Master thesis

Investigating the effect of set partitioning strategies on the performance of robust model predictive controllers for monotone systems

Thomas Jan Kalenkiewicz
Matr. 202520

First examiner:	Prof. Dr.-Ing. Sergio Lucia
Second examiner:	Prof. Dr.-Ing. Norbert Kockmann
Advisor:	Moritz Heinlein
Year:	2025
Ident.:	BCI-PAS-2025-M01

Technische Universität Dortmund
Faculty of Biochemical and Chemical Engineering
Laboratory for Process Automation Systems

technische universität dortmund

process automation systems

Abstract

In an increasingly complex world, control strategies must also grow in sophistication. Today's major concerns, environmental protection, efficiency, and resource conservation, require ever more capable methods. Traditional control frameworks, such as linear control, remain widespread in industry thanks to their well-established theoretical foundations and relatively simple implementation. They often yield satisfactory results, but when applied to complex non-linear systems, they can exhibit limitations in control performance, constraint handling, and even stability. Advanced Process Control (APC) techniques and, in particular, Model Predictive Control (MPC) address these shortcomings and represent a key element of future control engineering [1].

Although MPC implementation is non-trivial and may require substantial computational resources depending on the model, it achieves very satisfying control performance, readily handles multiple input, multiple output (MIMO) systems, and offers exceptional flexibility. Models can be linear or highly non-linear in various application domains, with constraints incorporated directly and tailored to the specific control problem within the MPC optimization [2], [3]. Due to the growing computational power and the ever-growing availability of microprocessors, the scope of application of MPC has expanded far beyond the process industries [4]. Today, MPC is used in many different fields like power electronics, embedded systems, autonomous driving, and heating, ventilation, and air conditioning (HVAC). In the HVAC domain, its performance advantages are particularly clear: robust MPC (RMPC) can regulate indoor temperatures under significant parametric uncertainties, such as fluctuating outdoor temperatures, while enforcing all constraints and achieving substantial energy savings [5]. In the process industry, there are also many examples in which MPC leads to considerable energy savings due to operation closer to the constraints [6].

This work presents three different set-partitioning strategies for the sets of future reachable system states. Through set partitioning, feedback is incorporated into model predictions. Each partitioning strategy is evaluated on two systems: a non-linear, monotone double integrator [7] and a continuous stirred tank reactor (CSTR), which serves as a significantly more complex and realistic application model [8]. Monotonicity is a key property as it enables efficient computation of future reachable sets. Since the CSTR is non-monotone, mixed monotonicity is employed by determining a decomposition function that splits the system dynamics into monotonically increasing and monotonically decreasing components [9]. Finally, a decomposition function derived from a monotone neural network model is investigated in terms of yielding tighter bounds on possible future state sets.

Contents

Notation

Numbers and Arrays

$\underline{a}$	Lower bound vector in $\mathbb{R}^n$ for definition of Tightness
$\overline{a}$	Upper bound vector in $\mathbb{R}^n$ for definition of Tightness
E_{A12}/R	Activation energy for reaction 1 and 2
E_{A13}/R	Activation energy for reaction 3
A	System matrix of the double integrator
A_r	surface area of the cooling jacket of the CSTR
B	Input matrix of the double integrator
c_A	Concentration of A in the CSTR in mol/l
c_{A0}	Inlet concentration of A in the CSTR in mol/l
c_B	Concentration of B in the CSTR in mol/l
$C_{p,k}$	Heat Capacity of the coolant
C_p	Heat capacity of the reactant mixture
F	Inlet-flow rate of c_A normalized by the CSTR's volume
F	Disturbance matrix of the double integrator
K	Feedback gain
$k_{1,0}$	reaction constant for reaction 1
$k_{2,0}$	reaction constant for reaction 2
$k_{3,0}$	reaction constant for reaction 3
k_w	Heat transfer coeefficient cooling jacket of the CSTR
u	Vector of control inputs
u_{min}	Vector of control inputs for lower bounding in $\mathbb{U}$
u_{max}	Vector of control inputs for upper bounding in $\mathbb{U}$

$\underline{u}$	Vector of control inputs for lower bounding
$\overline{u}$	Vector of control inputs for upper bounding
$u_{[0:N-1]}$	A sequence of state vectors from 0 to $N-1$, such that $u_{[0:N-1]} = (u_0, \ldots, u_{N-1})$
u_0	Initial value of u
u_s	Setpoint for u
L	Number of layers in a NN
m_k	coolant mass in the CSTR
N	Prediction horizon
N_{sim}	Simulation steps in a MPC-loop
n_x	Number of system dimensions
n_u	Number of input dimensions
n_p	Number of uncertainty dimensions
N_s	Total number of subregions
N_{s1}	Total number of subregions in one step for Alternating Variable Partitioning
N_{s2}	Total number of subregions in second step for Alternating Variable Partitioning
N_r	Robust Control Horizon
n_{c_j}	Number of partitions in dimension j
p	Vector for uncertain parameters
p_{min}	Vector of uncertainties for lower bounding in $\mathbb{P}$
p_{max}	Vector of uncertainties for upper bounding in $\mathbb{P}$
$\underline{p}$	Vector of uncertainties for lower bounding
$\overline{p}$	Vector of uncertainties for upper bounding
$p_{[0:N]}$	A sequence of state vectors from 0 to N, such that $p_{[0:N]} = (p_0, \ldots, p_N)$
Q	Positive Semidefinite Weight Matrix
Q_k	Heat removal through the cooling-jacket of the CSTR in kJ/mol
R	Positive Semidefinite Weight Matrix
$\underline{R}$	Lower bound vector in $\mathbb{R}^n$ for definition of Tightness
$\overline{R}$	Upper bound vector in $\mathbb{R}^n$ for definition of Tightness

ΔH_{AB}	Reaction enthalpy for reaction 1
ΔH_{BC}	Reaction enthalpy for reaction 2
ΔH_{AD}	Reaction enthalpy for reaction 3
ρ	Density of the reaction mixture in the CSTR
t	Time
θ	Temperature in the CSTR in °C
θ_K	Temperature in the cooling-jacket of the CSTR in °C
x	Vector of states
$\underline{x}$	Vector of states for lower bounding
$\overline{x}$	Vector of states for upper bounding
x_{min}	Vector of states for lower bounding in $\mathbb{X}$
x_{max}	Vector of states for upper bounding in $\mathbb{X}$
$x_{[0:N]}$	A sequence of state vectors from 0 to N, such that $x_{[0:N]} = (x_0, \ldots, x_N)$
x_0	Initial value of x
x_s	Setpoint for x
δ	Sampling time
μ	Optimal control policy
$\mu_{[0:N-1]}$	A sequence of optimal control policies
ω_i	Weighting factor of a distinct scenario within the cost function in a scenario tree
W_{2d}^{+}	Positive Weights of monotone NN in last layer
W_{2d}^{-}	Negative Weights of monotone NN in last layer
W	Weights of monotone NN
V	Terminal Cost
V_r	reactor volume of the CSTR

Sets and Graphs

$\mathbb{R}$	The set of real numbers
Ω	The set of the terminal region

IR	Interval of an Hyperrectangle-overapproximation
$\mathcal{M}(x)$	Class of control policies, e.g. affine control laws
$\mathcal{X}_i$	Reachable set of states at time i
$\mathcal{X}_{[0:N]}$	Sequence of reachable sets
$\mathcal{U}_i$	Set of control inputs at time i within the corresponding $\mathcal{X}_i$
$\mathcal{X}_i$	Reachable set of system states at time i
$\mathcal{X}_{RCIS}$	Robust Control Invariant set (RCIS)
$N-step\,\mathcal{X}_{RCIS}$	N-step RCIS
$\mathcal{X}_{i,RCIS}$	I-th set of $N-step\,\mathcal{X}_{RCIS}$
$\mathbb{S}$	Uncertainty sets in tube-based MPC
$\mathbb{S}_i$	Uncertainty set in tube-based MPC for time i
$\mathbb{X}_{tight}$	Tightened state set for tube-based MPC
$\mathbb{U}_{tight}$	Tightened control set for tube-based MPC
$\mathcal{S}$	Subregions in one dimenion

Indexing

i	Index for point in time
j	Dimensional index
t	Index for counting the subregions in $\mathcal{S}_j^t$
t_{CL}	Simulation time per MPC-iteration
u_i	Control inputs at time i
u_i^s	Control inputs belonging at time i belonging to subregion s
u_i^p	Control inputs belonging at time i belonging to uncertainty scenario p
p_i	Uncertainty realization at time i
x_i	States at time i
$x_i^{\pm}$	States at the lower left and upper right corner of a hyperrectangle at time i
$p_i^{\pm}$	Uncertainty realizations at the lower left and upper right corner of a hyperrectangle

τ	Index for time bounded to a given Interval
$u_{\text{RCIS},i}^{[1:N_{s,i}]}$	Control input for subregion s in i-th set of $\mathcal{X}_{RCIS}$
$\mathcal{X}_i^s$	States in subregion s at time i
$\mathcal{X}_{i+1}^{s'}$	States in subregion s' at time i
d_j	Component j of decomposition function with $j \in \{1, \ldots, n_x\}$
$d_{min,j}$	Component j of lower bounding decomposition function with $j \in \{1, \ldots, n_x\}$
$d_{max,j}$	Component j of upper bounding decomposition function with $j \in \{1, \ldots, n_x\}$
$\mathcal{I}$	Interval listing all scenarios in a scenario-tree

Calculus

$\frac{dc_A}{dt}$	Differential equation of c_A
$\frac{dc_B}{dt}$	Differential equation of c_B
$\frac{d\vartheta}{dt}$	Differential equation of ϑ
$\frac{d\vartheta_K}{dt}$	Differential equation of ϑ_K
$\frac{dc_{A,\text{min}}}{dt}$	Min-decompostionfunction of c_A
$\frac{dc_{A,\text{max}}}{dt}$	Max-decompostionfunction of c_A
$\frac{dc_{B,\text{min}}}{dt}$	Min-decompostionfunction of c_B
$\frac{dc_{B,\text{max}}}{dt}$	Max-decompostionfunction of c_B
$\frac{d\vartheta_{\text{min}}}{dt}$	Min-decompostionfunction of ϑ
$\frac{d\vartheta_{\text{max}}}{dt}$	Max-decompostionfunction of ϑ
$\frac{d\vartheta_{K,\text{min}}}{dt}$	Min-decompostionfunction of ϑ_K
$\frac{d\vartheta_{K,\text{max}}}{dt}$	Max-decompostionfunction of ϑ_K

Functions

$x_{i+1} = F(x_i, u_i, p_i)$	Discrete-time system dynamics
$\dot{x} = f(x, u, p)$	Continuous-time system dynamics

$\mathcal{X}_i(x_0, \mathbb{U}_{i-1}, \mathbb{P}) = g(\mathcal{X}_{i-1}(x_0, \mathbb{U}_{i-2}, \mathbb{P}))$	Function to recursively propagate reachable sets from former reachable sets
$J\big(x(t),\, u_{[0:N-1]},\, N\big)$	Performance objective in the context of an optimization problem
J_{CL}	Quadratic closed-loop cost function
$L(x, u)$	Stage cost in $J\big(x(t),\, u_{[0:N-1]},\, N\big)$
$P_N(x_0)$	Optimal Control Problem initialized at x_0 for nominal case
$P_{ROC,N}(x_0)$	General formulation of ROCP
d	Decomposition function
d_{min}	Lower bounding decomposition function
d_{max}	Upper bounding decomposition function
d_{NN}	Decompositionfunction of NN
z_1	Forward Pass through NN
z_1	Forward Pass through NN
$P_{\text{CL}}^{\text{alt-const\_1}}(x_0)$	ROCP of Alternating Constant Partitioning in step 1
$P_{\text{CL}}^{\text{alt-const\_2}}(x_0)$	ROCP of Alternating Constant Partitioning in step 2
$P_{\text{CL}}^{\text{alt-var\_1}}(x_0)$	ROCP of Alternating Variable Partitioning in step 1
$P_{\text{CL}}^{\text{alt-var\_2}}(x_0)$	ROCP of Alternating Variable Partitioning in step 2
tanh(x)	hyperbolic tangent $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
f_L	Mapping function of NN
f_{NN}	Prediction function of NN

Datasets and Distributions

$\mathbb{X}$	A Closed set of considered states
$\mathbb{X}_d$	State set of the double integrator
$\mathbb{X}_{CSTR}$	State set of the CSTR
$\mathbb{X}_f$	Terminal set of states

$\mathbb{U}$	A Closed set of considered control inputs
$\mathbb{U}_d$	Set of control inputs of the double integrator
$\mathbb{U}_{CSTR}$	Input set of the CSTR
$\mathbb{P}$	A compact set of considered uncertainty realizations
$\mathbb{P}_d$	Uncertainty set of the double integrator
$\mathbb{P}_{CSTR}$	Uncertainty set of the CSTR

Abbreviations

A	Cyclopentadiene
APC	Advanced Process Control
B	Cyclopentenol
C	Cyclopentanediol
OL	Open Loop
CL	Closed Loop
CSTR	Continuous Stirred Tank Reactor
D	Dicyclopentadiene
DAE	Differential-Algebraic Equation
DMC	Dynamic Matrix Control
DE	Difference Equation
GPC	Generalized Predictive Control
HJB	Hamilton-Jacobi-Bellman
LQ	Linear Quadratic
MPC	Model Predictive Control
MSE	Mean Squared Error
NLP	Nonlinear Program
NN	Neural Network
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
RCI	Robust Controlled Invariance
RCIS	Robust Control Invariant Set

RMPC	Robust Model Predictive Control
ROCP	Robust Optimal Control Problem

List of Figures

List of Tables

Chapter 1

Introduction

In modern control engineering, the demand for advanced control strategies grows in response to increasingly complex system dynamics and increasing performance requirements. This thesis is positioned within the field of Advanced Process Control, with a particular emphasis on Robust Model Predictive Control (RMPC). The following two sections provide both a historical perspective on RMPC and an outline of the specific research objectives pursued in this work.

1.1 Historical Background

Advanced Process Control (APC) refers to relatively new and innovative control methods. When classical techniques such as PID algorithms fail or complex systems must be regulated near their operational constraints, APC offers a genuine alternative. However, APC should be understood not only as a competitor to established methods but also as a complement. For example, it can operate on higher control layers and steer PID-based loops by providing dynamic set-point commands[4],[10]. Model Predictive Control (MPC) is perhaps the best-known APC technique and is often used synonymously with APC, even if somewhat inaccurately [3]. Its roots lie in the chemical and petrochemical industries of the early 1980s, where dynamic matrix control (DMC) and generalized predictive control (GPC) pioneered multivariate, constrained regulation of process plants [1]. At that time, limited computing power severely restricted the applicability of MPC, especially to more complex and fast system dynamics [4],[5]. During the past decades, dramatic increases in computational capacity have enabled the adoption of MPCs in contexts demanding both optimal control and stringent operational constraints. MPC has demonstrated cost savings of the order of tens of percent in numerous industrial case studies [5],[10]. Building on this historical foundation, MPC can be defined as a model-based control strategy that explicitly incorporates future predictions and hard constraints into a receding-horizon optimization aiming to minimize a cost-function. Its flexibility allows the use of linear or highly non-linear models and the constraints are tailored directly to the control problem [11]. In practice, however, certain model parameters can only be estimated within some bounds rather than known exactly. Robust Model Predictive Con-

trol (RMPC) extends nominal MPC by explicitly accounting for future realizations of uncertainties in its state predictions. This ensures that all forecast trajectories respect their constraints, a capability of vital importance in safety-critical applications [12]. With these fundamentals in place, state-of-the-art set-partitioning approaches and two novel set-partitioning strategies that reduce problem complexity while preserving performance are introduced.

1.2 State of the Art and Motivation

Fundamentally this work investigates RMPC. Sets of possible future states are predicted at each step of the prediction horizon, which are called reachable sets. One of the computationally most efficient ways to overapproximate the reachable sets is via multidimensional intervals, which are also termed hyperrectangles [13]. To efficiently compute these hyperrectangles, monotonicity [14] or mixed-monotonicity [15] is utilized. For non-monotone systems, a decomposition function separates the system dynamics into monotonically increasing and decreasing components. Following the works [14] and [15], these hyperrectangular reachable states are divided into subregions, each assigned a specific control input in order to incorporate feedback into the predictions, allowing the RMPC to react to different possible future realizations of parametric uncertainties, a concept known as recourse [14],[15]. This set-partitioning approach, hereafter full partitioning, keeps the number of subregions constant on the prediction horizon, resulting in a linear growth in complexity of the optimization problem with horizon length. However, in the full partitioning approach, the state dimensions chosen to partition are partitioned in the same way in each prediction step, leading to exponential growth of complexity with the state dimension [15]. To address this, two alternative set partitioning strategies are proposed in this work in section 2.8.

Alternating-Constant Partitioning: Alternates set partitioning across state dimensions at each prediction step, maintaining a constant number of subregions per prediction step, but with alternating alignment of these subregions since different dimensions are partitioned in each prediction step.

Alternating-Variable Partitioning: Is a more flexible version of the alternating-constant partitioning approach. Here, the set partitioning across state dimensions per prediction step is alternating, and in addition the number of subregions varies from one prediction step to the next. So, the number of subregions per prediction step switches between higher and lower partition resolutions.

These methods aim to achieve a favorable trade-off between control performance and optimization complexity and avoid having a large number of subregions. All three approaches are evaluated first on a monotone double integrator [7], then on a more complex non-monotone continuous stirred-tank reactor (CSTR) [8]. Afterwards, an approximate decomposition function for the CSTR is extracted from a monotone neural

network. For the sake of less conservative set approximations, the tightness of the approximate decomposition function should be compared with another decomposition function for the CSTR, based on interval arithmetics [16].

Chapter 2

Key concepts of Model Predictive Control

In this chapter, the fundamental concepts and methodologies that form the basis for the subsequent discussions of feedback-based set partitioning strategies and neural network results are presented.

2.1 Introduction to Model Predictive Control (MPC)

Model Predictive Control (MPC) is a widely adopted control strategy that emerged in the process industries during the 1980s. It is particularly suited for systems subject to constraints on state or control inputs. At its core, MPC uses a mathematical model of the plant to predict future system behavior and, based on these predictions, solves an optimization problem to compute control signals that respect constraints. Traditional controllers typically rely on previous measurements to adjust inputs and cannot explicitly incorporate constraints during computation [11].

In contrast, MPC repeatedly solves a finite-horizon optimal control problem (OCP) online, updating at each sampling instant. The solution comprises an open-loop trajectory of predicted states and the corresponding control inputs that minimize a predefined cost function, often the Mean Squared Error (MSE) between the predicted states to a predefined setpoint. Once the OCP is solved, only the first control input of the optimal sequence is applied to the real system. The procedure is then repeated from the new measured or estimated state, forming the characteristic MPC feedback loop.

Concretely, MPC operates according to these three steps [17]:

1. **State Estimation:** Measure or estimate the state of the current system.
2. **Prediction and Optimization:** Using the system model, calculate the optimal future state and control trajectories over a prediction horizon.

3. **Receding Horizon Implementation:** Apply only the first control input of the computed sequence to the real system and update the state after the sampling interval and return to step 1.

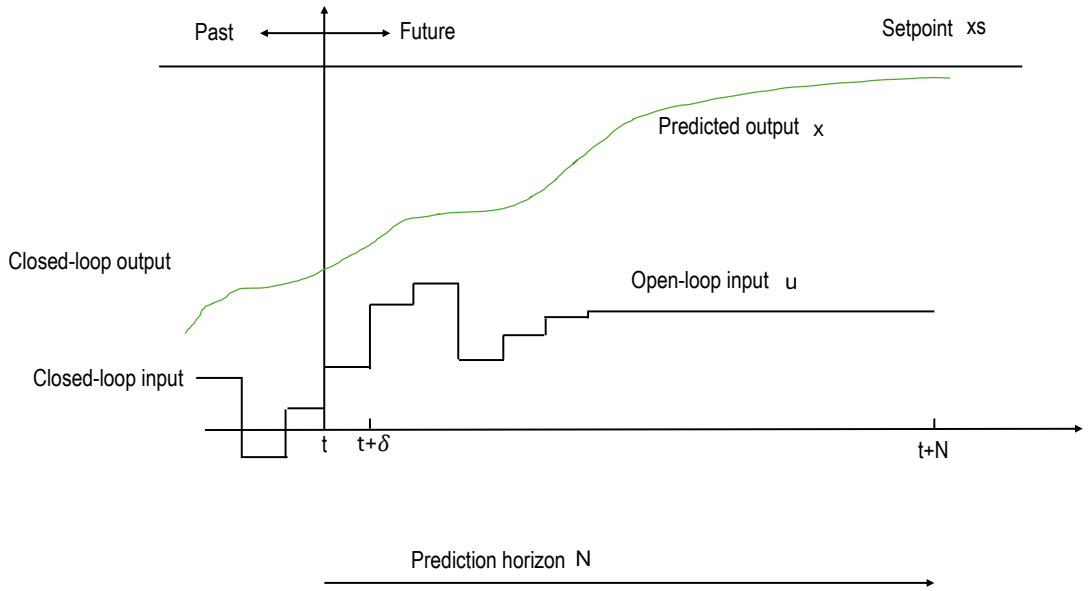

Figure 2.1: Schematic illustration of an MPC example trajectory [11]

Figure 2.1 schematically illustrates this process. Along the time axis, the plot is divided into the past and the future relative to the current time t. On the vertical axis, a representative state variable is shown, which the controller aims to drive toward a setpoint x_s. At time t, the state of the system $x(t)$ is measured or estimated. An open-loop control sequence $u_{[0:N-1]}$ is then computed over a prediction horizon (N), minimizing a cost function that predicts the system's evolution $x_{[0:N]}$ over N. The term "Open-loop" implies that these inputs are optimized purely on the basis of the model and the current state $(x(t_0))$.

