can assumed by constant zero, when the system is stable around the origin. Given the definitions above, it is possible to introduce the tracking MPC by utilizing the difference, or error, of the current state to the reference $e_k = x_k - x_{\text{ref},k}$ as a new state. In other words, the tracking MPC is a stabilizing MPC for the error dynamics of the system. Tracking MPC won't be considered in the follwing. For more details see [4].

3 Synthesis of Terminal Conditions

Followed by the introduction of terminal conditions and their application in MPC, the synthesis of terminal cost E, the terminal region $\mathcal{E}$ and the locally stabilizing linear controller $\varphi(x_k)$ is presented. The procedures are based on [2]. This section draws the connection to SOS programming. The procedure is as follows:

Step 1. Solve the linear control problem based on the jacobian linearization of the nonlinear system and obtain the feedback matrix K. Note that it is required for the system to be stabilizable around the origin. [1]

Step 2. Choose a constant $\kappa \in [0, \infty)$ satisfying

$$\kappa < -\lambda_{\max}(A_K) \tag{5}$$

where $A_K \coloneqq A + BK$ expressing the closed loop dynamics of the linearized and locally controlled system. The operator $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of a matrix. Furthermore, solve the lyapunov equation

$$(A_K + \kappa I)^{\mathrm{T}} P + P (A_K + \kappa I) = -Q^* \tag{6}$$

where $Q^* = Q + K^T R K \in \mathbb{R}^{n \times n}$ itself is positive definite and symmetric. The obtained matrix P is positive definite and symmetric. The matrices Q and R might be used for *Step 1* to obatain the linear feedback matrix K based on LQ-regulators.

Step 3. Find the largest possible α_1 s.t $Kx \in \mathcal{U} \quad \forall x \in \Omega_{\alpha_1}$.

Step 4. Find the largest possible $\alpha \in (0, \alpha_1]$ s.t. $\{x_k \in \mathbb{R}^n \mid x_k^T P x_k \leq \alpha\}$ is an inner approximation of the region of attraction of the closed loop dynamics of the linearly controlled system.

3.1 Lyapunov's second method for stability

In order to estimate a region of attraction, Lyapunov's second method for stability is used. The method makes use of the concept of *dissipativity*. Consider the dynamical system $f(x) = \dot{x}$ and a *storage function* $V : \mathbb{R}^n \to \mathbb{R}$ where $\mathbb{R}^n$ denotes the state space. The system is asymptotically stable in the sense of Lyapunov if the following conditions hold: [3]

$$V(x) = 0 \text{ if and only if } x = 0 \tag{7}$$

$$V(x) > 0 \text{ if and only if } x \neq 0 \tag{8}$$

$$\dot{V}(x) = \nabla V \cdot f(x) < 0 \quad \forall x \neq 0 \tag{9}$$

In that case, $V(x)$ is calles a *Lyapunov function*. Note that these conditions are not sufficient to prove global asymptotical stability. For that to be the case, $V(x)$ also has to be radially unbounded.

3.2 SOS Programming [3]

SOS programming is used to *verify* wether a given, potentially multivariate, polynomial $p(x) \in \mathbb{R}[x]$, where $\mathbb{R}[x]$ denotes the ring of polynomial in $x \in \mathbb{R}^n$ with real coefficients, is a sum-of-squares polynomial. A polynomial $p \in \mathbb{R}^n$ is sum-of-squares if there exists $q_1, \ldots, q_m$ s.t.

$$p(x) = \sum_{i=1}^{m} q_i^2(x) \tag{10}$$

The set of SOS polynomials will be denoted by $\Sigma[x]$. If a polynomial is sum-of-squares, it is also a nonnegativ polynomial $p(x) \geq 0$. Subsequently, if we want to proof wether a polynomial is nonnegative or not, we can now relax this problem to the question, wether the the polynomial is sum-of-squares or not. The advantage of this relaxation over the orginal problem is that solving the problem to verify if a polynomial is sum-of-squares can be done in polynomial time. Proofing if a polynomial is nonnegative is a NP-hard problem. The verification on whether a polynomial is sum-of-squares or

[1] A system is said to be stabilizable if all uncontrollable states are stable by themself or vice versa defined, all unstable states are controllable [6]

not is done by solving a Semidefinite Program (SDP).