The first element of $u_{[0:N-1]}$ is applied to the real system now known as the closed-loop input, and after a sampling time δ, the actual state is again measured. This new state initializes the next OCP, shifting N forward by δ, thus closing the feedback loop. The actual resulting system trajectory represents the closed-loop behavior and appears on the left side of the figure, representing the past behavior [11].

For the system model, one typically employs a set of ordinary differential equations (ODE), a set of differential-algebraic equations (DAE) or in the discrete case a set of difference equations (DE). The control inputs are discretized over δ and held constant during each interval to render the OC numerically tractable [18].

In the following, the essential elements of MPC in accordance with Figure 2.1 are presented by formulating the discrete-time OC:

$$\underset{u_{[0:N-1]}}{\text{minimize}} \quad J\big(x(t),\, u_{[0:N-1]},\, N\big) = \sum_{i=0}^{N-1} L\big(x_i,\, u_i\big) + V\big(x_N\big) \tag{2.1a}$$

subject to (s.t.):

$$x_{i+1} = F(x_i, u_i) \quad \text{(system dynamics)} \tag{2.1b}$$

$$x(t_0) = x_0 \quad \text{(initial condition)} \tag{2.1c}$$

$$u(\tau) \in \mathbb{U},\ \tau \in [0,\, N-1] \quad \text{(control constraints)} \tag{2.1d}$$

$$x(\tau) \in \mathbb{X},\ \tau \in [0,\, N] \quad \text{(state constraints)} \tag{2.1e}$$

$$x(N) \in \Omega \quad \text{(terminal constraint)} \tag{2.1f}$$

The cost function J in (2.1a) denotes the total cost on the prediction horizon. J is a performance metric that should be minimized by a sequence $u_{[0:N-1]}$. Generally, L and V could assume different forms to capture various performance criteria. The quadratic form in (2.2) is a common choice for setpoint tracking or regulation objectives, where Q and R are positive semidefinite weight matrices, x_s and u_s are the desired steady-state values, and $V\big(x(t+N)\big)$ acts as the terminal cost measuring deviation from the setpoint at the end of the horizon [19].

$$L\big(x,u\big) \;=\; \big(x(t)-x_s\big)^T Q\,\big(x(t)-x_s\big) \;+\; \big(u(t)-u_s\big)^T R\,\big(u(t)-u_s\big) \tag{2.2}$$

The constraints in (2.1b)–((2.1f)) are typical components of an MPC-OCP. They are explained shortly in the following:

- **System Dynamics** (2.1b)**:** defines the model, where the model usually includes multiple outputs and various control inputs
- **Initial Condition** (2.1c)**:** Fixes the initial state $x(t_0)$ at each sampling instant to the latest measured or estimated state.
- **Control Constraints** (2.1d) Impose bounds on the control inputs $u \in \mathbb{X}$, where $\mathbb{U}$ is commonly a closed "box" set:

$$\mathbb{U} \;=\; \big\{\, u \in \mathbb{R}^m \;\big|\; u_{min} \;\leq\; u \;\leq\; u_{max} \big\} \tag{2.3}$$

- **State Constraints** (2.1e)**:** Restrict the state of the system to a closed set $\mathbb{X}$, often specified by bounds:

$$\mathbb{X} \;=\; \big\{\, x \in \mathbb{R}^n \;\big|\; x_{min} \;\leq\; x \;\leq\; x_{max} \big\} \tag{2.4}$$

  State constraints may reflect physical limitations (e.g. temperature, pressure) or safety and quality requirements that must not be violated [19].
- **Terminal Constraint** (2.1f)**:** Forces the predicted state at time $t+N$ to lie within a terminal region Ω, promoting closed-loop stability since Ω is chosen to be invariant under a suitable control law [20].

For compactness, the OCP comprising Equations (2.1a)–(2.1f) is denoted by $P_N(x_0)$, highlighting its dependence on the current state $x_0 = x(t)$. The terminal constraint (2.1f) completes the problem formulation by guaranteeing a stabilizing invariant region. Additional constraints specific to certain applications may be integrated as needed, and will be discussed later in this work.

2.2 Model predictive Control in the context of optimal control

Modern optimal control theory is based on Pontryagin's minimum principle and the Hamilton-Jacobi-Bellman equation (HJB), which yield analytical solutions for OCPs in certain settings. For linear systems with quadratic cost (LQ problems), one can derive feedback laws valid for an infinite prediction horizon [18]. However, when dealing with non-linear dynamics, the HJB equation becomes intractable, and one resorts to solving a receding horizon open loop OCP repeatedly in real time, each time initialized at the newly measured state [11]. Direct numerical methods are then employed to translate the infinite-dimensional OCP into a finite-dimensional nonlinear program (NLP). A common discretization technique, also used in this work, is orthogonal collocation, which approximates $x_{[0:N]}$ while keeping $u_{[0:N-1]}$ constant in the different intervals, resulting in an NLP that can be solved with standard optimizers [21].

2.3 Robust Model Predictive Control

While nominal MPC provides a powerful framework for constraint handling and satisfactory performance under ideal conditions, it assumes a perfect model without disturbances or uncertainties. In real-world applications, however, system parameters are often subject to uncertainty due to factors such as external disturbances, measurement noise, or parametric variations. To address these challenges, Robust Model Predictive Control (RMPC) extends nominal MPC by explicitly accounting for bounded uncertainties in the prediction model.

2.3.1 RMPC as an extension of nominal MPC

Thus far, nominal MPC has been described, which assumes that the model perfectly matches the real system, and thus uncertainties are not considered. Although nominal MPC is valid if the model is exact and disturbances are absent, real-world systems often exhibit uncertainties, such as additive disturbances, parametric inaccuracies, or measurement noise. To address these challenges, RMPC explicitly accounts for uncertainties in the prediction model [12],[17],[19].

A typical continuous-time system representation for RMPC includes uncertainty parameters (p):

$$\dot{x} = f(x, u, p) \tag{2.5}$$

Here, $x \in \mathbb{R}^n$ are the system states, $u \in \mathbb{R}^m$ the control inputs, and $p \in \mathbb{R}^q$ captures uncertainty like parameter variations or external additive disturbances. The function

$$f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q \longrightarrow \mathbb{R}^n \tag{2.6}$$

maps $(x,\ u,\ p)$ to the state derivative $\dot{x}$. The uncertainties are assumed to lie within a known compact set $\mathbb{P}$, typically expressed as

$$\mathbb{P} \ = \ \left\{ p \in \mathbb{R}^q \ \middle|\ p_{\min} \ \leq \ p \ \leq \ p_{\max} \right\}, \tag{2.7}$$

capturing, for example, bounded parametric deviations [17].

In RMPC formulations, one designs control laws that ensure feasibility and stability different realizations of $p \in \mathbb{P}$. This can lead to different RMPC formulations. Min–max approaches for example consider worst considerations of p, which guarantee that state and control constraints are satisfied despite worst-case disturbances. A detailed treatment of important RMPC strategies will follow in Chapter 4.

2.3.2 Open-loop and closed-loop RMPC

RMPC strategies can be broadly classified into open-loop and closed-loop approaches. Open-loop RMPC minimizes a cost function with respect to a sequence of robust yet conservative control inputs $u_{[0:N-1]}$, specifically designed to counteract the worst-case parameter scenario as shown in (2.8). In comparison to the nominal MPC described in (2.1a)-(2.1f) the costs and constraints are adjusted according to the worst case realizations of p.

$$\min_{u_{[0:N-1]}} \max_{p_{[0:N]}} J(x(t), u_{[0:N-1]}, p_{[0:N]}, N) \tag{2.8}$$

Although this enables robustness, it often results in suboptimal performance as only one realization of p throughout N is considered. Thus, the real system operates may too far from the real bounds [19], [22].

Closed-loop RMPC addresses this limitation by accounting for multiple possible parameter realizations at various prediction steps, rather than focusing solely on the single worst-case scenario. It incorporates feedback strategies in its predictions that enable adaptive control inputs based on future state measurements or estimations. This adaptability, known as recourse, is the primary distinction between open- and closed-loop RMPC [17] [22]. As a result, closed-loop RMPC produces trajectories characterized by state sets($\mathcal{X}_{[0:N]}$) rather than discrete states, as shown in (2.9) . In each prediction step, the true state of the system is within the predicted set($\mathcal{X}_i$) [12].

$$\mathcal{X}_{[0:N]} = \{\mathcal{X}_0, \ldots, \mathcal{X}_N\} \tag{2.9}$$

In closed-loop RMPC, feedback is explicitly integrated into predictions through control policies, which consist of time-dependent control laws as shown in (2.10).

$$\mu_{[0:N-1]} = \{\mu_0(\cdot), \ldots, \mu_{N-1}(\cdot)\} \tag{2.10}$$

These control laws map possible future states directly to control inputs, that is, by explicitly responding to future uncertainty realizations [12], [19]. Without restrictions, the resulting robust optimal control problem (ROCP) becomes infinite-dimensional. To ensure a tractable finite-dimensional solution, the structure of control laws is typically predefined to belong to a certain class $\mathcal{M}(x)$, for example affine control laws $\mathcal{M}(x) = \{(\mathbf{K}, \mathbf{v}) \in \mathbb{R}^{n_u \times n_x} \times \mathbb{R}^{n_u} \mid \boldsymbol{\mu} = \mathbf{K}\mathbf{x} + \mathbf{v}\}$, leading to the generalized robust optimal control formulation.

$$P_{ROC,N}(x_0) = \min_{\mu_{[0:N-1]} \in \mathcal{M}(x)} J_N(x, \mu_{[0:N-1]}) \tag{2.11a}$$

subject to (s.t.):

$$\mathcal{X}_{i+1} = g(\mathcal{X}_i, \mu, p) \quad \text{(reachable set-propagation)} \tag{2.11b}$$

$$\mu_{[0:N-1]} \in \mathbb{U},\ \tau \in [t,\, t+N-1] \quad \text{(control constraints)} \tag{2.11c}$$

$$\mathcal{X}_i \in \mathbb{X},\ \tau \in [t,\, t+N] \quad \text{(constraints for reachable sets)} \tag{2.11d}$$

$$x(t_0) = x_0 \quad \text{(initial constraint)} \tag{2.11e}$$

Here $\mathcal{M}$ in (2.11a), denotes the set of admissible control policies [19].

2.4 Recursive Feasibility of an Optimization Problem

MPC involves iterative resolution of an optimal control OCP, repeatedly initialized at each time step. Recursive feasibility refers to the property that ensures the consistent solvability of a recurring OCP while being compliant with all constraints. Provided that the initial OCP

$$P_N(x_0) \tag{2.12}$$

is solvable, subsequent optimizations initiated from future states remain consistently feasible [23]. In the subsequent time step, a new feasible, but not necessarily optimal, input trajectory can be constructed from the old input trajectory which is shifted one timestep ahead and then a terminal set poses a feasible input for the last prediction step. In general, terminal constraints as in (2.13) are an important ingredient in proving recursive feasibility [23], [19].

$$x(N) \in \mathbb{X}_f \tag{2.13}$$

The terminal constraint ensures the indefinite feasibility of the ROCP, as once entered $\mathbb{X}_f$ the system can remain there, as there are always control inputs that make the system stay in $\mathbb{X}_f$. In Figure 2.2 an example trajectory starting in $x(0)$ and ending in $\mathbb{X}_f$ visualizes the effect on $x(N)$ because it must end in $\mathbb{X}_f$ to ensure recursive feasibility. Determining such a terminal set that guarantees an invariant control law is difficult for nonlinear systems.

Due to existing uncertainties, recursive feasibility is lost for nominal MPC since the optimal solutions from the old trajectory do not need to be feasible control inputs for

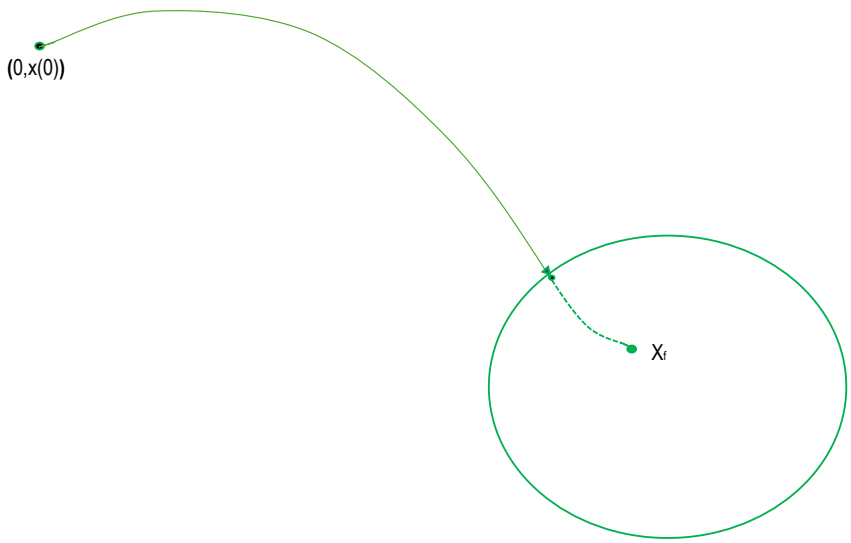

Figure 2.2: Illustration of an example trajectory and $\mathbb{X}_f$ [11]

the new trajectory. Due to uncertainty realizations not considered in nominal MPC, the input solution from the previous trajectory could lead to constraint violations. Thus, guaranteeing recursive feasibility in the robust case for a robust optimal control problem is more challenging. Instead of just one nominal trajectory of future states the whole reachable set of all possible future realizations of the states due to the uncertainty needs to be considered. For the ROCP, a terminal set constraint ensures the existence of a feasible input at the end of the prediction horizon. However, for the robust terminal set, there needs to exist an input, ensuring it's invariance under all uncertainties. In related works like [14] and [15] the terminal constraint ensures that $x(N)$ lies within an invariant terminal set , ensuring that once the system enters this set, it remains indefinitely within this safe region. Together with the bounding of the reachable sets the terminal constraint ensures the indefinite feasibility of the ROCP, as once entered $\mathbb{X}_f$ the system can remain there, as there are always control inputs that make the system stay in $\mathbb{X}_f$ [23].

Reachable sets are another key concept used throughout this work, particularly to demonstrate recursive feasibility. Therefore, the following section provides a more detailed introduction to reachable sets and outlines their fundamental principles.

2.5 Computation of Reachable Sets

As previously discussed, closed-loop RMPC generates state-set trajectories rather than discrete state trajectories due to inherent parameter uncertainties [12]. Starting from x_0, the sequences of reachable state sets are computed at each subsequent time step up to N, considering all realization of p within $\mathbb{P}$. Consequently, the sequence of reachable sets in (2.9) forms an envelope that encompasses all potential trajectories of the uncertain system, providing a bounded region within which the true system can evolve [12], [16]. Figure 2.3 provides a schematic illustration of the concept of

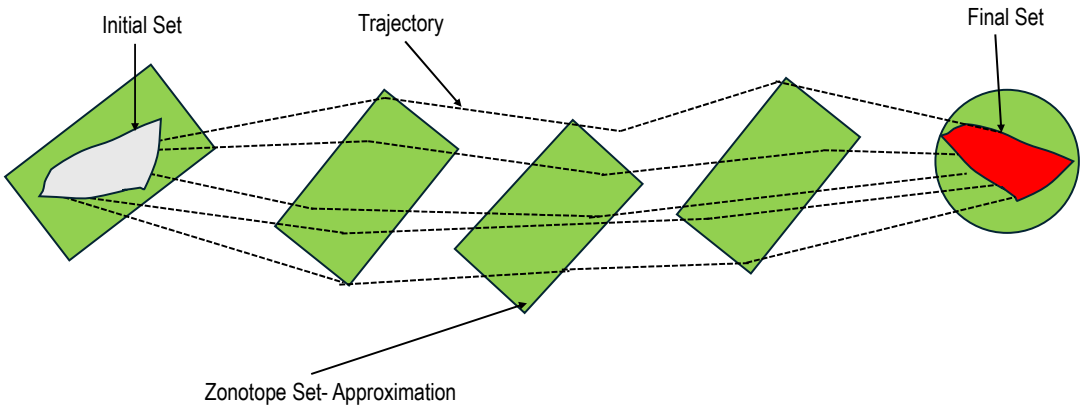

Figure 2.3: Schematic representation of rechable sets in a trajectory starting from an initial set [24]

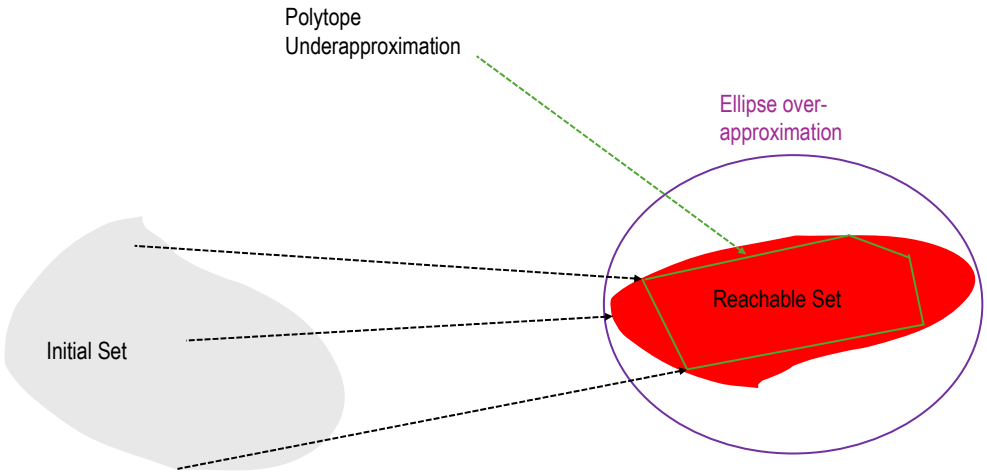

Figure 2.4: Illustration of different approximation schemes for the reachable set in red [25]

reachable sets. Unlike a scenario involving a single discrete initial state, the figure presents trajectories originating from an initial state, effectively capturing a range of possible system evolutions up to the final reachable set. Reachable sets can generally be efficiently approximated using geometric representations, such as zonotopes [24]. Closed-loop RMPC tends to produce tighter reachable sets due to recourse, keeping the trajectories closer to each other [16], [19]. The recursive process for computing reachable sets is mathematically expressed in (2.14)

$$\mathcal{X}_i(x_0, \mathcal{U}_{i-1}, \mathbb{P}) = g(\mathcal{X}_{i-1}(x_0, \mathcal{U}_{i-2}, \mathbb{P})) \tag{2.14}$$

This recursive definition clearly conveys that the reachable set at step i,$\mathbb{X}_i$, is generated directly from the reachable set in the previous prediction step $i-1$, making $\mathbb{X}_i$ the one-step reachable set from $\mathbb{X}_{i-1}$. Thus, the set $\mathbb{X}_i$ can also be interpreted as the i-step reachable set originating from the initial state x_0 [16], [25]. Calculating precisely reachable sets, particularly for non-linear systems, is generally challenging or computationally infeasible [16], [25]. Consequently, geometric approximations, such as ellipsoids and polytopes, represent practical approaches to achieve a balance between computational efficiency and acceptable accuracy [26]. Figure 2.4 further visualizes different approximation approaches for reachable sets. An initial set in gray is depicted, from which several trajectories propagate to a subsequent reachable set shown in red, with trajectories illustrated schematically via arrows [25]. Common approximation techniques for reachable sets, such as ellipsoidal overapproximations and polytopic underapproximations, can be seen. As already mentioned, there are different approaches to approximate reachable sets. An *underapproximation* of a reachable set contains nearly all - but not necessarily all - of the states that are truly reachable. This can be useful in case there should be no false positive states in the reachable set. An *overapproximation*, on the other hand, fully contains the actual reachable set, including all genuinely reachable states, but may also include boundary points that the system cannot actually reach. This conservatism is inherent to overapproximation.

When reachable sets are over-approximated iteratively over multiple time steps, starting from an initial set, the conservatism accumulates progressively. This phenomenon is referred to as the *wrapping effect*. Thus, overapproximations should be made as tight as possible to minimize conservatism and to closely bound the true reachable set.

Figure 2.5 illustrates the concept of over-or underapproximationg reachable sets in a two-dimensional example. The blue box denotes an initial set of states. From this set, several trajectories propagate into the actual one-step reachable set, depicted in purple. Using multidimensional hyperintervals, different overapproximations can be constructed. A minimal hyperinterval that tightly encloses the reachable set is shown in red, while a slightly larger and more conservative overapproximation is also indicated in green.

An interval overapproximation $[\underline{R}, \overline{R}] \subseteq \mathbb{R}^n$ is considered a *tight overapproximation* of a set $\mathbb{X} \subseteq \mathbb{R}^n$ if, for all other interval overapproximations $[\underline{a}, \overline{a}] \subseteq \mathbb{R}^n$, the following condition holds [25]:

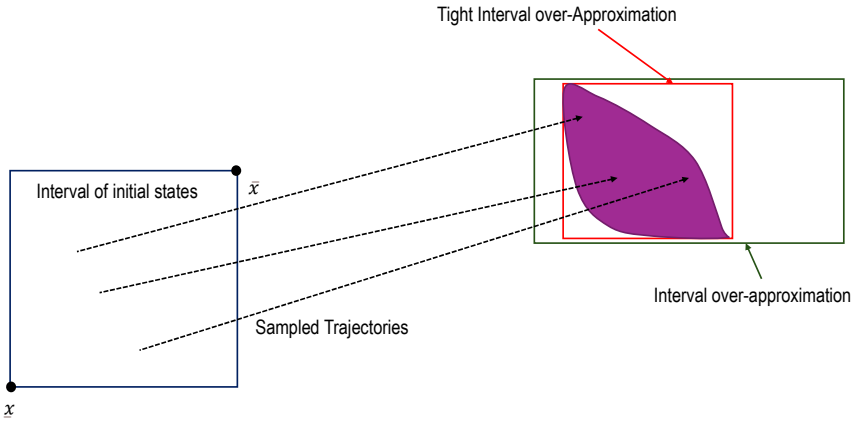

Figure 2.5: Illustration of a tight hyperrectangular overapproximation in red and a non-tight over-approximation of the reachable set [25]

$$\mathbb{X} \subseteq [\underline{a}, \overline{a}] \Rightarrow [\underline{R}, \overline{R}] \subseteq [\underline{a}, \overline{a}] \tag{2.15}$$

In the subsequent sections, it will be discussed how monotonicity can be leveraged to simplify the computation of such hyperrectangular reachable set sets.

2.5.1 Key concept: Monotonicity

Monotonicity plays a central role in the computation of reachable sets in the RMPC approaches investigated in this work. For this reason, monotonicity is briefly introduced here.

Most definitions and equations are presented for the discrete-time case. Though they can be extended to continuous-time systems, as ODE and DAE systems can be transformed into finite dimensional discrete systems with full discretization approaches like orthogonal collocation [21], [27].

A dynamical system F, as defined by

$$F : \mathbb{X} \times \mathbb{U} \times \mathbb{P} \to \mathbb{X}, \quad x^+ = F(x, u, p) \tag{2.16}$$

is said to be *positively monotone* or *cooperative* with respect to x if

$$F(x_1, u, p) \geq F(x_2, u, p) \quad \forall u \in \mathbb{U}, \forall p \in \mathbb{P}, \tag{2.17}$$

for all $x_1, x_2 \in \mathbb{X}$ such that $x_1 \geq x_2$ element-wise.

If instead

$$F(x_1, u, p) \leq F(x_2, u, p) \quad \forall u \in \mathbb{U}, \forall p \in \mathbb{P}, \tag{2.18}$$

then F is said to be *negatively monotone* or *competitive*. Strict inequalities in (2.17) or (2.18) along with $x_1 > x_2$ define *strict monotonicity*.

Analogously, monotonicity can be defined to be positive monotone with respect to p like in (2.19) or negative monotone with respect to p like in (2.20):

$$F(x,u,p_1) \geq F(x,u,p_2) \quad \forall x \in \mathbb{X}, \forall u \in \mathbb{U} \quad \text{if } p_1 \geq p_2, \tag{2.19}$$

$$F(x,u,p_1) \leq F(x,u,p_2) \quad \forall x \in \mathbb{X}, \forall u \in \mathbb{U} \quad \text{if } p_1 \geq p_2. \tag{2.20}$$

Unless otherwise stated, this work refers to positive monotonicity [14], [28].

Figure 2.6, adapted from [29], illustrates the behavior of system trajectories under varying parameter values p. The trajectories are bounded above and below by evaluations at the maximum and minimum values of p, respectively, illustrated by the red or blue arrow. In 2.6 the system response increases with increasing values of p and decreases with decreasing values of$p$ [29], [30].

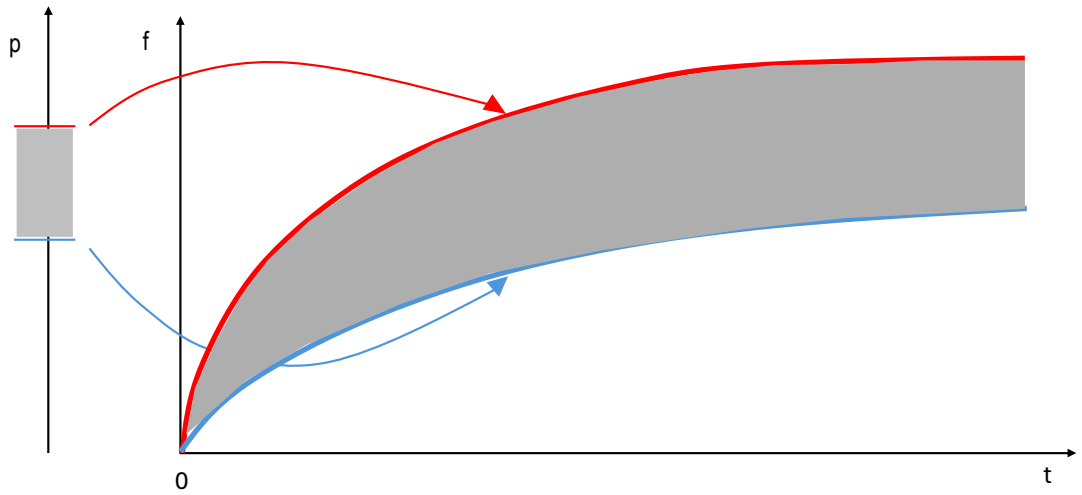

Figure 2.6: Systemtrajectories of a monotone system that are bounded by their parameter values (p) [29]

Monotonicity allows reachable sets to be computed more efficiently than traditional approximation methods. By evaluating the system function only at the extreme corners of a state and parameter interval, tight overapproximations in the form of multi-dimensional intervals, termed hyperrectangles, can be constructed [25], [13]. Figure 2.7 illustrates this principle in 2D, where a tight interval overapproximation, drawn in purple, is formed from only two evaluations of F at the bounds of the initial state and the parameter intervals.

Let x and p be bounded componentwise by $\underline{x}, \overline{x}$ and $\underline{p}, \overline{p}$ as defined in (2.4) and (2.7). Then, for a fixed input u, the image of F lies within the interval

$$F(x,u,p) \in [F(\underline{x},u,\underline{p}), F(\overline{x},u,\overline{p})] \equiv IR \quad \forall u \in \mathbb{U}.$$

If F is monotone, IR is the tightest possible hyperrectangle [9], [25].

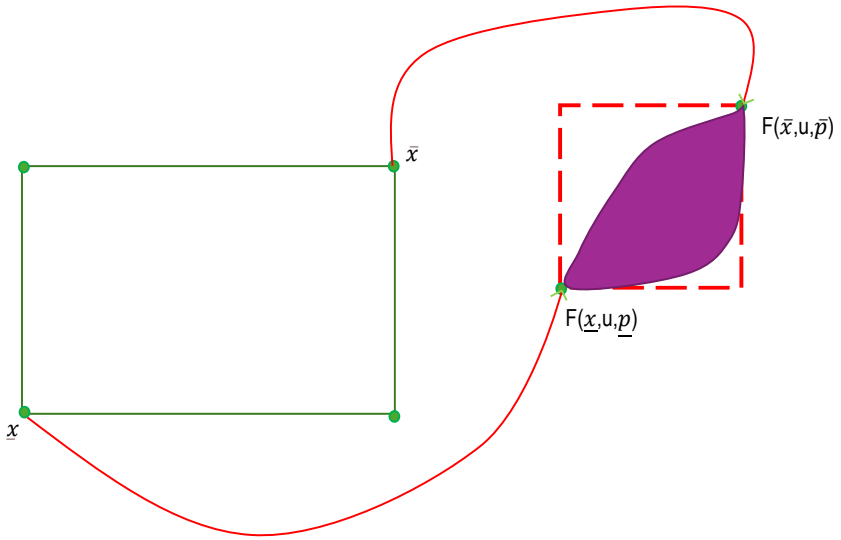

Figure 2.7: Illustration of a tight hyperrectangular overapproximation with a decompositionfunction in red the reachable set in purple [25]

2.5.2 Key-concept: Mixed-Monotonicity

When the system dynamics F are not monotone as defined in (2.17)–(2.20), the concept of *mixed-monotonicity* can be employed, which generalizes monotonicity [9]. A *decomposition function* (d) splits F into increasing and decreasing components.d, which itself is positively monotone in its first arguments and negative monotone in its last arguments, maps from doubled input dimensions and satisfies

$$d : \mathbb{X} \times \mathbb{P} \times \mathbb{U} \times \mathbb{X} \times \mathbb{P} \to \mathbb{X}. \tag{2.21}$$

F is mixed monotone with respect to d if the following hold:

$$F(x, u, p) = d(x, p, u, x, p) \quad \forall x \in \mathbb{X}, \forall p \in \mathbb{P}, \tag{2.22}$$

$$d(x_1, p_1, u, x_2, p_2) \leq d(x_3, p_3, u, x_2, p_2) \quad \text{for } x_1 \leq x_3, p_1 \leq p_3, \tag{2.23}$$

$$d(x_1, p_1, u, x_4, p_4) \leq d(x_1, p_1, u, x_2, p_2) \quad \text{for } x_2 \leq x_4, p_2 \leq p_4. \tag{2.24}$$

These conditions define the cooperative and competitive dynamics within d. A continuous-time analogue is provided in [9]. In the following it is explained how reachable sets for monotone and mixed-monotone systems can be computed, and in this context a possible method for the computation of d is given too.

When F is not monotone, mixed monotonicity still enables the computation of hyperrectangular overapproximations. Analogous to (2.5.1), a mixed-monotone system satisfies:

$$F(x, u, p) \in [d(\underline{x}, \underline{p}, u, \overline{x}, \overline{p}), d(\overline{x}, \overline{p}, u, \underline{x}, \underline{p})] \equiv IR \quad \forall u \in \mathbb{U}.$$

Unlike in the monotone case, IR is not guaranteed to be the tightest possible IR. The conservatism of the overapproximation depends heavily on the choice of d [9], [25]. For any system F as defined in (2.16), if monotone or not, there is a valid decomposition

function d that can be formulated as the solution of an optimization problem shown in (2.25). For discrete-time systems, the component d_j of d can be defined by [9]:

$$d_j\big(x_1, p_1, u, x_2, p_2\big) \;=\; \begin{cases} \min\limits_{\substack{\tilde{x}\in[x_1,x_2]\\ \tilde{p}\in[p_1,p_2]}} F_j\big(\tilde{x},\, u,\, \tilde{p}\big), & \text{if } \begin{bmatrix} x_1 \\ p_1 \end{bmatrix} \;\leq\; \begin{bmatrix} x_2 \\ p_2 \end{bmatrix}, \\ \max\limits_{\substack{\tilde{x}\in[x_2,x_1]\\ \tilde{p}\in[p_2,p_1]}} F_j\big(\tilde{x},\, u,\, \tilde{p}\big), & \text{if } \begin{bmatrix} x_2 \\ p_2 \end{bmatrix} \;\leq\; \begin{bmatrix} x_1 \\ p_1 \end{bmatrix}. \end{cases} \tag{2.25}$$

Although this provides a theoretical basis for constructing the tightest possible d, it is often computationally impractical for real-time RMPC because the optimization problem is not solvable or d is piecewise defined and scales badly with the dimensions of x. Therefore, compromises are made to ensure that d remains differentiable and computationally efficient [9], [15], [16]. In this work, interval arithmetics is used to construct d for a non-linear, non-monotone continuous system model, following the approaches of [16] and [15]. Interval arithmetics enable the construction of a decomposition function d that is differentiable and potentially applicable within an real RMPC loop. The individual output components d_j are obtained by relaxing the theoretical formulation of Equation (2.25) using interval arithmetics. This relaxation yields a function d consisting solely of min-max expressions, under the assumption that the corresponding system function F_j can be decomposed into separable terms. As a result, the obtained d can be conservative, but is therefore differentiable and tractable.

The procedure for obtaining one possible formulation of with interval arithmetics$d_j^{\min}$ with interval arithmetics is illustrated in (2.26). A similar formulation using max-operators instead of min-operators can be used to derive $d_j^{\max}$. $d_j^{\max}$ and $d_j^{\min}$ are both decompositonfunction that can used together to bound the overapproximate the reachable set as rectangle [15].

$$\begin{aligned} \tilde{d}_j(x_1, p_1, u, x_2, p_2) &= \min_{\tilde{x}\in[x_1,x_2],\, \tilde{p}\in[p_1,p_2]} F_j^a(\tilde{x}, u, \tilde{p}) + \min_{\tilde{x}\in[x_1,x_2],\, \tilde{p}\in[p_1,p_2]} F_j^b(\tilde{x}, u, \tilde{p}) \\ &\leq \min_{\tilde{x}\in[x_1,x_2],\, \tilde{p}\in[p_1,p_2]} \Big[F_j^a(\tilde{x}, u, \tilde{p}) + F_j^b(\tilde{x}, u, \tilde{p})\Big] = d_j(x_1, p_1, u, x_2, p_2) \end{aligned} \tag{2.26}$$

2.6 Common Feedback-Strategies in RMPC

The essential control-theoretical foundations have already been introduced in the preceding chapters. Based on that, this section presents key closed-loop RMPC strategies. Specifically, it introduces Multi-Stage RMPC and Tube-based RMPC as foundational methods, before discussing set-partitioning-based RMPC approaches, which form the core of this thesis.

2.6.1 Multi-stage MPC

In Multi-Stage MPC, the evolution of uncertain parameters p over a prediction horizon N is modeled using a scenario tree, as illustrated in Figure 2.8. The root node x_0^1 represents the state of the system currently measured or estimated. The branches of the tree represent different possible values that p may assume in each prediction step, reflecting multiple possible evolutions of the state of the system x. Thus, each node in the tree corresponds to a possible state trajectory resulting from a specific realization of p, while the whole path from the root x_0^1 to a leaf node x_N represents one scenario. In this context, the superscript in x, p, u denotes the overall scenario, while

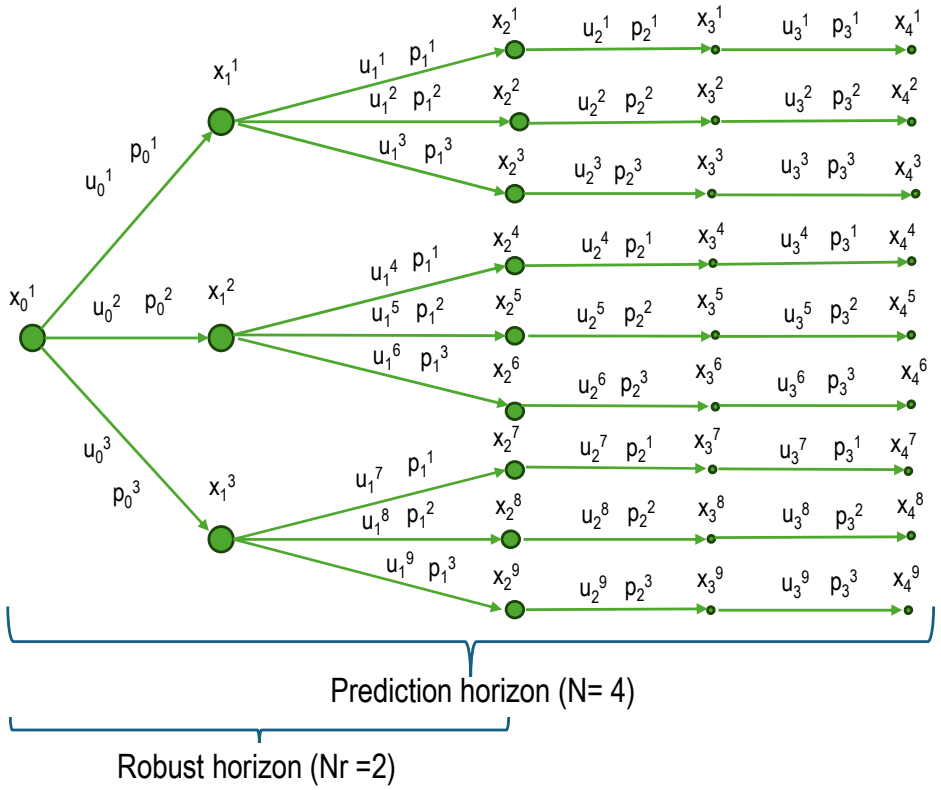

Figure 2.8: Schematic representation of a scenario tree for three parametric uncertainties, N =4 and N_r = 2 [18]

the subscript refers to the prediction step. The branching structure models combina-

tions of extreme values for p, assuming that it is a bounded but unknown parameter that may vary at each time step. Although these variations may not occur in reality, the multistage MPC proactively considers them by adapting control inputs u for each branch [31], [18].

This branching enables the incorporation of *recourse*: for each node in the scenario tree and at each prediction step, a corresponding control input is calculated that accounts for possible future realizations of p. The structure of the scenario tree reflects the fact that future measurements or estimates of x will become available, allowing the controller to respond appropriately. However, it should be noted that at a given node, all outgoing inputs u must be equal, since the future is unknown and cannot be anticipated. Thus, *nonanticipativity* is enforced [6], [18].

Although Figure 2.8 shows a prediction horizon of $N = 4$, branching is typically limited to a *robust horizon* N_r. This is a practical trade-off between robustness, performance, and computational tractability. The underlying ROCP scales exponentially with N_r and the dimensionality of p [18]. Therefore, beyond N_r, it is commonly assumed that the parameter p remains constant, corresponding to the last branch in the scenario. The goal is to accurately predict the near future, as the closed-loop MPC controller re-initializes at each time step by measuring or estimating x_0^1 and solving the ROCP again to generate a new scenario tree [18], [6]. A general form of the Multi-stage ROCP is given in (2.27)–(2.30). Here $\mathcal{I}$ represents an interval that lists all scenarios.

$$\min_{x,u} \sum \omega_i J_i(X_i, U_i) \tag{2.27}$$

subject to:

$$x^{j,k+1} = f(x^{j,k}, u^{j,k}, p^{j,k}) \qquad \forall (j, k+1) \in \mathcal{I} \tag{2.28}$$

$$g(x^{j,k}, u^{j,k}) \leq 0 \tag{2.29}$$

$$u^{j,k} = u^{j',k} \qquad \text{if } x^{j,k} = x^{j',k} \tag{2.30}$$

If Equation (2.30) was replaced by a constraint that enforces all inputs in the same prediction step to be equal, the recourse property would be lost, no feedback would be included in the predictions and the overall performance would suffer [18]. If p were exactly captured by the tree structure of the scenario, then Multi-stage MPC would the best possible performance in the presence of p, since the optimal inputs u can be calculated for all scenarios up to the horizon N [18], [22].

2.6.2 Tube-based MPC

Although Multi-Stage MPC can provide optimal performance, it has considerable limitations, most notably, its poor scalability and high online computational cost. Tube-based MPC offers a viable alternative with runtime complexity similar to that of nominal MPC. In tube-based MPC, a nominal OCP is solved repeatedly using tightened constraints with regards to x and u. Typically, two controllers are employed:

- A *nominal controller* computes the nominal state and the input trajectories $\tilde{x}_{0:N}, \tilde{u}_{0:N}$ based on the tightened constraints.
- An *ancillary controller* ensures that the actual system follows the nominal trajectory.

The ancillary controller, combined with the tightened constraints, ensures that the true trajectories $x_{[0:N]}$ remain within a bounded uncertainty set ($\mathbb{S}$), which can be imagined as a tube around the nominal state $\tilde{x}$. If $\mathbb{S}$ is represented by a multidimensional hyperrectangle, this structure is illustrated in Figure 2.9. The blue trajectory represents the nominal one, while the green areas show the $\mathbb{S}_i$ to different times.

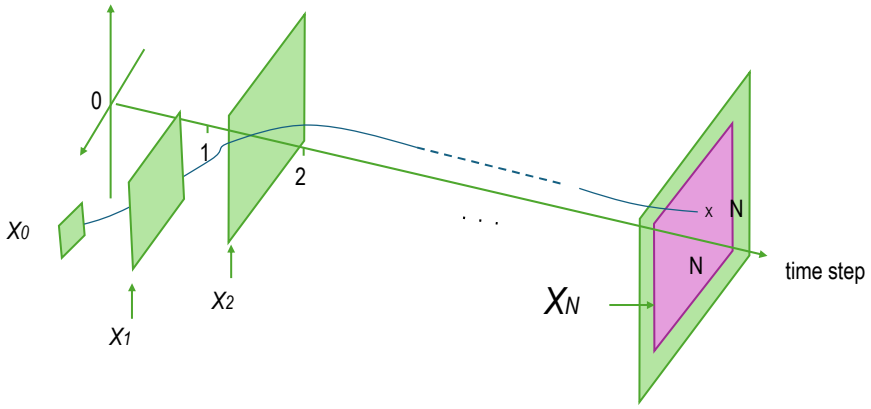

Figure 2.9: Scheme of tube-based MPC with low-complexity hyperrectangular sets in green and nominal trajectory in blue [17]

Tube-based MPC is illustrated in (2.31)–(2.33) using a linear, discrete-time system with additive disturbances. The ancillary controller introduces recourse into the MPC framework via state feedback and aims to make the actual trajectories follow the nominal ones as close as possible. This is achieved using a control law of the form

$$\mu_i(x) = u + K(x - \tilde{x}) \tag{2.31}$$

where K is a stabilizing feedback gain [19].