Lemma 1. Let $p(x) \in \mathbb{R}[x]$ be a polynomial of degree d. Then, $p(x) \in \Sigma[x]$ if and only if there exists a symmetric matrix $Q \in \mathbb{S}^l$ being positive semidefinite (all eigenvalues of Q are nonnegative) such that

$$p(x) = z(x)^\top Q z(x) \tag{11}$$

for some monomial vector $z(x) \in \mathbb{R}[x]^l$ with $\deg(z(x)) \leq \frac{d}{2}$.

Using Lemma 1, we further relaxed the problem of verifying if a polynomial is sum-of-squares to the problem of finding a positive semidefinite matrix Q. This task is done via a SDP.

3.3 Sublevel set optimization

In *Step 3* and *4* one tries to find the largest sublevel set of a given function satisfying certain constraints. These optimization problems can be formulated to one single problem. The idea is to setup a nonlinear root search for the parameter α. An iterative algorithm, e.g. a bisection algorithm, is used to maximize the parameter. The problem can be formulated as follows:

$$\max_{\alpha} \quad \text{Volume}\left(\{x \in \mathbb{R}^n \mid x^T P x \leq \alpha\}\right) \tag{12}$$

The resulting constraints for the optimiztion are

$$s_j \in \Sigma[x] \quad \forall j[1 \dots 2m+1] \tag{13}$$

$$-s_{1\dots m}(-x^T P x + \alpha) + (K_i x + u_{\max,i}) \in \Sigma[x] \quad \forall i \in [1, m] \tag{14}$$

$$-s_{m+1\dots 2m}(-x^T P x + \alpha) + (-K_i x - u_{\min,i}) \in \Sigma[x] \quad \forall i \in [1, m] \tag{15}$$

$$V - \epsilon(x^T x) \in \Sigma[x] \tag{16}$$

$$s_{2m+1}(V - \alpha) - \nabla V \cdot f - \epsilon(x^T x) \in \Sigma[x] \tag{17}$$

where m is denoting the number of inputs to the system. Note that K_i is the i-th row vector of the feedback matrix K. This enables asymetric input constraints $u_{\max,i}$ and $u_{\min,i}$. Costraint (18) and (19) are base on 3.1. These constraints ensure dissipativity of the system inside the found sublevel set and therefore asymptotical stability. The polynomial $V = x^T P x$ is the lyapunov function in that sense. Note that V is indeed a lyapunov function because the matrix P is positive definite. According to 3.1 it is necessary to proof positive definiteness to ensure stability, instead of positive semidefiniteness. The term $\epsilon(x^T x)$ ensures positive definiteness for (18) and (19) instead of just positive semidefiniteness which would follow from (18) and (19) being sum-of-squares. The constraints given above form a set containment problem. The setup of the constraints and the problem are based on the following procedure, called *Generalized S-procedure*.

Generalized S-procedure [5]

Given $h, f_0, \dots, f_r \in \mathbb{R}[x]$, if there exist $p \in \mathbb{R}[x]$, and $s_1, \dots, s_f \in \Sigma[x]$ s.t.

$$\text{ph} \ -\sum_{j=1}^{r} s_j f_j + f_0 \in \Sigma[x] \tag{18}$$

then the following set containment holds:

$$\{x \in \mathbb{R}^n \mid h(x) = 0, f_1(x) \geq 0, \dots, f_r(x) \geq 0\} \subseteq \{x \in \mathbb{R}^n \mid f_0(x) \geq 0\} \tag{19}$$

The verification of the set containment can be implemented inside a bisection algorithm to test the containment for a given α. That way, the optimization problem (12) is solved.

4 Exemplary Application

Based on an exemplary system a MPC with terminal conditions is designed using the procedures described above.

4.1 Setup and Implementation

The exemplary system is the ball beam system. It consists of a ball rolling on a beam. The beams angle is controlled by a motor. Gravity acting on the ball accelerates it accoding to the angle of the beam. The system is described by the following equations of motion:

$$0 = \left(\frac{J}{R^2} + m\right)\ddot{r} + mg\sin(\alpha) = -mr\dot{\alpha}^2 \tag{20}$$

States are chosen as $x = \begin{bmatrix} r & \dot{r} & \alpha & \dot{\alpha} \end{bmatrix}^T$ and the input as $u = \ddot{\alpha}$. The systems dynamics are given by

$$\dot{x} = f(x,u) = \begin{bmatrix} x_2 \\ mx_1x_4^2 - mg\left(\frac{x_3^5}{120} - \frac{x_3^3}{6} + x_3\right)\cdot k \\ x_4 \\ u \end{bmatrix} \tag{21}$$

Note that the sin() term is approximated by a Taylor series epansion of degree five in order to obtain a polynomial system. The Taylor approximation for the sin() only holds in a certain viscinity around the origin. In the case demonstrated, the simulation is terminated when the angle of the beam α is $|90°|$. In the interval $[-90°, 90°]$ the approximation is valid. The constant k combines the physical parameters of the system. Table 1 gives an overview of the different MPC controler setups used.

Table 1: overview of additional terms

Controller	**Name**	**Explanation**
QIH	Quasi-Inifnite Horizon	terminal region + costs
TR	Terminal Region	just terminal region
TC	Terminal Constraint	terminal constraint ends in equil.
NTC	No Terminal Conditions	no terminal conditions at all

Remark 1. In general, it might not be the case that the simulation temrinates before the Taylor approximation is not valid anymore. This could lead to an additional constraints for the sublevel set optimization (12)

$$-s_{1\ldots l}(-x^TPx + \alpha) + (x_n + \gamma_{\max,r}) \in \Sigma[x] \quad \forall l \in [1,r] \tag{22}$$

$$-s_{1\ldots l}(-x^TPx + \alpha) + (-x_n - \gamma_{\min,r}) \in \Sigma[x] \quad \forall l \in [1,r] \tag{23}$$

where $\gamma_{\max,r}$ and $\gamma_{\min,r}$ are the bounds of the interval where the Taylor approximation holds for a state x_n. The number of constraints depends on the number of Taylor approximations r used.

The TC and NTC controller consists just of the MPC. The QIH and TR controller have linear controller which takes over as soon as they enter the terminal region. The time step for the closed loop simulation is set to $\Delta t_{\text{cl}} = 0{,}01\,\text{s}$. The time step for the MPC is set to $\Delta t_{\text{mpc}} = 0{,}05\,\text{s}$. The prediction horizon is set to $N = 1\,\text{s}$. No noise is added to the states and the true model is known. The stage costs are chosen to

$$F\left(x_{i|k}, u_{i|k}\right) = \frac{1}{2}\|x_{i|k}\|_2 + \frac{1}{2}\|u_{i|k}\|_2. \tag{24}$$

4.2 Implementation Details

The example is implemented in MATLAB R2023b. The temrinal conditions are synthesized with help of the open-source software package SOSOPT [8]. SOSOPT is a MATLAB toolbox for solving sum-of-squares problems. It is used to verify the set containment described in (12). The MPC is imeplemented with the help of the open-source software package CasADi [1]. CasADi is software framework for numerical optimization, especially suited for optimal control and therefore MPC. The closed simulation is implemented via a simple Euler integration scheme. The optimal control problem is solved through a multiple shooting method. The solver used for the nonlinaer optimization problem is IPOPT [9]. IPOPT is a robust interior-point solver that can be interfaced with CasADi.

4.3 Results

In the following, the results of the simulations as well as the synthesis of the terminal conditions are presented.

Synthesized terminal conditions

The steps described in 3 are applied to the ball beam system. Different linear controllers are designed based on the jacobian linearization. Figure 2 shows the resulting terminal region for two different linear controllers. The LQR offers a larger region of attraction and will be used for the QIH and TR controllers.