For the system, the true states x evolve within a set around the nominal system states $\tilde{x}$:

$$x_{i+1} = Ax_i + B\mu_i(x) + p = Ax_i + B(u + K(x - \tilde{x})) + p \tag{2.32}$$

$$\tilde{x}_{i+1} = A\tilde{x}_i + Bu \tag{2.33}$$

Each trajectory affected by uncertainty at time i x_i is enclosed within a bounded region $\mathbb{S}$ around $\tilde{x}_i$. $\mathbb{S}$ serves as an outer bound for all uncertainty-affected trajectories resulting from Equation (2.32). Starting from an x_0, reachable sets of system states over N prediction steps can be computed using this structure, forming the set trajectory

$$\mathcal{X}(i, x_0, u) = \{\tilde{x}_i\} + \mathbb{S} \tag{2.34}$$

Each such set is centered around the corresponding nominal $\tilde{x}_i$ and exhibits invariance with respect to the disturbance $p \in \mathbb{P}$. The union $\mathcal{X}_{[0N],x_0,u)}$ then bounds all possible system trajectories over N prediction steps. Based on the known disturbance tube $\mathbb{S}$, the state and input constraints can be tightened accordingly as:

$$\mathbb{X}_{tight} = \mathbb{X} \ominus \mathbb{S} \tag{2.35}$$

$$\mathbb{U}_{tight} = \mathbb{U} \ominus K\mathbb{S} \tag{2.36}$$

assuming that $\mathbb{S}$ is constant over N. [19].

In Figure 2.10, an example 2D nominal trajectory $\tilde{x}_{[0:N]}$ is shown as a solid line, the actual trajectory $\tilde{x}_{[0:N]}$ as a dashed line, and the sets $\mathcal{X}_{[0N],x_0,u)}$ are represented as circles centered around $\tilde{x}_i$, each enclosing the corresponding true x_i.

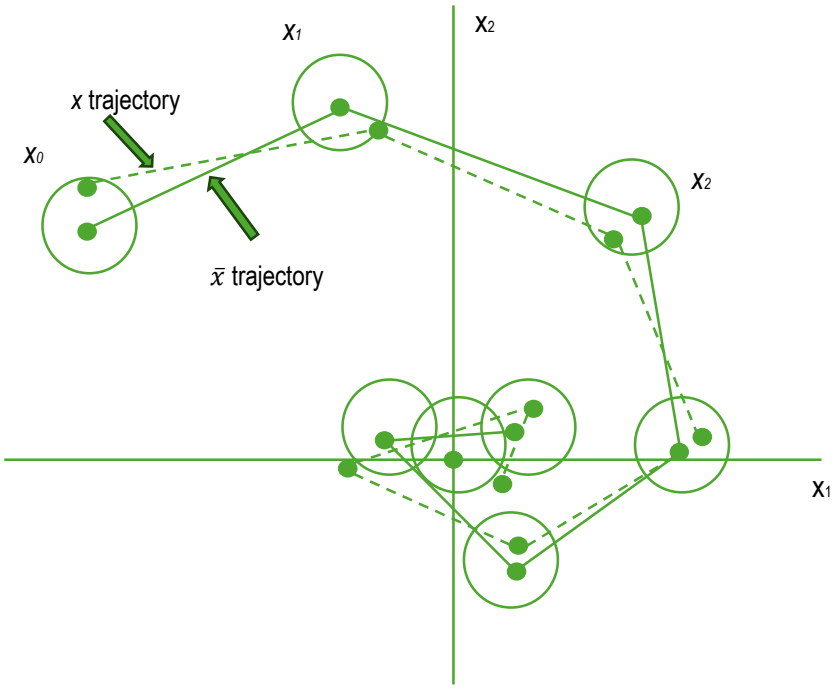

Figure 2.10: Scheme of tube-based MPC in 2D with circles as reachable sets [19]

For nonlinear systems, the ancillary controller takes a more complex form and the resulting ROCP becomes correspondingly more involved. Various Tube-based MPC formulations exist, differing in how the $\mathbb{S}$ and the ancillary controller are computed. In the non-linear case, determining $\mathbb{S}$ is non-trivial and is often approximated, though this can be done offline, enabling significant run-time savings.

Advantages of Tube-based MPC include:

- relatively simple implementation,
- guaranteed recursive feasibility and stability under uncertainty, and
- robust constraint satisfaction via set tightening [19]

The main drawback is its suboptimal control performance. Since the constraints on x and u are tightened, robustness is inherently traded off against performance [32].

Motivated by Multi-Stage MPC, which targets optimal robust control performance, and inspired by the low-complexity benefits of Tube MPC, the following sections introduce feedback-based set-partitioning strategies. These strategies, serving as the core contribution of this thesis, aim to combine strong robustness guarantees with computational tractability.

2.7 Set-partitioning-Schemes

Before introducing feedback-based set-partitioning strategies, the necessary set-partitioning schemes need to be discussed. Both aim to divide the reachable set, typically represented as a multidimensional hyperrectangle in $\mathbb{R}^n$, into multiple smaller sets termed subregions that allow for more granular and targeted control actions.

- **Grid-like partitioning:** In this approach, the dimensions of the state space to be partitioned are first selected. The hyperrectangle is then split orthogonally along these dimensions into a uniform grid structure. Each cell of this grid represents a subregion. Figure 2.11 shows on the left a two-dimensional example in which a hyperrectangle is partitioned once along the second dimension and twice along the first. This scheme is simple and systematic, but can be inflexible when varying resolution is desired across dimensions and subregions.
- **Search-tree-like partitioning:** In contrast, this method introduces hierarchical flexibility. The hyperrectangle is recursively partitioned, allowing each subregion to be partitioned independently on the basis of the level. The later a dimension is partitioned, the more independently the division in the resulting subregions can be chosen. The order in which the dimensions are partitioned and the number of partitions per dimension determine the structure of the resulting tree. As illustrated in Figure 2.11 on the right, the first dimension is split first, generating two subregions. Each of these can then be partitioned further in subsequent dimensions, for example, the left subregion is split twice along dimension two, while the right one is split only once.

This transition from grid-like to search-tree-like partitioning reflects a shift from uniformity to adaptability. Although the grid-based approach subdivides all regions uniformly regardless of their location or sensitivity, the search tree-like partitioning method gives the optimizer freedom to define partition boundaries and depths per

subregion. This makes it particularly suitable for robust feedback strategies, where certain directions or regions of the reachable set may exhibit greater uncertainty or dynamic sensitivity [14], [15].

Figure 2.11: Left grid-like set partitioning scheme and on the right search-tree-like set partitioning scheme [15]

Figures 2.12 and 2.13 present a three-dimensional hyperrectangle and its recursive partitioning process using a search tree-like partitioning scheme. The initial set $\mathcal{S}_0^0$ represents the initial set that is considered, for example a reachable set of system states of a rector cascade. As partitioning progresses, new subregions emerge $\mathcal{S}_j^t$, where the lower index j refers to the tree level, which is the dimension, and the upper index t indicates the subregion at that level. For example, $\mathcal{S}_1^1$ corresponds to the second subregion after the first dimension is split. In general, the ordering of the dimensions can also be arbitrarily chosen such that the levels in Figure 2.13 could be arranged in a different sequence.

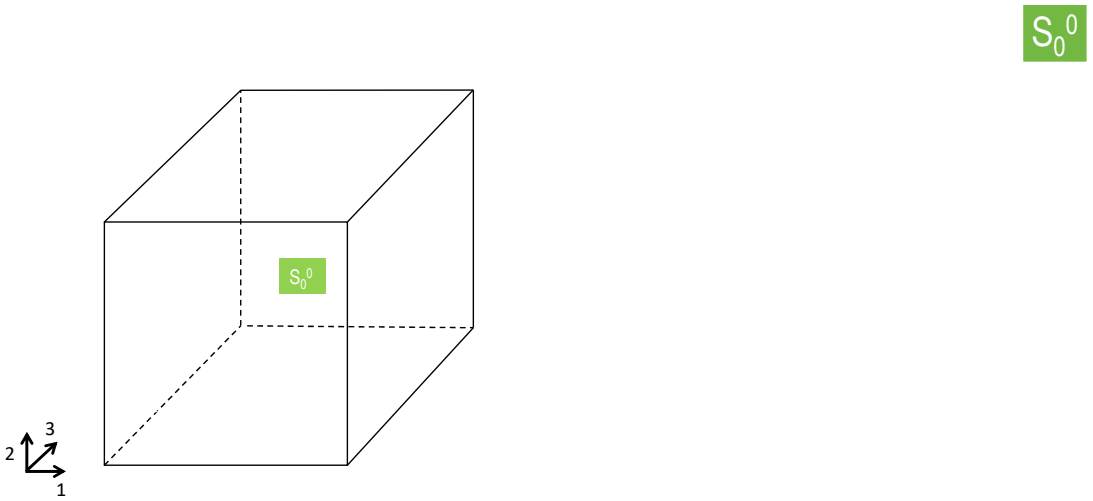

Figure 2.12: 3D hyperrectangle reprsenting initial set $\mathcal{S}_0^0$ [15]

The total number of resulting subregions N_s can be calculated using (2.37).

$$N_s = \prod_{j=1}^{n_x} (n_{c,j} + 1) \tag{2.37}$$

where n_c, j is the number of partitions along dimension j, and n_x is the total number of state dimensions. This formula assumes the same number of subregions in each branch in a respective level of the tree, as is the case in Figure 2.13.

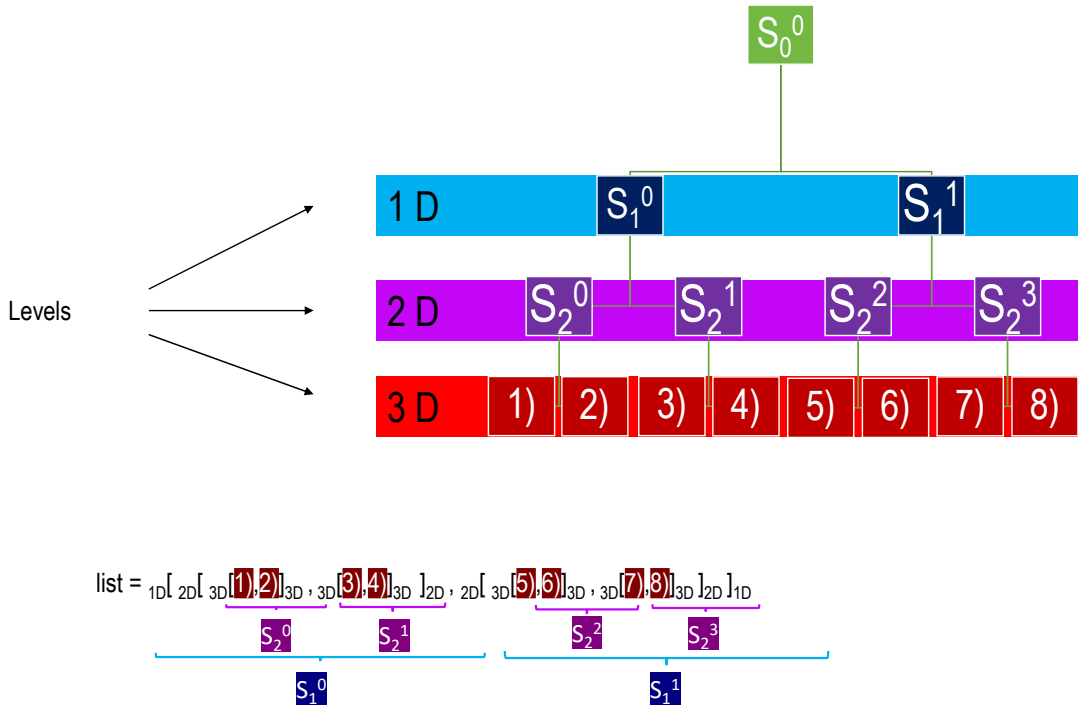

Figure 2.13: Search-tree-like partitioning scheme for hyperrectangle in 2.12 and representation of partitioning scheme as nested list [15]

To make this recursive structure easier to implement and visualize, the partitioning tree can also be represented as a *nested list*. Figure 2.13 shows this representation. The brackets denote the depth of the tree and each element corresponds to a subregion at a given level. The deeper the nesting, the later the corresponding dimension is split. The elements in the innermost brackets represent the final subregions. The evolution of the full tree is illustrated in Figure 2.14. Each subregion is color-coded according to the dimension in which it was partitioned.

Further implementation details are provided in the appendix in A.1. These include:

- step-by-step visualizations of how the search-tree-like partitioning scheme unfolds across dimensions.
- the corresponding nested list representations for each stage,
- the fully constructed partition tree,
- and pseudocode describing how subregions are aligned and indexed throughout the tree-building process in Algorithm 1 and 2.

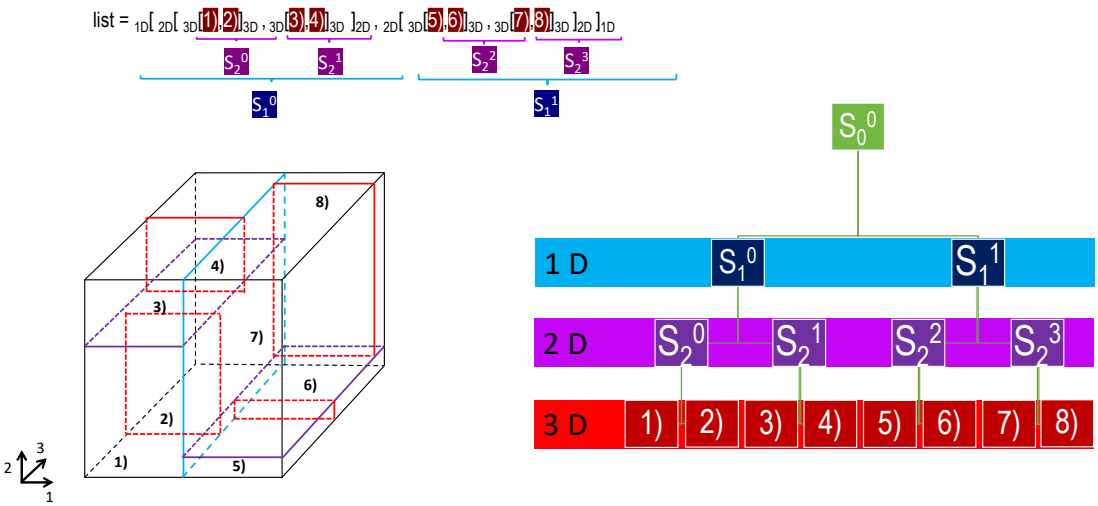

Figure 2.14: Fully partitioned hyperrectangle from 2.12 with regards to search-tree-like partitioning scheme [15]

2.8 Feedback-based set-partitioning MPC

Building on the structure of search-tree-like partitioning and leveraging monotonicity or mixed-monotonicity, three feedback-based set-partitioning MPC strategies are introduced. These approaches incorporate feedback into the prediction stage by explicitly assigning constant control actions to subregions of reachable sets. This introduces recourse into the RMPC-approaches.

An analogy to Tube MPC lies in the generation of a trajectory of reachable sets as described in (2.9). Each reachable set $\mathcal{X}_i$ is partitioned into N_s subregions according to the search-tree-like partitioning scheme and each subregion receives its own constant control input u^s, enabling the representation of distinct uncertainty scenarios across state-space dimensions. This principle underlies feedback-based set-partitioning MPC and resembles the idea of multi-stage MPC.

The major advantage of RMPC with feedback-based set-partitioning lies in the linear scaling of the complexity of the ROCP with respect to N and N_s, in contrast to the exponential growth of complexity of multi-stage MPC with N_r. This efficiency comes from overapproximating the propagated subregions by a hyperrectangle $\mathcal{X}_{i+1}$, which is then again partitioned.

The bounding principle of feedback-based set-partitioning MPC is illustrated in Figure 2.15 for a two-dimensional reachable set. The propagated subregions in red are enclosed by a bounding hyperrectangle in black, which is again partitioned. This bounding step is repeated across the N prediction steps. Despite the advantages in scala-

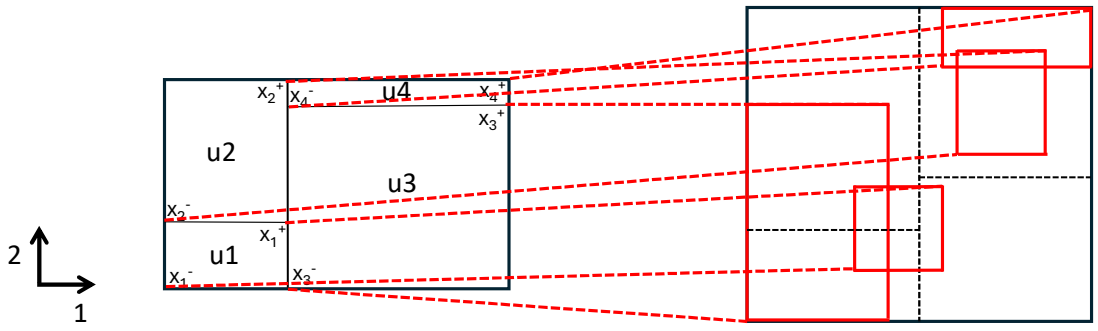

Figure 2.15: Bounding principle of feedback-based set-partitioning MPC with regards to search-tree-like partitioning scheme [15]

bility and robustness for complex systems, the method introduces conservativeness inherently. The $\mathcal{X}_i$ do not represent the exact reachable sets, as they merely encapsulate a superset of possible states [14], [15]. As shown in Figure 2.15, the red subregions propagated do not fill the entire volume of the bounding hyperrectangle.

The propagation and constraint formulation for $\mathcal{X}_i$ relies solely on the lower left corner, representing minimum values of x, p, and the upper right corner, representing maximum values of x, p, due to the hyperrectangle structure. This simplification is valid since p is treated as an additive or parametric disturbance bounded by the compact set $\mathbb{P}$ as defined in (2.7). Similarly, x and u are bounded by the closed sets $\mathbb{X}$ and $\mathbb{U}$

defined in (2.4) and (2.3). These bounds will be assumed from here on and not stated again explicitly.

With monotonicity or mixed-monotonicity, the reachable sets $\mathcal{X}_i$ can be efficiently overapproximated as multidimensional hyperrectangles. In the monotone case, these approximations are even tight [13], while in the non-monotone case, their quality depends on the decomposition function d, and a wrapping effect may occur [25]. The hyperrectangular-shaped set trajectory $\mathcal{X}_{[0:N]}$ thus defines, similarly to tube-based MPC, a sequence of sets along which the actual system evolves from a given initial state $x(0)$. Figure 2.9 gives a visual analogy.

The MPC set-partitioning strategies based on the feedback that follow are based on the works [14] and [15]. The first set-partitioning strategy, referred to as *Full Partitioning MPC*, is followed by two alternating variants. All of these represent closed-loop RMPC approaches, differing primarily in the choice of partitioned dimensions per prediction step, the ordering of dimensions or in the number of subregions per time step.

2.8.1 Full partitioning MPC

This strategy was introduced in [14] for a monotone thermostat model and in [15] for a non-monotone cascade of continuous stirred-tank reactors. As its name suggests, Full Partitioning MPC partitions the entire $\mathcal{X}_i$ at each prediction step along predefined dimensions using a fixed search-tree-like partitioning scheme. No temporal variation is introduced, and the partitioning scheme remains unchanged in all steps. This scheme results in a uniform search-tree-like-structured subdivision of $\mathbb{X}_{[0:N]}$, producing N_s subregions per prediction step. Each subregion $s \in \mathbb{S}$ in a given $\mathbb{X}_i$ is associated with its own control input u_i^s. As a result, the optimization problem explicitly incorporates feedback by allowing the control law to branch depending on which subregion is reached, yielding a piecewise-constant feedback policy. Figure 2.16 illustrates two

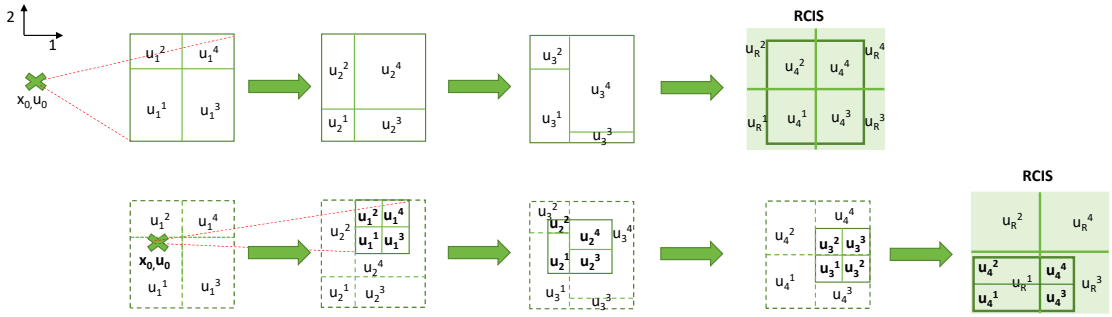

Figure 2.16: Illustration of full-partitioning MPC for 2D system with N=4 MPC [15]

two subsequent predictions of the reachable sets and their corresponding partitioning for four prediction steps in a two-dimensional system. The lower trajectory is computed one sample time step after the upper one. The control inputs of the lower trajectory are highlighted in bold. Recursive feasibility is visualized through containment by showing that $\mathbb{X}_i$ of the lower trajectory visualized by solid green lines lies within the shifted sets of the upper trajectory shown by dashed green lines. If the subregions of

the lower trajectory are properly aligned with those of the upper one, the control inputs from the upper trajectory can be reused.

One important concept in the context of recursive feasibility for the feedback-based set-partitioning MPC approaches,introduced in the following sections, is robust contolled invariance (RCI). RCI ensures that a system, initialized or once entered, remains within $\mathcal{X}_{RCIS}$, despite the presence of disturbances p, because there exists a control input u that can adequately counteract any such p. A $\mathcal{X}_{RCIS}$ can be regarded as a reachable set, represented in this work as a hyperrectangle, which recursively maps back into itself.

If $\mathcal{X}_{RCIS}$ is partitioned and feedback is assigned to the different subregions, the RCI property can be formalized as follows [33], [34]:

$$\forall x_0 \in \mathcal{X}_{RCIS},\ \exists\, u_{\text{RCIS}}^{[1:N_s]} \in \mathcal{U} \text{ such that } f(x_{\text{RCIS}}^{(s\pm)}, u_{\text{RCIS}}^{s}, p^{\pm}) \in \mathcal{X}_{RCIS},\ \forall s \in \mathcal{S}. \tag{2.38}$$

Equation (2.38) can be analogously extended using a decomposition function d in the mixed-monotone case [15].The final prediction step must lie within $\mathcal{X}_{RCIS}$, which can itself be partitioned according to the same scheme, enabling feasible terminal inputs for the lower trajectory.

As illustrated in Figure 2.16, a feasible trajectory can be generated by determining which subregion of the upper trajectory each control input belongs to and assigning it accordingly [14], [15]. According to (2.37), the N_s increases exponentially with the number of partitioned dimensions, resulting in exponential growth of the underlying ROCP with respect to the state dimension.

Despite this, Full Partitioning MPC offers significant performance advantages over open-loop methods, even with a small number of subregions. If N_s is chosen as one, the recourse property is lost and the approach transforms into a conservative open-loop RMPC approach [14]. To mitigate this curse of dimensionality, the next two sections introduce alternating schemes that preserve recourse while reducing computational complexity by varying partition order or resolution across the prediction horizon.

2.8.2 Alternating Constant Partitioning MPC

In Alternating Constant Partitioning MPC, the partitioning scheme is allowed to vary from one prediction step to the next, while keeping N_s constant. This means that at each prediction step $i \in \{0, \dots, N\}$, different state dimensions can be selected for partitioning, and in addition the order in which dimensions are split may change. The goal is to reduce the computational burden by minimizing N_s and distributing the feedback for different dimensions across the prediction horizon so that all dimensions are partitioned but not always at the same time. By varying the partitioning scheme, the dimensions of the reachable set receive a different feedback resolution at different prediction time steps.

Compared to Full Partitioning MPC, this approach is more flexible and allows one to focus on certain dimensions of $\mathcal{X}_i$ at different prediction steps, thus reducing the exponential growth in N_s associated with a fixed search-tree-like partitioning scheme.

For example, in a two-dimensional system, dimension one may be partitioned in step $i = 1$, while dimension two is partitioned in step $i = 2$, and so forth. This leads to a dynamic orientation of the subregions in each step of the prediction horizon. In different prediction steps, the subregions cover different parts of $\mathcal{X}_i$ and change orientation according to the prediction step. This effect is clearly visualized in Figure 2.17 over multiple prediction steps. In the upper trajectory, the subregions at the first prediction step span the entire second dimension and only a portion of the first each. In contrast, the subregions at the second step span the entire first dimension and only a portion of the second. The control inputs u_1^1 and u_2^1 differ accordingly as they must cover different regions of their $\mathcal{X}_i$.

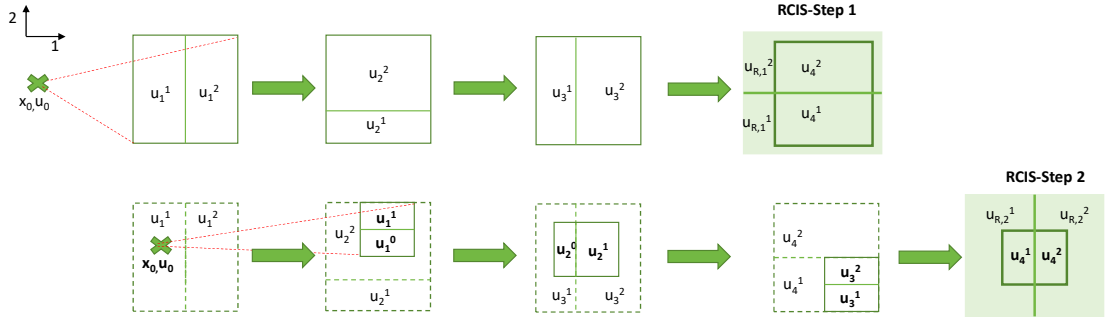

Figure 2.17: Illustration of alternating constant partitioning MPC for 2D system with N=4

Recursive feasibility is ensured by maintaining consistency in how subregions are aligned across prediction steps. Specifically, two ROCPs are solved in alternation, each associated with a alternating search-tree-like partitioning scheme. Defining these ROCPs as $P_{\text{CL}}^{\text{alt-const\_1}}(x_0)$ and $P_{\text{CL}}^{\text{alt-const\_2}}(x_0)$, the solver switches between two predefined alternating partitioning schemes in each subsequent sampling step so that after one sampling step, the newly generated subregions of $\mathcal{X}_i^s$ can be aligned with those of the previous trajectory $\mathcal{X}_{i+1}^{s'}$. This principle is shown in Figure 2.17, where the subregions of the lower trajectory can be aligned with the subregions of the upper trajectory, thus a feasible solution can be generated from the subregion.

Due to the alternating partitioning schemes in the alternating partitioning MPC approaches $N - step\, \mathcal{X}_{RCIS}$ as a generalization of $\mathcal{X}_{RCIS}$ can be considered for the terminal constraint in the context of recursive feasibility.

A $N-step\, \mathcal{X}_{RCIS}$ is defined as a sequence of reachable sets that periodically propagate into each other [7]:

$$N - step\, \mathcal{X}_{RCIS} = \{\mathcal{X}_{0,\text{RCIS}}, \ldots, \mathcal{X}_{N,\text{RCIS}}\}. \tag{2.39}$$

If the system reaches any set in this sequence, a sequence of control policies $\mu_i = u_{\text{RCIS},i}^{[1:N_{s,i}]}$ exists such that for all subregions in each $\mathcal{X}_{i,RCIS}$, the system evolves to the next set $\mathcal{X}_{i+1,RCIS}$, with $\mathcal{X}_{N,RCIS} \rightarrow \mathcal{X}_{0,\text{RCIS}}$. A schematic example of a 3-step RCIS is shown in 2.18.

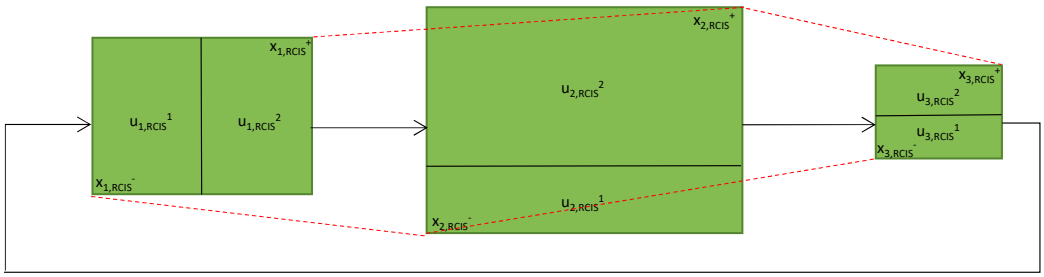

Figure 2.18: Schematic representation of 3-step RCIS

To ensure recursive feasibility at the terminal step $\mathcal{X}(N)$, a $2-step\,\mathcal{X}_{RCIS}$ can be utilized consisting of two reachable sets $\mathcal{X}_{RCIS,0}$ and $\mathcal{X}_{RCIS,1}$, that are partitioned according to the alternating search-tree-like partitioning scheme. In the end, $\mathcal{X}(N)$ must align with the correct $\mathcal{X}_{RCIS,i}$ depending on which partitioning scheme is used in $\mathcal{X}(N)$ or, more precisely, on which dimensions are partitioned in the prediction step N.

To conclude, Alternating constant partitioning MPC poses potential to reduce the complexity of the ROCP compared to Full Partitioning MPC. By ensuring that only one or a few dimensions of are partitioned in a prediction step and thus distributing the feedback over time, N_s per prediction can be kept smaller than in full partitioning MPC. At the same time, feedback is preserved because every state dimension is still addressed throughout the prediction horizon.

The following approach can be seen as a generalization of Alternating Constant Partitioning MPC as it allows for more flexibility in terms of varying N_s over time.

2.8.3 Alternating Variable Partitioning MPC

In Alternating Variable Partitioning MPC, the search-tree-like partitioning scheme allows for periodic variation not only in the order of partitioned dimensions and the number of partitions per dimension, but also in the total number of subregions across different prediction steps.

Alternating Variable Partitioning MPC introduces additional flexibility by allowing N_s to vary over time. This enables the reachable sets to alternate between a high number of subregions and a low number, such that in alternating prediction steps a high feedback resolution or a small feedback resolution can be injected in subsequent prediction steps and thereby balancing feedback quality and computational effort.

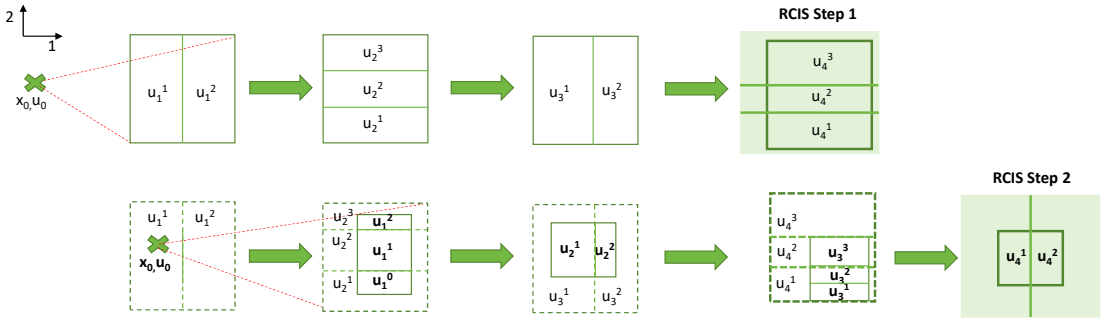

Figure 2.19: Illustration of alternating variable-partitioning MPC for 2D system with N=4

Figure 2.19 illustrates this principle for a two-dimensional system with four prediction steps. The $\mathcal{X}_i$ alternate between N_s=2 and N_s=3. The upper trajectory represents an existing feasible solution, in which $\mathcal{X}_1$ dimension one is partitioned once, and in $\mathcal{X}_2$, dimension two is partitioned twice. The lower trajectory represents a subsequent trajectory, for which a feasible solution can be inherited from the upper trajectory. $\mathcal{X}'_1$ is first partitioned twice in dimension two, and $\mathcal{X}'_2$ is partitioned once in dimension one. The subregions of $\mathcal{X}'_1$ and X'_2 can be aligned same as the following $\mathcal{X}'_i$ until N-1 prediction steps. In combination with the $2-step\,\mathcal{X}_{RCIS}$, the control inputs u_i^s for the lower trajectory can be inherited. Same as with Alternating Constant Partitioning MPC, recursive feasibility is again achieved by correctly aligning the subregions between consecutive trajectories of reachable sets.

Analogously to Alternating Constant Partitioning MPC, the two different ROCPs $P_{\text{CL}}^{\text{alt-variable\_1,N}}(x_0)$ and $P_{\text{CL}}^{\text{alt-variable\_2,N}}(x_0)$ are solved in subsequent sampling steps, with periodically varying constraints on the the search-tree-like partitioning scheme with respect to the ordering of the dimensions and a flexible number of partitions in the different dimensions. The optimization approaches for the three feedback-based set-partitioning approaches can be found in the appendix in B.0.2 and B.0.3.

2.9 Model Systems

All three presented set-partitioning strategies are applied to two systems of differing complexity in order to enable a comparative evaluation. These systems are briefly introduced in the following section.

2.9.1 Double Integrator

The first system is a discrete-time, nonlinear, monotone, two-dimensional double integrator, as described in [7]. This system is later used to investigate the influence of the parameterization of the alternating partitioning strategies on the control performance. The discrete system dynamics are given in Equation (2.40), with the corresponding matrices provided in Equation (2.41).

In Equation (2.40), the index i denotes an arbitrary prediction step, $x_i \in \mathbb{R}^2$ is the state vector, and $u_i \in \mathbb{R}^2$ is the control input. The setpoint is defined as $x_\mathrm{s} = (5,\, 2)^T$.

$$x_{i+1} = Ax_i + Bu_i + F\sqrt{x_i^T x_i} \tag{2.40}$$

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad F = \begin{pmatrix} p \\ p \end{pmatrix} \tag{2.41}$$

The state set $\mathbb{X}_d$, input set $\mathbb{U}_d$, and uncertainty set $\mathbb{P}_d$ are defined as box constraints in Equations (2.42)-(2.44):

$$\mathbb{X}_d = \left\{ x \in \mathbb{R}^2 \,\middle|\, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq x \leq \begin{pmatrix} 10 \\ 10 \end{pmatrix} \right\} \tag{2.42}$$

$$\mathbb{U}_d = \left\{ u \in \mathbb{R}^2 \,\middle|\, \begin{pmatrix} -10 \\ -5 \end{pmatrix} \leq u \leq \begin{pmatrix} 10 \\ 5 \end{pmatrix} \right\} \tag{2.43}$$

$$\mathbb{P}_d = \{ p \in \mathbb{R} \,|\, 0 \leq p \leq 0.3 \} \tag{2.44}$$

2.9.2 Continuous Stirred Tank Reactor (CSTR)

The second system is a nonlinear, non-monotone continuous stirred-tank reactor (CSTR) used for the production of cyclopentol (B). The reaction scheme is shown in 2.20. Cyclopentadiene (A) is converted into cyclopentol (B) through acid-catalyzed

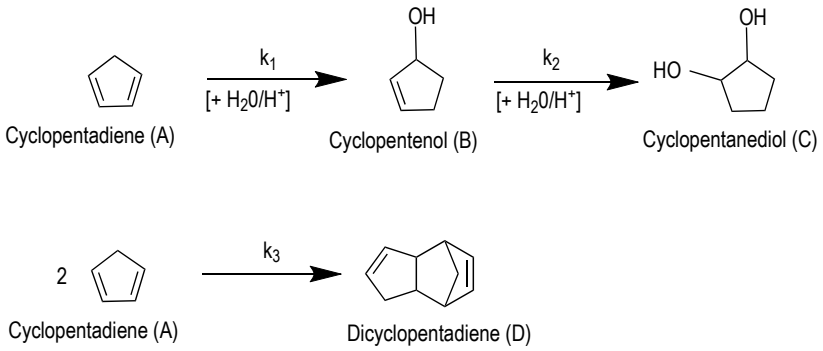

Figure 2.20: Reaction scheme for the production of cyclopentenol (B) from cyclopentadiene (A) [8]

electrophilic addition of water. Due to the high reactivity of B, a subsequent reaction may occur in which an additional water molecule adds to the remaining double bond of B, forming cyclopentanediol (C) as a byproduct. Furthermore, A is highly reactive and may undergo a Diels–Alder reaction to form dicyclopentadiene (D) [8].

A schematic diagram of the CSTR is shown in Figure 2.21, adopted from [8]. The first control variable F denotes the volumetric inflow rate normalized by the reactor volume V_r. F contains reactant A with a concentration c_{A0} and a temperature ϑ_0.

The second control variable is the extracted thermal energy, which is removed through the cooling jacket of the reactor and will simply be called *cooling rate* $\dot{Q}_k$.

The CSTR state variables are the concentration of reactant A within the reactor c_A, the concentration of product B c_B, the reactor temperature ϑ and the temperature of the cooling jacket ϑ_k. The governing equations are provided in Equations (2.45) to (2.48), which describe a non-linear fourth-order system with eleven uncertain parameters. It is assumed that the system operates with constant density ρ and follows an ideal residence time distribution.

$$\frac{dc_A}{dt} = F\big(c_{A0} - c_A\big) - k_{1,0}\exp\Big(-\tfrac{E_{A12}}{R\,(\vartheta+273.15)}\Big)\,c_A - k_{3,0}\exp\Big(-\tfrac{E_{A3}}{R\,(\vartheta+273.15)}\Big)\,c_A^2\ , \quad (2.45)$$

$$\frac{dc_B}{dt} = -\,F\,c_B + k_{1,0}\exp\Big(-\tfrac{E_{A12}}{R\,(\vartheta+273.15)}\Big)\,c_A - k_{2,0}\exp\Big(-\tfrac{E_{A12}}{R\,(\vartheta+273.15)}\Big)\,c_B\ , \quad (2.46)$$

$$\begin{aligned}\frac{d\vartheta}{dt} = {} & F\,\big(\vartheta_0 - \vartheta\big) + \frac{k_w\,A_r}{\rho\,C_p\,V_r}\,\big(\vartheta_k - \vartheta\big) - \frac{k_{1,0}\exp\big(-\frac{E_{A12}}{R\,(\vartheta+273.15)}\big)\,c_A\,\Delta H_{AB}}{\rho\,C_p} \\ & - \frac{k_{2,0}\exp\big(-\frac{E_{A12}}{R\,(\vartheta+273.15)}\big)\,c_B\,\Delta H_{BC}}{\rho\,C_p} - \frac{k_{3,0}\exp\big(-\frac{E_{A3}}{R\,(\vartheta+273.15)}\big)\,c_A^2\,\Delta H_{AD}}{\rho\,C_p}\ , \end{aligned} \quad (2.47)$$

$$\frac{d\vartheta_K}{dt} = \frac{1}{m_k\,C_{p,k}}\Big(Q_k + k_{w,c}\,A_r\,(\vartheta - \vartheta_K)\Big)\ . \quad (2.48)$$

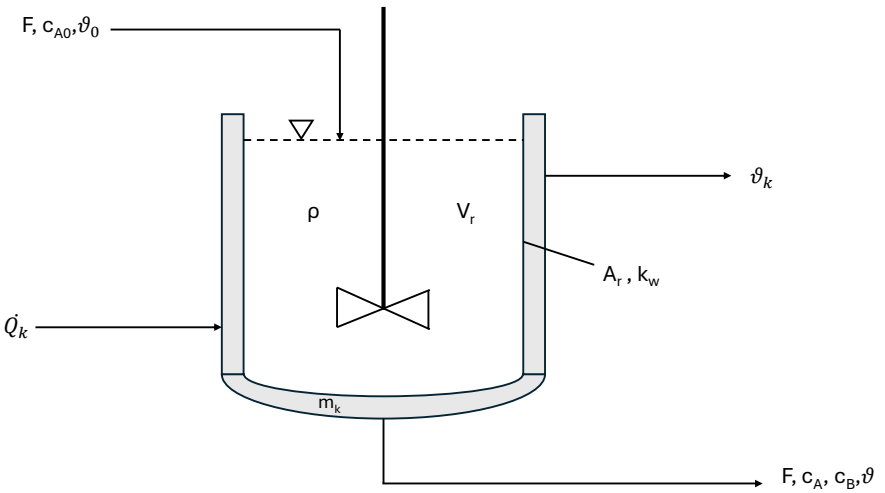

Figure 2.21: Sketch of the CSTR for the production of Cyclopentenol (B) [8]

Individual parameters, together with their associated uncertainties, are listed in the Appendix in C.1. The CSTR setpoints are given in the Appendix in C.2. These setpoints were adopted from [35] because they yield higher selectivity and substantially lower energy consumption than those of [8]. Equations (2.49) and (2.50) define, as box constraints, the admissible sets of state and control variables, respectively.

$$\mathbb{X}_{CSTR} = \Big\{\, x \in \mathbb{R}^4 \;\Big|\; \begin{pmatrix} 0 \\ 0 \\ 90 \\ 90 \end{pmatrix} \leq x \leq \begin{pmatrix} 3 \\ 2 \\ 120 \\ 120 \end{pmatrix} \Big\} \quad (2.49)$$

$$\mathbb{U}_{CSTR} = \left\{ u \in \mathbb{R}^2 \;\middle|\; \begin{pmatrix} 3 \\ -9000 \end{pmatrix} \leq u \leq \begin{pmatrix} 35 \\ 0 \end{pmatrix} \right\} \tag{2.50}$$