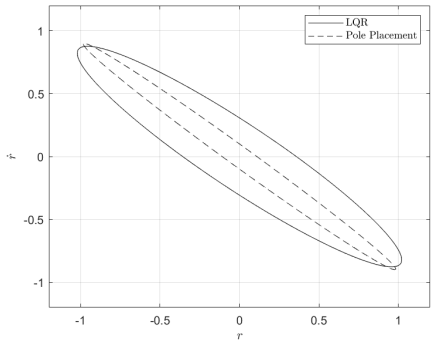

Figure 2: comparison of the synthesized terminal region with a LQR controller and a pole placement

Note that 2 shows the sliced terminal region for the states r and $\dot{r}$. The Q and R matrix for the LQR controller are chosen as $Q = 0.25 \cdot \mathbf{I}^4$, where $\mathbf{I}^4$ is dnoting the 4×4 identity matrix and $R = 1$. The matrices are the same as for Lyapunov equation in *Step 2*.

MPC Performance Analysis

Figure 3 to 6 show the results of the simulations. Most obvious is the rather poor performance of the NTC controller. It has highest excitations especially in state $x_3 = \alpha$ and $x_4 = \dot{\alpha}$ (see fig. 3).
Compared to the other controllers, the NTC controller also takes longest to fully converge to the equilibrium. However, the NTC controller is still able to stabilize the system. The most performant controller is the TC controller. It has the largest occuring input to the systems and the fastest covergence. This result was to expect, because the TC controller (the MPC) has to bring the system to the equilibrium in the given time horizon of 1 s.
The QIH controller and the TR controller show similar performance. Compared to the NTC and TC controller, their perofemance is in between. With their respective design approach in mind, this result was also to expect. The MPC of the QIH and TR controller just has to get the system into the terminal region.

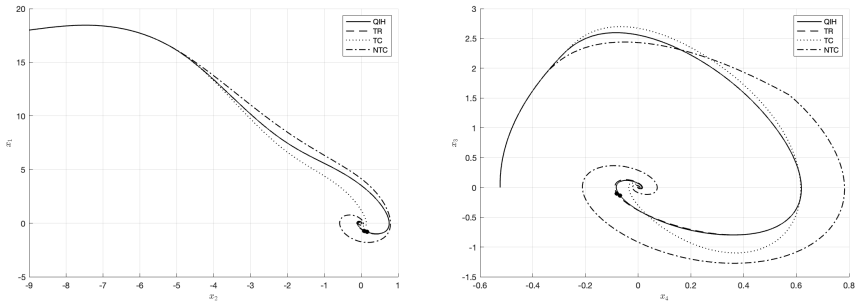

Figure 3: state space trajectories for defined control setups

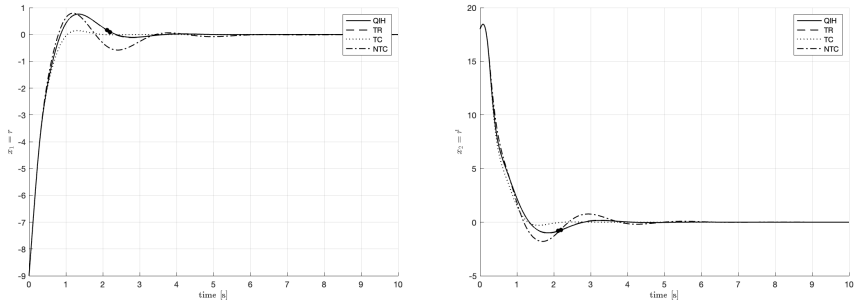

Figure 4: trajectories for states x_1 and x_2

Figure 6 shows that the QIH and TR controller have overall the lowest input to the system which can be considered as more efficient. Furthermore, the QIH and TR controller have very similar trajectories and behavior. The terminal costs don't seem to have an significant impact for this specific example. The black dots in all result figures show the switching point of the dual mode controller, so the point from where on the linear controller controls the system. The QIH controller enters the terminal region slightly earlier than the TR controller.