For the CSTR, an analytical minimum and maximum decomposition function via interval arithmetics can be obtained, as described in 2.26. d_{min} and d_{max} for the CSTR are given component-wise in 2.51-2.58.

$$\begin{aligned}\frac{dc_{A,\min}}{dt} &= F\left(c_{A0,\min} - c_{A,\min}\right) - k_{1,\max}\exp\left(-\tfrac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\right) c_{A,\min} \\ &\quad - k_{3,\max}\exp\left(-\tfrac{E_{A3}}{R\,(\vartheta_{\max}+273.15)}\right) c_{A,\min}^2 \end{aligned} \tag{2.51}$$

$$\begin{aligned}\frac{dc_{A,\max}}{dt} &= F\left(c_{A0,\max} - c_{A,\max}\right) - k_{1,\min}\exp\left(-\tfrac{E_{A12}}{R\,(\vartheta_{\min}+273.15)}\right) c_{A,\max} \\ &\quad - k_{3,\min}\exp\left(-\tfrac{E_{A3}}{R\,(\vartheta_{\min}+273.15)}\right) c_{A,\max}^2 \end{aligned} \tag{2.52}$$

$$\begin{aligned}\frac{dc_{B,\min}}{dt} &= -F\,c_{B,\min} + k_{1,\min}\exp\left(-\tfrac{E_{A12}}{R\,(\vartheta_{\min}+273.15)}\right) c_{A,\min} \\ &\quad - k_{2,\max}\exp\left(-\tfrac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\right) c_{B,\min} \end{aligned} \tag{2.53}$$

$$\begin{aligned}\frac{dc_{B,\max}}{dt} &= -F\,c_{B,\max} + k_{1,\max}\exp\left(-\tfrac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\right) c_{A,\max} \\ &\quad - k_{2,\min}\exp\left(-\tfrac{E_{A12}}{R\,(\vartheta_{\min}+273.15)}\right) c_{B,\max} \end{aligned} \tag{2.54}$$

$$\begin{aligned}\frac{d\vartheta_{\min}}{dt} &= F\left(\vartheta_{0,\min} - \vartheta_{\min}\right) + \frac{k_{w,\min}\,A_{r,\min}}{\rho_{\max}\,C_{p,\max}\,V_{r,\min}}\left(\vartheta_{k,\min} - \vartheta_{\min}\right) \\ &\quad - \frac{k_{1,\max}\exp\left(-\frac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\right) c_{A,\min}\,\Delta H_{AB,\max}}{\rho_{\max}\,C_{p,\max}} \\ &\quad - \frac{k_{2,\max}\exp\left(-\frac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\right) c_{B,\min}\,\Delta H_{BC,\max}}{\rho_{\max}\,C_{p,\max}} \\ &\quad - \frac{k_{3,\max}\exp\left(-\frac{E_{A3}}{R\,(\vartheta_{\max}+273.15)}\right) c_{A,\min}^2\,\Delta H_{AD,\max}}{\rho_{\max}\,C_{p,\max}} \end{aligned} \tag{2.55}$$

$$\begin{aligned}\frac{d\vartheta_{\max}}{dt} &= F\left(\vartheta_{0,\max} - \vartheta_{\max}\right) + \frac{k_{w,\max}\,A_{r,\max}}{\rho_{\min}\,C_{p,\min}\,V_{r,\max}}\left(\vartheta_{k,\max} - \vartheta_{\max}\right) \\ &\quad - \frac{k_{1,\min}\exp\left(-\frac{E_{A12}}{R\,(\vartheta_{\min}+273.15)}\right) c_{A,\max}\,\Delta H_{AB,\min}}{\rho_{\min}\,C_{p,\min}} \\ &\quad - \frac{k_{2,\min}\exp\left(-\frac{E_{A12}}{R\,(\vartheta_{\min}+273.15)}\right) c_{B,\max}\,\Delta H_{BC,\min}}{\rho_{\min}\,C_{p,\min}} \\ &\quad - \frac{k_{3,\min}\exp\left(-\frac{E_{A3}}{R\,(\vartheta_{\min}+273.15)}\right) c_{A,\max}^2\,\Delta H_{AD,\min}}{\rho_{\min}\,C_{p,\min}} \end{aligned} \tag{2.56}$$

$$\frac{d\vartheta_{K,\min}}{dt} = \frac{1}{m_k\,C_{p,k,\max}}\left(Q_k + k_{w,c,\min}\,A_r\,(\vartheta_{\min} - \vartheta_{K,\min})\right) \tag{2.57}$$

$$\frac{d\vartheta_{K,\max}}{dt} = \frac{1}{m_k\,C_{p,k,\min}}\left(Q_k + k_{w,c,\max}\,A_r\,(\vartheta_{\max} - \vartheta_{K,\max})\right) \tag{2.58}$$

The CSTR represents a realistic chemical industry example used to compare the performance of the three feedback-based set partitioning strategies with each other. It

can be investigated whether a monotone neural network, trained on these nominal trajectories, can provide a usable approximation of d for the CSTR model. In the next subsection, neural networks and monotonicity constraints for neural networks will be explained briefly.

2.10 Fundamentals of Neural Networks

Neural networks (NNs) are a cornerstone of deep learning and are employed successfully across a wide range of applications—today they are indispensable for solving complex tasks such as speech recognition or drug discovery with reasonable effort [36], [37]. NNs scale well to large datasets and can uncover intricate patterns within them [36]. The term "neural network" stems from the fact that NNs are inspired by the function and interplay of many neurons in the brain. However, NNs are not intended to replicate the exact biological processes of the brain but rather to address complex machine learning problems effectively. In addition to neuroscience, fields such as applied mathematics and statistics have been crucial to the success of modern NNs [37].

Fundamentally, a NN serves as a function approximator for an unknown target function f^*. Given pairs of input data x and corresponding labels y, the NN can learn to map $x \mapsto y$ by adjusting its parameters ϑ so that $f^*(x)$ is as closely approximated as possible. Architecturally, an NN is a composition of functions: linear-affine transformations alternate with non-linear activation functions, where the output of one activation layer becomes the input to the next linear layer. Concretely, each layer L consists of a linear–affine mapping

$$f_L : \mathbb{R}^{n_{L-1}} \to \mathbb{R}^{n_L}, \quad f_L\big(x_{L-1}\big) = W_L\, x_{L-1} + b_L, \tag{2.59}$$

followed by a nonlinear activation g_L. Starting from an input $x \in \mathbb{R}^{n_x}$, the NN produces an output $y \in \mathbb{R}^{n_y}$ by chaining these layers and activations:

$$f_{\mathrm{NN}}(x,\, \vartheta) = f_L \circ g_{L-1} \circ f_{L-1} \circ \cdots \circ g_1 \circ f_1(x). \tag{2.60}$$

The parameters ϑ collect all weight matrices W_L and bias vectors b_L for L layers. During training, these are adjusted so that the NN output y_{NN} matches the true labels y as closely as possible.

The choice of activation function, the number of layers L, and the width of each layer are key hyperparameters of the NN and can be optimized via systematic hyperparameter tuning. "Feedforward" means that the input x is processed layer by layer until the final output y_{NN} is produced.

The universal approximation theorem guarantees that a sufficiently large NN can approximate any continuous function to arbitrary accuracy, but it does not specify how many layers or neurons are needed, making the theorem primarily of theoretical interest. In practice, deep networks ($L > 1$) often outperform shallow ones ($L = 1$) even if the latter has a large width [37].

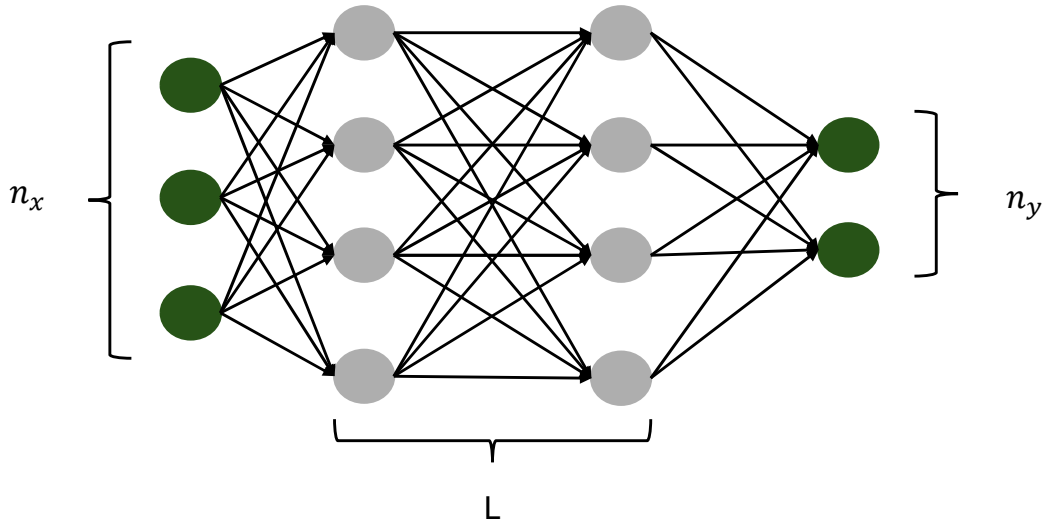

Figure 2.22: Schematic representation of a deep feedforward NN [38]

Training proceeds iteratively via gradient descent methods: a predefined loss metric like MSE for regression tasks is minimized so that the error between y_{NN} and the true labels y is driven down [37]. The dataset of input–output pairs is typically split into training and validation sets (often in a 70:30 ratio). During each training epoch, the weights and biases of the NN are adjusted using backpropagation and (mini-)batch stochastic gradient descent. Instead of computing the gradient over the entire training set, the gradient is approximated by randomly sampled mini-batches of size m, which yields both faster convergence and a degree of implicit regularization. At each epoch, both the training loss and the validation loss,based on MSE, are evaluated: both should be low, and the training loss should not be significantly lower than the validation loss. Additional regularization techniques, such as L2 weight decay, dropout layers, or early stopping, can help prevent overfitting. The goal of training is to achieve good generalization so that the NN can accurately predict unseen inputs. Finally, a separate test set can be used, distinct from the training and validation data, to verify the generalization of the model at the end of training [36], [37].

2.11 Monotonically constrainted Neural Network

NNs can also be constrained so that prior knowledge about the system to be approximated is explicitly incorporated. This includes monotonicity, which plays an essential role in this work. In a feedforward NN, monotonicity can be enforced in various ways. One approach is to add penalty terms to the loss function that penalize any violation of monotonicity. Alternatively, a range of NN architectures have been proposed that build monotonicity directly into their structure. This has the advantage that monotonicity is guaranteed in case the NN has a sufficient performance [39], [40], [41], [42], [43]. In this work, a *partially monotone* NN for the CSTR is considered in which monotonicity with respect to x and p is achieved by constraining all relevant weights

to be nonnegative. Furthermore, hyperbolic tangent (tanh) as the activation function is used, which itself is strictly increasing.

By training a NN that is partially monotone in both the state x and the parameters p, but not the control inputs u, it should be investigated to what extent a monotone NN can serve as a data-driven method to generate a valid decomposition function. The training trajectories are generated with the CSTR from section 2.9.2.

Figure 2.23 illustrates the architecture of the NN being partially monotone with regards to x and p. Partial monotonicity in x and p holds because all weights connecting those inputs up to the output layer are constrained to be positive. The control input u enters the network only at layer 2a. In that layer, partial monotonicity is ensured by splitting the weight matrix W_{2a} into two blocks: W_x, which multiplies the (x, p)-components and is constrained to be elementwise nonnegative, and W_u, which multiplies the u-component and remains unconstrained during training. Layers 1, 2b, and 2c likewise have monotone nonnegative weights, ensuring that the output of layer 2c is monotone in x and p. Since W_u is allowed to take both positive and negative values, the overall network is not guaranteed to be monotone in u. However, if W_{2a} were forced to be entirely non-negative, then the entire NN would be constrained to be monotone in all inputs. The Architecture of the NN is chosen based on some hyperparameter tuning studies that can be found in the Appendix in C.3.

2.11.1 Decomposition function from Neural Network

From the architecture of the neural network shown in 2.23, in principle a d for the CSTR dynamics can be approximated that satisfies the conditions in 2.22 and 2.24 and is monotone in both x and p. To obtain NN-based d, the weights in the final layer of the NN are divided into their positive part W_{2d}^{+} and their negative part W_{2d}^{-}. By performing two forward passes, z_1 and z_2, each of which is partially monotone in x and p ,but potentially using different inputs (x_1, p_1) and (x_2, p_2)), the NN's decomposition function d_{NN} can be defined as

$$d_{\mathrm{NN}}(x_1,\, x_2,\, u,\, p_1,\, p_2) \;=\; W_{2d}^{+}\, z_1(x_1,\, u,\, p_1) \;+\; W_{2d}^{-}\, z_2(x_2,\, u,\, p_2)\,. \tag{2.61}$$

In the chapters that follow, various case studies are presented, and both the different partitioning strategies and the neural network described above are discussed in light of the results.

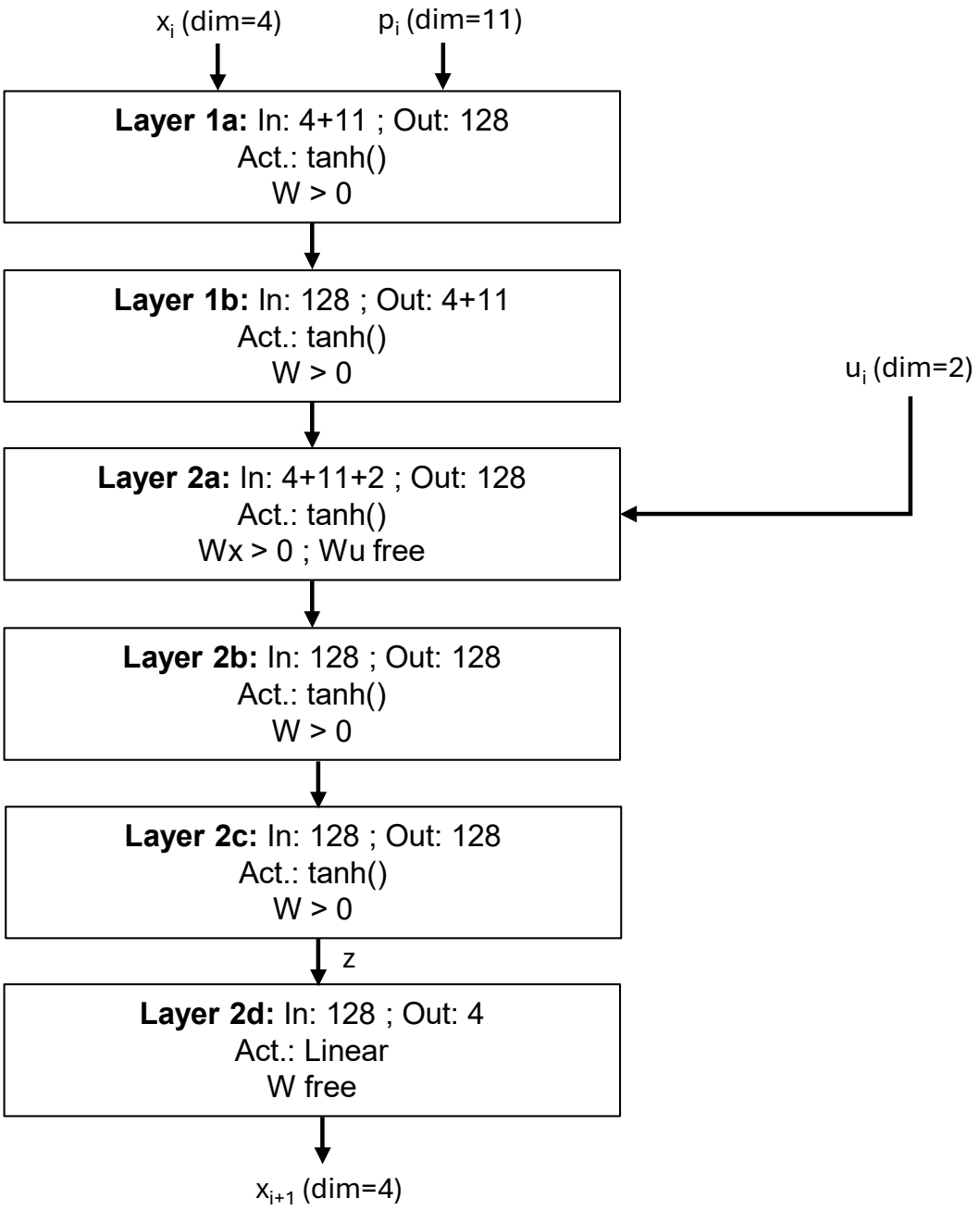

Figure 2.23: Structure of partially monotone NN with respect to x and u [38]

Chapter 3

Case studies

In the chapters that follow, case studies on the double integrator and the CSTR are presented to showcase the effectiveness of the proposed set-partitioning strategies. Afterwards the monotone NN's ability to generate a decomposition function, comparing its tightness and bounding performance against interval-arithmetics–based decompositions are evaluated. All optimizations were carried out with IPOPT, using the MA57 solver, in conjunction with CasADi . Simulations and Sampling are performed using the do-mpc framework [44], [45], [27]. Computations were executed on an Intel Core i5-12500H processor running at 4.5 GHz.

3.1 Case studies with double integrator

To begin with, the double integrator system from section 2.9.1, which is less complex, is used to compare the three feedback-based set-partitioning strategies. Specifically, the objective is to assess whether the Alternating-Constant Partitioning MPC and Alternating-Variable Partitioning MPC offer performance or runtime advantages over the established Full Partitioning MPC.

For this purpose, both the closed-loop cost (J_{CL}) and the computation time per MPC iteration-step (t_{CL}) are evaluated. J_{CL}, given in 3.1, includes a quadratic stage term, similar to 2.2, but is computed based on the true system states x_{CL} and inputs u_{CL}, thus providing a measure of closed-loop performance for a tracking objective.

$$J_{\text{CL}}(i) = (x_{i,\text{CL}} - x_{\text{s}})^T Q (x_{i,\text{CL}} - x_{\text{s}}) + (u_{i,\text{CL}} - u_{i-1,\text{CL}})^T R (u_{i,\text{CL}} - u_{i-1,\text{CL}}) \tag{3.1}$$

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

The setpoint $x_s = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$ is used, making J_{CL} a suitable indicator of tracking performance. The variable t_{CL} denotes the time required to complete one iteration in the closed MPC loop with the real system.

For the simulations, the double integrator system is evaluated with a constant uncertainty parameter $p = 0.15$. To compare the three partitioning strategies, both J_{CL} and t_{CL} are analyzed under different variations of two key parameters with regards to flexibility of the set-partitioning strategies, that is: the order in which the dimensions are partitioned, termed as ordering, and the total N_s within the reachable sets.

Figures 3.1 to 3.4 show t_{CL} on the left and J_{CL} on the right plotted against N_s for the three partitioning strategies. Orange dots represent Alternating-Constant Partitioning MPC, green dots represent Alternating-Variable Partitioning MPC, and blue dots represent Full Partitioning MPC. Each figure corresponds to a different ordering of the partitioned dimensions x_1 and x_2, which is indicated in the figure captions.

All four possible ordering-combinations for the alternating strategies are tested. In each case, the ordering alternates between two configurations in successive prediction steps, while Full Partitioning MPC maintains a constant partitioning scheme, and thus a fixed ordering across the prediction horizon. For Full Partitioning MPC, each state dimension is always partitioned identically. In Alternating Constant Partitioning MPC, N_s remains constant, but partitioning alternates between dimensions. In Alternating Variable Partitioning MPC, the number of subregions varies, and the average N_s is plotted, calculated from $N_{s1} = N_s - 2$ and $N_{s2} = N_s + 2$. In the appendix the Figures 3.4 to A.88 represent the closed-loop trajectories for the different settings in the Figures 3.1 to 3.4 for both the states and the control inputs of the double integrator. The trajectories are plotted in blue while the constraints are plotted with red dashed lines and the setpoints are illustrated with red solid lines.

In Figures 3.1 to 3.4 J_{CL} for Full Partitioning MPC remains consistently low and changes very little as N_s increases. In contrast, Alternating Variable Partitioning MPC shows markedly improved tracking performance with larger N_s, and its J_{CL} even falls below that of Full Partitioning when N_s is sufficiently high. Alternating Constant Partitioning MPC behaves similarly to Full Partitioning: its J_{CL} decreases only slightly, though at $N_s = 54$ it does dip marginally below the Full Partitioning value.

A closer look at the closed-loop trajectories for Full Partitioning and Alternating Constant Partitioning MPC at $N_s = 9$ and $N_s = 54$ in the Appendix in Figures A.5, A.11, A.12, A.18 reveals that increasing N_s reduces oscillations. For $N_s = 54$, both alternating partitioning strategies yield trajectories that track the setpoints almost as closely as Full Partitioning MPC, although small oscillations persist in Alternating Variable Partitioning MPC as can be seen in Figure A.25. These residual oscillations are inherent to the alternating partitioning strategies. They can be further attenuated by partitioning more state dimensions or by choosing an even larger N_s, but the very flexibility in the varying N_s is also why they remain.

In practice, one should select N_s for the alternating methods so as to achieve good tracking performance with minimal oscillations, without making N_s so large that t_{CL}

becomes excessive. Oscillations in the control inputs are particularly undesirable, since they can increase actuator wear.

For the double-integrator system, Full Partitioning MPC already achieves excellent tracking at $N_s = 9$ with minimal computation time. Indeed, t_{CL} is lowest, and nearly identical, for Full Partitioning and Alternating Constant MPC, whereas Alternating Variable MPC typically incurs a significantly higher t_{CL}. As N_s grows, the gap in t_{CL} between Alternating Constant Partitioning MPC, Full Partitioning MPC and Alternating Variable MPC widens further.

Figures 3.1– 3.4 clearly reflect these trends: t_{CL} is highest for Alternating Variable Partitioning MPC, while Full Partitioning and Alternating Constant Partitioning MPC remain lower. In terms of J_{CL}, Full Partitioning MPC is best at $N_s = 9$, but for larger N_s both alternating partitioning strategies yield even lower values for J_{CL}. All three strategies are recursively infeasible for $N_s < 9$. Thus, by choosing a moderate N_s and tolerating only small oscillations, Alternating Constant and Alternating Variable Partitioning MPC can achieve perfectly acceptable tracking performance. The specific ordering of dimensions has only a minor effect and does not alter these overall trends in J_{CL} and t_{CL}.

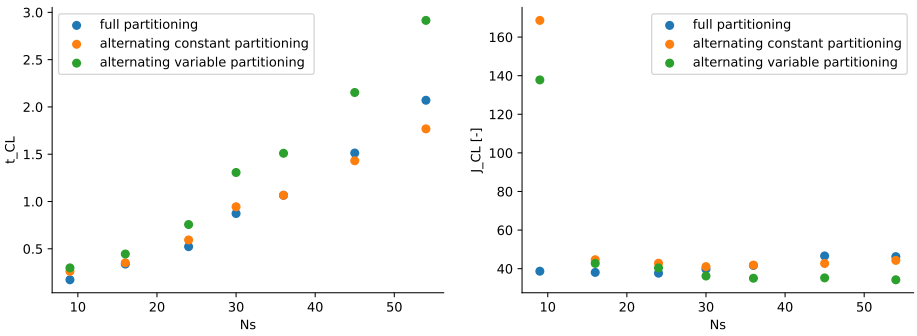

Figure 3.1: Comparison of Full Partitioning MPC, Alternating Constant Partitioning MPC and Alternating Variable Partitioning MPC with J_{CL} and t_{CL} over N_s for ordering [0,1][0,1]

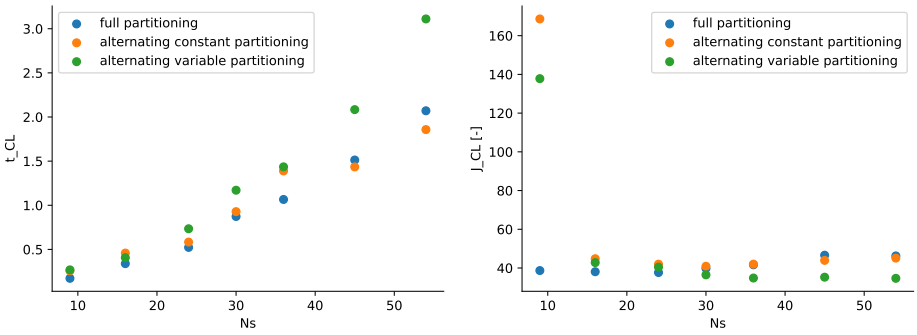

Figure 3.2: Comparison of Full Partitioning MPC, Alternating Constant Partitioning MPC and Alternating Variable Partitioning MPC with J_{CL} and t_{CL} over N_s for ordering [0,1][1,0]

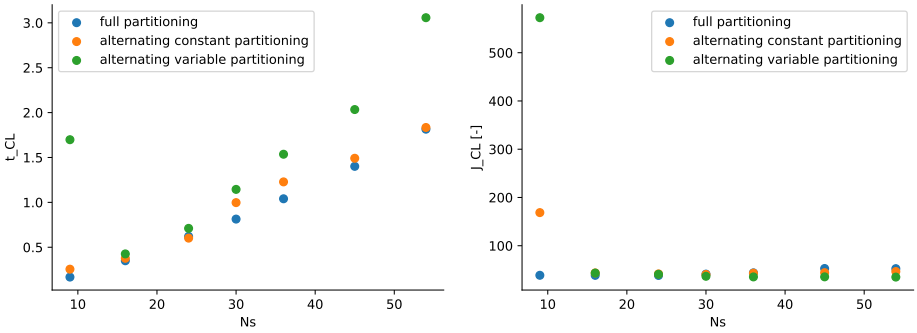

Figure 3.3: Comparison of Full Partitioning MPC, Alternating Constant Partitioning MPC and Alternating-Variable Partitioning MPC with J_{CL} and t_{CL} over N_s for ordering [1,0][1,0]

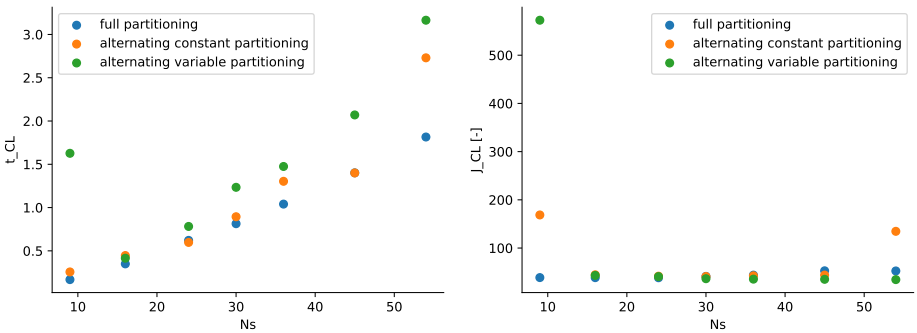

Figure 3.4: Comparison of Full Partitioning MPC, Alternating Constant Partitioning MPC and Alternating Variable Partitioning MPC with J_{CL} and t_{CL} over N_s for ordering [1,0][0,1]

It is clearly visible that the alternating set-partitioning MPC approaches induce oscillations in both the states (x_{CL}) and the control inputs (u_{CL}) of the closed loop. These oscillations decrease as N_s increases in the alternating strategies. This can be attributed to the fact that with larger N_s, the alternating partitioning strategies increasingly resemble Full Partitioning MPC, effectively covering all dimensions of x, albeit partitioned with varying frequency across prediction steps.

If, however, only one dimension is partitioned per prediction-step in an alternating fashion as for example, x_1 in one step and x_2 in the next, the oscillations—especially in u_1 and u_2—persist, and the tracking performance degrades in comparison when both dimensions are partitioned, but with a different number of partitions distributed in differnt prediction steps. This can be validated in Figure A.89, in which Alternating-Constant Partitioning is applied with N_s = 16, but only one dimension is partitioned per prediction step resulting in a worse tracking performance and higher J_{CL} than compared with Figure A.76. Here Alternating Constant Partitioning MPC is applied with N_s = 16, but the two dimensions obtain a different number of partitions in different prediction steps. With this partitioning scheme, the closed-loop trajectory approaches the setpoint closer and also there are slightly less oscillations especially with respect to u_{CL}.

Figure 3.5 visualizes a possible reason for the oscillations with regards to u for a two-dimensional system by showing the coverage of different subregions that are marked in red. The Alternating Constant partitioning MPC first partitions dimension one and then dimension two in the next prediction step. The red area indicates the spatial region covered by a given subregion in $\mathcal{X}_i$ across these steps. As evident, the red subregion covers the dimensions of $\mathcal{X}_i$ to different extents in successive steps. Consequently, the control inputs u_i^s associated with these red subregions also differ, as they cover different areas of state space.

The oscillations observed in the closed-loop inputs are thus inherent to the alternating partitioning MPC approaches. These can be mitigated by ensuring that more than one dimension is partitioned in consecutive prediction steps. By varying the partitioning frequency across dimensions over the prediction horizon, feedback can be distributed more evenly over time. As shown in Figures A.5 to A.88, the closed-loop trajectories of the alternating partitioning strategies approach those of Full Partitioning MPC with higher values of N_s.

In conclusion, for the alternating partitioning strategies, the ordering of dimensions appears to have only a minor influence on the tracking performance, as both t_{CL} and J_{CL} show similar trends across different orderings. A much more decisive factor is the value of N_s and the frequency with which different dimensions are partitioned across the prediction horizon. The oscillations in the control inputs are intrinsic to these strategies and may persist even if relatively small. However, low values of J_{CL} with alternating partitioning strategies indicate a trade-off between slower computation due to increased flexibility and eventually better closed-loop performance than Full Partitioning MPC if N_s is sufficiently high.

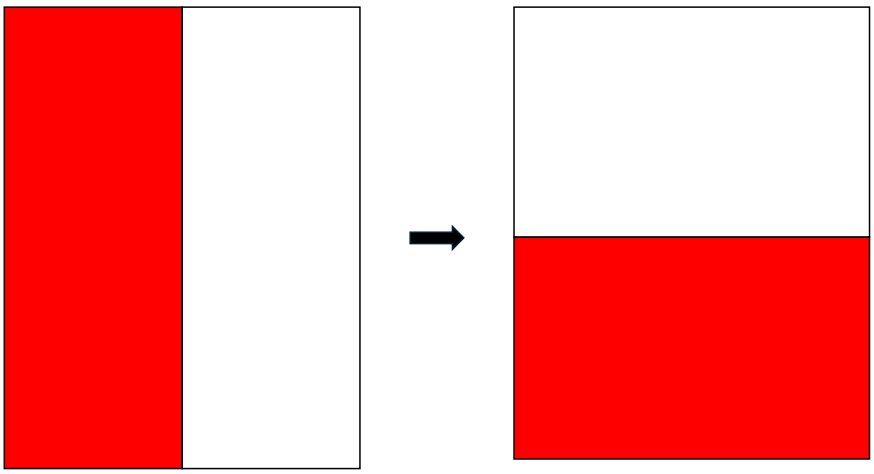

Figure 3.5: Schematic representation of spatial alignment of subregions in two consecutive prediction steps

3.2 Case studies with CSTR

In the following section, the three MPC set-partitioning strategies are applied to the previously introduced CSTR from section 2.9.2, which serves as a more complex and realistic model system. The goal is to validate how well the alternating MPC set-partitioning strategies perform when applied to the CSTR.

The CSTR features 11 parameters that are subject to uncertainties. They can be seen in Table C.1 in the appendix. Furthermore, the CSTR is a non-monotone system. A potential decomposition function is defined via interval arithmetics, as given by Equations 2.51 to 2.58 in section 2.9.2.

However, the decomposition function d_{max} obtained from interval arithmetics yields excessively large overapproximations for ϑ and ϑ_K, which are several orders of magnitude higher than the considered temperature interval for these variables. Figure 3.6 illustrates this issue clearly. The Figure contains four plots for the trajectories of the state variables c_A, c_B, ϑ and ϑ_K of the CSTR. The red dashed lines represent the constraints while the red solid lines represent the setpoints. The green curves represent boundings around 10 nominal trajectories with varying combinations of uncertain parameters shown in blue. The green curves are based on the analytical decomposition-function which can be formulated as d_{min} for providing lower bounds and d_{max} for providing upper bounds around the nominal trajectories same as described in 2.9.2. By observing the bounding trajectories for ϑ and ϑ_K, the excessive conservatism of d_{min} and d_{max} becomes immediately apparent.

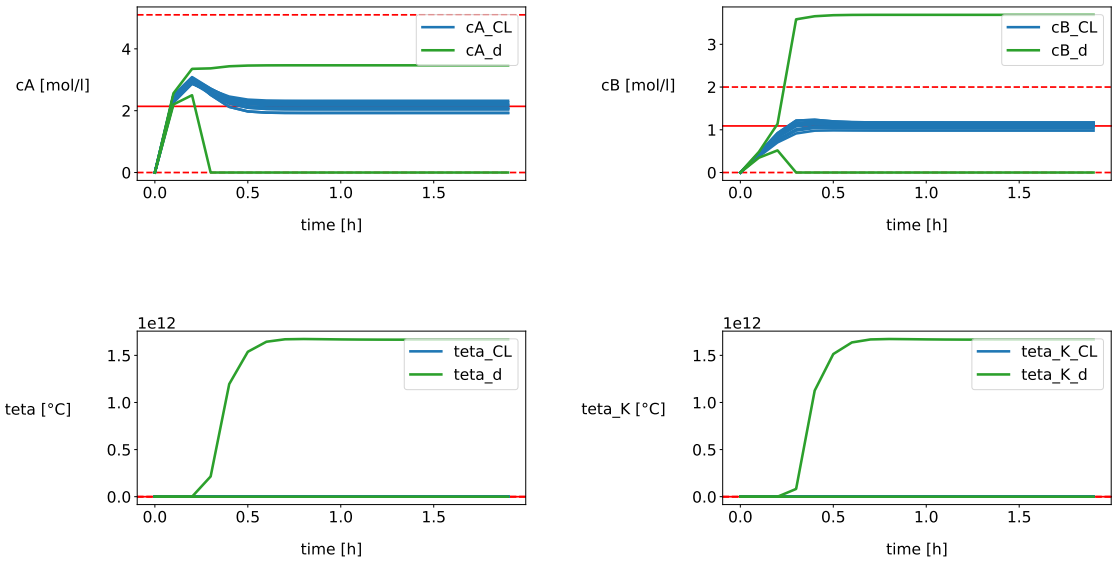

Figure 3.6: Illustration of bounding of the analytical decomposition function from section 2.9.2 shown in green and the nominal trajectories in blue

The analytical decomposition function obtained via interval arithmetics is therefore ineffective for computing reachable set overapproximations in the context of the mentioned partitioning strategies. Specifically, ϑ and ϑ_K are severely over-approximated. Based on this intuition, d_{max} is relaxed. It is hypothesized that excessive overapproximations arise from the exponential arrhenius terms in 2.52 and 2.54 high values

for $c_{A,max}$ and $c_{B,max}$. Consequently, in 2.52 and 2.54 $\vartheta_{\min}$ is replaced by $\vartheta_{\max}$, so that the maximal values for c_A and c_B are reduced and thus the over-approximations with regards to ϑ and ϑ_K are reduced. The Following relaxed terms are obtained for $d_{cAmax,relax}$ and $d_{cBmax,relax}$ and are given below.

$$\begin{aligned}\frac{dc_{A,\max,relax}}{dt} &= F\left(c_{A0,\max} - c_{A,\max}\right) - k_{1,\min}\exp\Big(-\tfrac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\Big)\,c_{A,\max} \\ &\quad - k_{3,\min}\exp\Big(-\tfrac{E_{A3}}{R\,(\vartheta_{\max}+273.15)}\Big)\,c_{A,\max}^2 \end{aligned} \tag{3.2}$$

$$\begin{aligned}\frac{dc_{B,\max,relax}}{dt} &= -F\,c_{B,\max} + k_{1,\max}\exp\Big(-\tfrac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\Big)\,c_{A,\max} \\ &\quad - k_{2,\min}\exp\Big(-\tfrac{E_{A12}}{R\,(\vartheta_{\max}+273.15)}\Big)\,c_{B,\max} \end{aligned} \tag{3.3}$$

The other components from d_{min} and d_{max} , 2.51, 2.53, 2.55, 2.57, 2.56 and 2.58, remain unchanged and retain the same form as defined in Section 2.9.2 .

The resulting relaxed decomposition function $d_{\max,\text{relax}}$ deviates slightly from the analytical form obtained by interval arithmetics and does not provide bounding guarantees. However, it yields very practical results when used in combination with the already proposed set-partitioning strategies.

For instance, it can be applied in Full Partitioning MPC, and simulations under parametric uncertainty demonstrate that this strategy remains robust and avoids constraint violations, whereas standard MPC might fail due to lack of uncertainty handling. Validating the robustness of the set-partitioning strategies in a realistic scenario and ensuring that the relaxed decomposition function still produces reliable and robust results will be done in the following.

Several simulations are presented in which the eleven parameters of CSTR are randomly varied. Figure 3.7 shows 100 nominal trajectories for the states c_A, c_B, ϑ, and ϑ_K of the CSTR over time. Each trajectory corresponds to a different random realization of the 11 uncertain parameters of the CSTR. The state constraints are visualized as dashed lines, indicating upper and lower bounds. It can be observed that some of the nominal trajectories violate these constraints, particularly for c_A and ϑ. Figure 3.8 also depicts uncertain trajectories but with Full Partitioning MPC. Again, 100 realizations of the uncertain parameters are used in the different trajectories. In contrast to the nominal MPC case, no constraint violations occur. Depending on the uncertainty realization, the operating point moves closer to or farther from the constraints. The same simulations with 100 realizations of the uncertain parameters are made for Alternating Constant MPC and Alternating Variable Partitioning MPC. The corresponding trajectories can be found in the Appendix in Figure A.112 and Figure A.113. No constraint violations can be observed for the alternating partitioning strategies either. However, oscillations are still observable in the trajectories, most pronounced with the Alternating Variable Partitioning MPC. It is plausible that these arise from the increased flexibility inherent to this partitioning strategy. The control inputs are ultimately computed in successive stages for varying numbers of subregions. Overall, the simulations demonstrate that the partitioning strategies can indeed yield robust per-

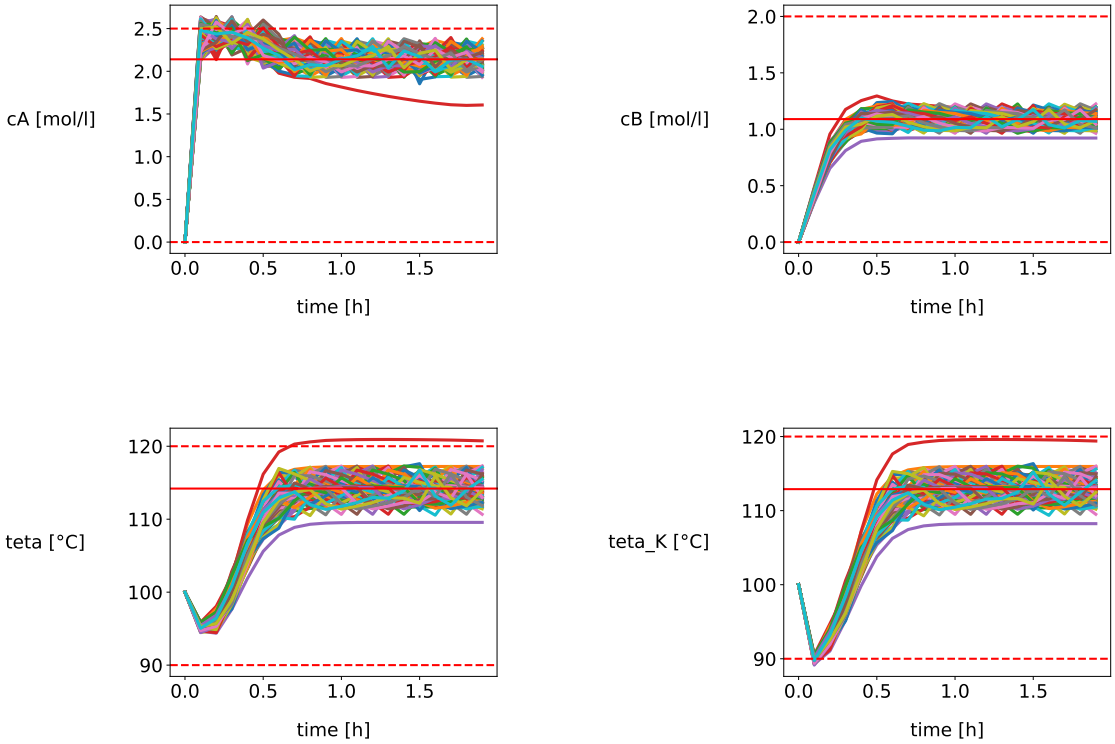

Figure 3.7: Nominal Trajectories for 100 random combinations of 11 uncertainty parameters of the CSTR

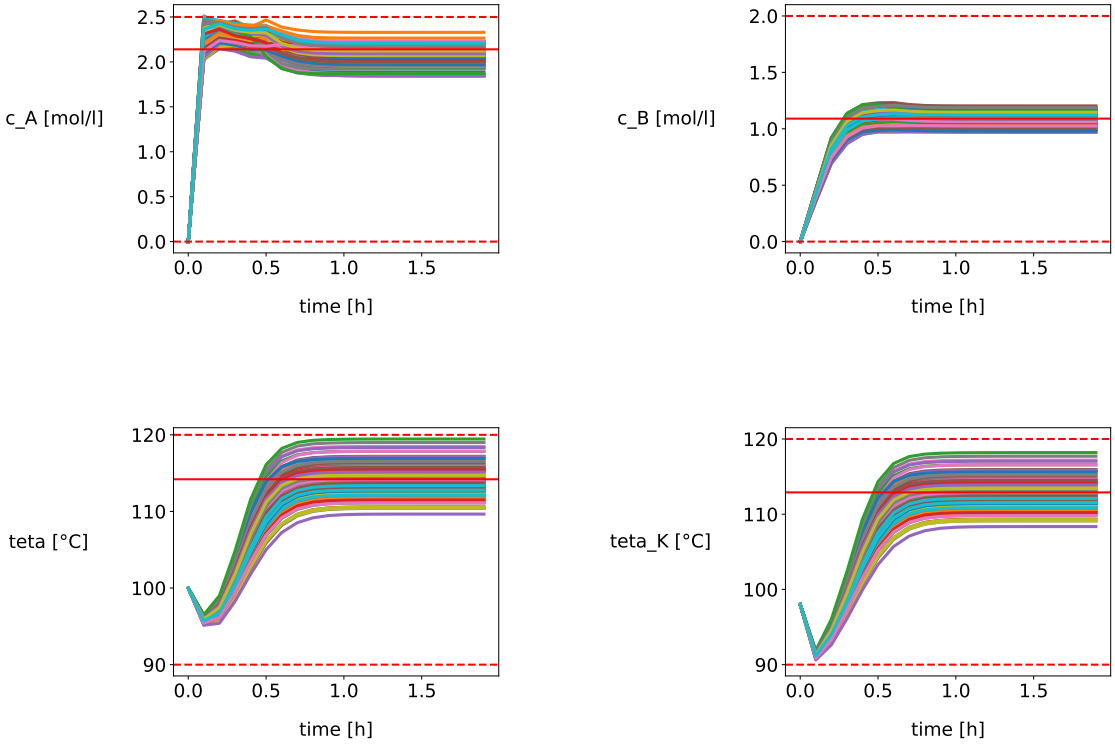

Figure 3.8: Full Partitioning MPC-Trajectories for 100 random combinations of 11 uncertainty parameters of the CSTR

formance also very close to constraints. The relaxed decomposition function likewise exhibits robust behavior under parametric uncertainty and thus will be employed in the subsequent analysis.