5 Conclusion & Outlook

This report covers the synthesis and exemplary application of terminal conditions for MPC using SOS programming. The underlying theory as well as the application is covered. SOS programmming is successfully used to estimate the terminal region for the MPC design. The example covered shows that a well desiged terminal region can lead to a more efficient control of the system. A drawback of the presented approach is the necessity of a system that is stable around the origin. This approach is therefore not suitbale for systems with periodic limit cycles. It can further be quite difficult to obtain a terminal region because of a dependency on the linear controller. Different linear controllers lead to different terminal regions. Obtaining a satisfying terminal region may be an iterative process. The presented approach is also limited to polynomial systems which can lead to the use of Taylor series expansions, as shown in the exmaple. To give an outlook for possible improvements, instead of optimizing the sublevel set for a fixed shape defined by P a $V - s$ iteration could be implemented. Using such an approach could allow for a more flexible terminal region, not restricted by to a shape that depends on the linear controller. Furthermore, in [7] it is demonstrated that the finite horizon OCP can be formulated as polynomial optimization problem and solved with SOS programs. It is shown that this can lead to significant performance improvements compared to solving the classical nonlinar optimization problem in MPC.

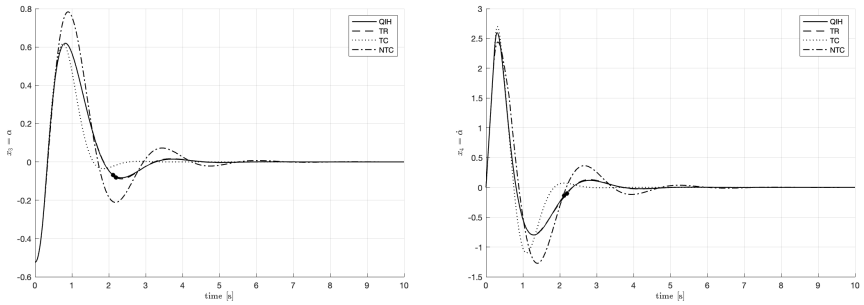

Figure 5: trajectories for states x_3 and x_4

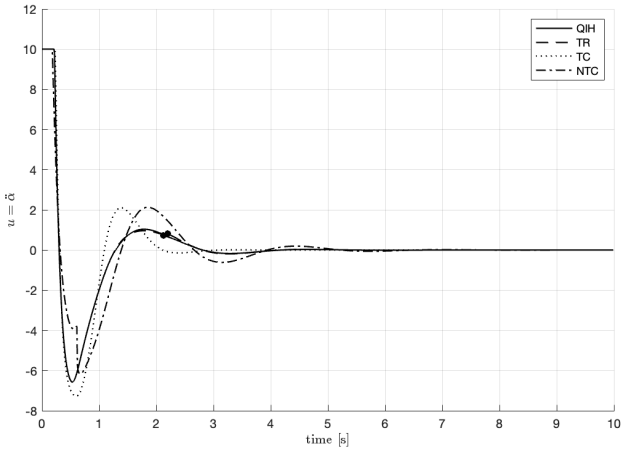

Figure 6: trajectories for input u

References

[1] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[2] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.

[3] Torbjørn Cunis and Renato Loureiro. Systems theoretical methods for flight control, 2023.

[4] Jürgen Pannek Lars Grüne. *Nonlinear Model Predicitve Control*. Springer Cham, 2 edition, 2016.

[5] Renato Loureiro. Nonlinear sos programming. Presentation at the Seminar on System Theoretic Methods for Flight Control, November 2023.

[6] Jan Lunze. *Regelungstechnik 2: Mehrgrößensysteme, Digitale Regelung*. Springer Vieweg, Berlin, Heidelberg, 9 edition, 2016. Published on October 1, 2016.

[7] T. Raff, C. Ebenbauer, R. Findeisen, and F. Allgöwer. Nonlinear model predictive control and sum of squares techniques. *Fast Motions in Biomechanics and Robotics, M. Diehl and K. Mombaur (eds.)*, 10:325–344, 07 2007.

[8] Peter Seiler. Sosopt: A toolbox for polynomial optimization, 2013.

[9] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.