Following the previous section, the alternating partitioning strategies are evaluated using the relaxed decomposition function and the CSTR as a reference model. To assess tracking performance, a quadratic cost function similar to the one used for the double integrator is applied, but with adjusted R and Q-Matrices with respect to the dimensions of the CSTR.

Figure 3.9 presents two plots showing t_{CL} and J_{CL} over different values of N_s for Full Partitioning MPC in blue, Alternating Constant Partitioning MPC in orange, and Alternating Variable Partitioning MPC in green. For this comparison, it is assumed that all the model parameters are constant and correspond to the reference values in Table C.1. Overall, the trends closely mirror those seen for the double-integrator:

- **Computation time:** Full Partitioning MPC almost always achieves the lowest t_{CL}, followed by Alternating Constant Partitioning MPC, then Alternating Variable Partitioning MPC.
- **Feasibility at $N_s = 0$:** Unlike the double integrator, Alternating Constant Partitioning MPC and Alternating Variable Partitioning MPC remain well-posed on the CSTR even when $N_s = 0$. In that case, Full Partitioning MPC and Alternating Constant Partitioning MPC both attain the smallest J_{CL}, whereas Alternating Variable Partitioning MPC yields the highest closed-loop cost.
- **J_{CL} from $N_s = 4$:** At $N_s = 4$, Alternating Variable Partitioning shows the lowest J_{CL}, then Alternating Constant Partitioning MPC, and finally Full Partitioning MPC.

In the Appendix Figures A.90 to A.107 illustrate the corresponding closed-loop trajectories for the four CSTR states c_A, c_B, ϑ, ϑ_K in the top row, and for the inputs (F, $\dot{Q}_K$) plus J_{CL} in the bottom row for the different settings in Figure 3.9. Red dashed lines mark the constraints, and solid red lines indicate the setpoints.

A comparison of the closed-loop trajectories for the different partitioning strategies at $N_s = 4$ in Figures A.90, A.96, A.102 reveals that Full Partitioning MPC exhibits larger deviations from the setpoints, hence worse tracking, than Alternating Constant Partitioning MPC, which shows virtually no oscillations. Alternating Variable Partitioning MPC also tracks very closely, but still displays noticeable oscillations in c_A and F. As N_s increases, the J_{CL} curves for Full and Alternating Constant Partitioning MPC converge so tightly that the orange data points obscure the blue. Even at $N_s = 40$, Alternating Constant Partitioning continues to follow the setpoints more closely than Full Partitioning, and Alternating Variable Partitioning MPC at $N_s = 40$ shows markedly reduced oscillations compared to $N_s = 4$. The trajectories of Alternating Constant Partitioning MPC and Alternating Variable Partitioning MPC become very similar at $N_s = 40$ as can be seen in Figures A.101 and A.107.

In summary, the CSTR results echo those of the double integrator: the added flexibility of the alternating partitioning schemes comes at the cost of increased computation

time, but delivers lower closed-loop cost. Moreover, the trajectory plots confirm that, for suitably chosen N_s, both Alternating Constant and Alternating Variable Partitioning MPC can track the setpoints even more tightly than Full Partitioning MPC. Residual oscillations—most pronounced in the Alternating Variable scheme at low N_s diminish as N_s grows.

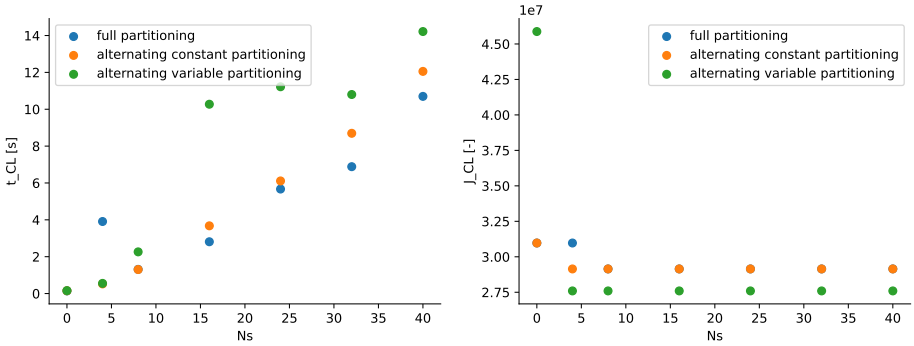

Figure 3.9: t_{CL} and J_{CL} for Full Partitioning MPC, Alternating-Constant MPC, and Alternating-Variable MPC using partitioning orders $[0, 1, 2, 3]$ and $[3, 2, 1, 0]$

3.3 Approximation capabilities of monotone Neural Network

The motivation behind employing a monotone NN in this work is to approximate a decomposition function with the approach from Section 2.11.1 that may yield tighter bounds than the analytical decomposition function derived using interval arithmetics. It should also be investigated how the approximate decomposition function of the NN performs compared to the relaxed decomposition function. The results based on this approach are presented and discussed in the following section.

The architecture of the monotone NN is shown in Figure 2.23. It is constructed to be positively monotone with respect to the variables x_i and p_i, which is enforced by constraining the relevant network weights to be positive. While u_i is also used as an input to the monotone NN, but no monotonicity constraint is imposed on it, since the goal is to compute reachable sets based on $x_i^{\pm}$ and $p_i^{\pm}$, which are then over-approximated as hyperrectangles.

The monotone NN is designed to predict the future states of the system x_{i+1} based on a 17-dimensional input vector comprising x_i, u_i, and p_i. Here, x_i and x_{i+1} are vectors that describe the CSTR states, namely concentrations c_A, c_B, reactor temperature ϑ, and coolant temperature ϑ_k. The control input u_i includes the normalized feed flow F_i and the cooling rate $\dot{Q}_k$. The uncertainty vector p_i consists of the 11 parameters c_{A0}, $k_{1,0}$, $k_{2,0}$, $k_{3,0}$, ΔH_{AB}, ΔH_{BC}, ΔH_{AD}, ρ, k_w, C_p, and $C_{p,k}$. For more information, refer again to Section 2.9.2.

The monotone NN is trained using 9280 nominal closed-loop MPC trajectories with $N_{sim} = 20$. These trajectories include random variations in both the initial states of the CSTR and the uncertainty realizations of p_i, to learn various kinds of system behavior. Additionally, a 2% Gaussian white noise is applied to x_i and u_i respectively to simulate noise disturbances and thus make the data more realistic. In the case of single-point predictions, the values x_i, u_i, and p_i of a single MPC iteration step serve as input, while x_{i+1} is the corresponding label, which the network aims to predict as accurately as possible. Similarly, x_i represents the label derived from x_{i-1}, u_{i-1}, and p_{i-1} from the preceding MPC iteration step. In total, 92800 input-output-label pairs are generated. The dataset is randomly divided into 70% training data, 15% validation data, and 15% test data.

The coverage plot for the training data are provided in the supplementary material. In the coverage plot, the four states of the reactor are plotted against the corresponding 17 inputs each in various subplots.Thus it can be seen which regions are well trained and which ones leak data. The spread of inputs demonstrates that the nominal closed-loop trajectories under varying uncertainties result in a comprehensive input space coverage. The state inputs and control inputs show a very dense coverage in some regions, whereas others are not covered and thus the NN will be able to only work precisely in the more dense regions, since only there is a sufficient amount of data. The stripe-like structures in the parameter-input subplots stem from the fact that the uncertainty realizations for the parameters are either time-invariant or randomized per trajectory, and the steady-state behavior of the closed-loop MPC leads to repeated parameter combinations over time which explains the strip-like structure.

Initially, the monotone NN is trained solely for single-point predictions. The loss function used is the MSE. To improve generalization and prevent overfitting, dropout layers and L2 regularization with a weight decay of 0.001 are used. A learning rate scheduler is used, which reduces the learning rate by 70% after 10 epochs of no improvement in the validation loss. The learning rate starts at 10^{-3} and is limited to a minimum of 10^{-5}. An early stopping criterion is used to terminate training if the validation MSE does not improve for 70 epochs. A batch size of 2000 and a maximum of 1050 epochs are set.

Table C.3 in the appendix presents alternative monotone NN architectures and their respective training, validation, and test MSEs. The chosen network achieves sufficient performance and is thus used to derive the decomposition function following the approach outlined in Section 2.11.1. According to the pair plot in Figure A.108, the monotone NN achieves high prediction accuracy for single-point predictions.

In Figure 3.10, the blue curves show nominal closed-loop trajectories for the states of the CSTR for constant parameters and without uncertainties, while the black curves represent the trajectories predicted by the single-point monotone NN that should approximate the blue nominal trajectories. The red solid lines represent the setpoints and the red dashed lines mark the constraints. Despite assuming constant parameters and no uncertainties, the black curves deviate significantly from the nominal ones, indicating poor trajectory predictions, which may be due to the accumulation of errors over time.

To address this issue, the monotone NN is re-trained using a trajectory-learning approach. In this case, the input at each prediction step is the output of the NN from the previous prediction step, that is, $x_{i,\text{NN}}$. The goal is to minimize the cumulative deviation throughout the trajectory, as the NN predictions should be accurate for the N_{sim} prediction steps to be able to approximate the learned trajectories. Figure 3.11 shows that the retrained monotone NN provides a much better approximation capability of the nominal trajectories. However, it may perform worse on isolated single-point predictions compared to the single-point trained network, as seen in the pair plot in A.110.The monotone neural network with trajectory learning is specifically trained to approximate the various nominal trajectories in the training data as accurately as possible, rather than focusing on single-point predictions.

The trajectory-trained monotone NN is used to derive a decomposition function following the procedure in Section 2.11.1. As already described, this decomposition function is derived using the positive and negative weights from the final layer of the monotone NN. In Figure 3.12, the bounding capabilities of the monotone NN decomposition function are compared with the relaxed decomposition function. Ten nominal trajectories are plotted in blue with different uncertainty realizations, including the worst-case uncertainty-combinations, together with the bounds provided by the NN-derived decomposition function in yellow and the bounding of the relaxed analytical decomposition function in green. The red solid lines mark the setpoints and the red dashed lines mark the constraints. With the exception of a slight violation for c_A at $t_{\text{sim}} \approx 0.3$, the yellow bounds encompass the nominal trajectories well and are significantly tighter than the green ones, especially in the steady-state region. Furthermore, it can be observed that the upper bound of the relaxed decomposition function is exceeded by the nominal trajectories, for all state variables at various points in time.

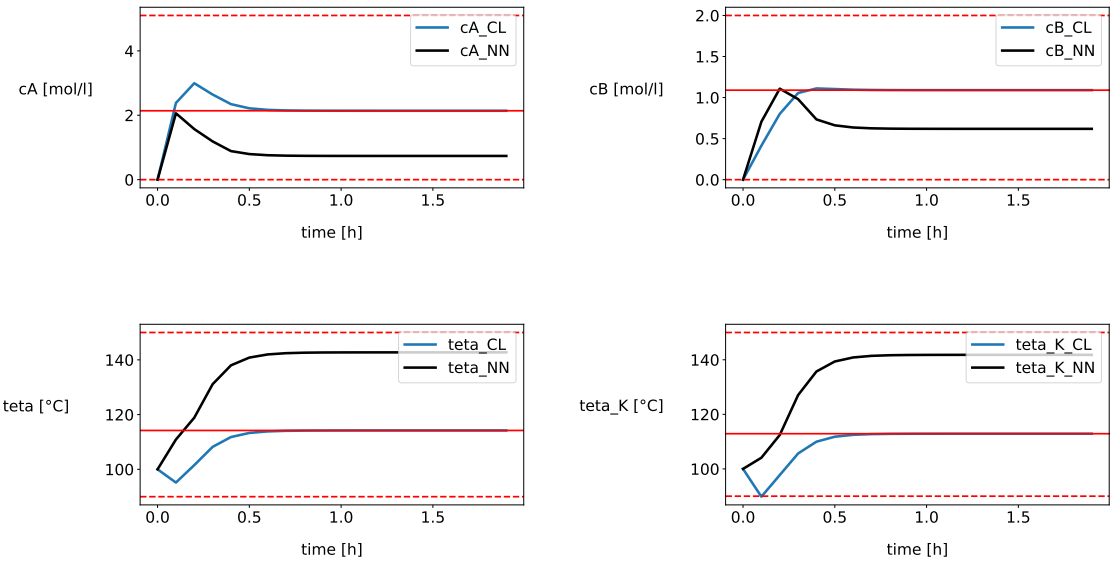

Figure 3.10: Nominal closed-loop trajectories for constant parameters and no uncertainties in blue and resulting trajectories of one-step NN in black

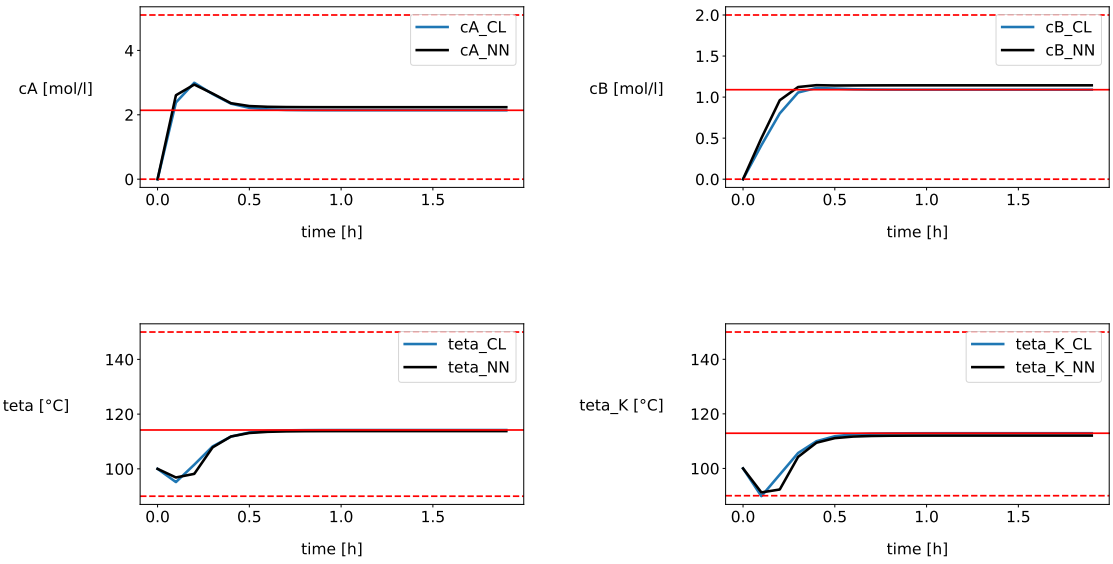

Figure 3.11: Nominal closed-loop trajectories for constant parameters and no uncertainties in blue and resulting trajectories of multi-step NN in black

In summary, the decomposition function derived from the trajectory-trained monotone NN provides especially tighter lower bounds in many regions than the relaxed analytical counterpart and encompasses nearly all trajectory realizations. Therefore, this approach can be a promising alternative when analytical methods via interval arithmetics result in overly conservative bounds.

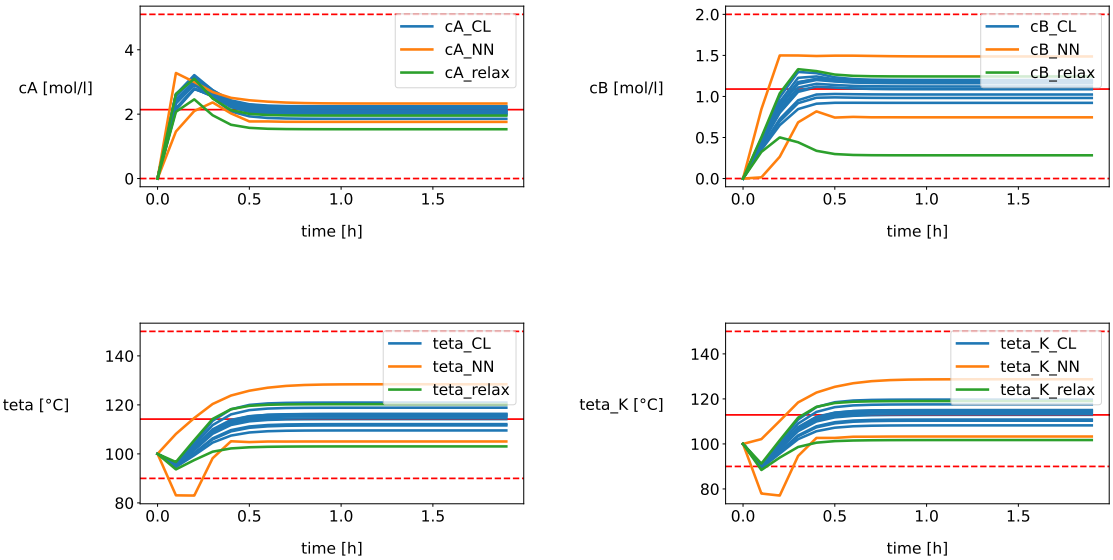

Figure 3.12: Illustration of the the tightness of the bounding of the relaxed decomposition function shown in green, the decompositionfunction from the monotone NN shown in orange and 10 nominal closed-loop trajectories with aribtrary realizations of the 11 uncertainty parameters including the worst case scenarios

Chapter 4

Conclusion

The focus of this work initially lies in the comparison of feedback-based MPC set-partitioning strategies. Building on the Full Partitioning MPC approach introduced in [14], [15], two additional feedback-based strategies are developed that provide increased flexibility in the choice of partitioning schemes.

Alternating Constant Partitioning MPC allows the search-tree-like-partitioning scheme to vary between prediction steps. Both the ordering of the partitioned dimensions and the number of partitions per dimension can be chosen freely, allowing feedback to be distributed across state dimensions in a time-varying fashion. *Alternating Variable Partitioning MPC* further generalizes this concept by also allowing the total number of subregions to vary across prediction steps. As a result, the resolution of the reachable sets—and thus the amount of feedback—can be periodically increased or decreased.

Recursive feasibility for all three strategies is discussed in detail in Section 2.7. The computation of reachable sets can be done efficiently with monotonicity or mixed-monotonicity. All three set-partitioning MPC strategies are first applied to a two-dimensional monotone double integrator. Here, the uncertain parameter is kept constant at its nominal value to isolate the effects of ordering and N_s and to compare the three partitioning approaches with each other.

Although all three strategies achieve the tracking objective and make J_{CL} nearly stagnate, Alternating Constant Partitioning MPC and Alternating Variable Partitioning MPC yield the lowest values, indicating superior tracking performance. The corresponding closed-loop trajectories for control inputs and states of the double integrator are shown in the appendix. These reveal that the alternating strategies tend to introduce oscillations in states and in the control inputs as a result of their time-varying partitioning schemes. However, these oscillations can be mitigated by increasing the number of subregions or by partitioning more dimensions per prediction step.

At all the alternating partitioning strategies are capable of bringing the states and control inputs even closer to the desired setpoints as Full Partitioning MPC. This comes at the cost of an increased computation time per MPC iteration compared to Full Partitioning MPC. Hence, alternating partitioning strategies offer a trade-off between tracking performance and increased computation time.

The three feedback-based MPC set-partitioning strategies are then extended to a more complex, non-monotone CSTR to validate the findings on a more realistic system. Similar trends are observed: a better tracking performance that brings the states of the CSTR closer to the desired setpoints comes at the cost of increased computation time per iteration. Again, oscillations in the control inputs are seen for smaller values of N_s, especially for the alternating variable partitioning, but can be mitigated with a higher N_s. For constant parameter realizations, satisfactory performance is observed for all three strategies: the tracking objectives are met, and the closed-loop trajectories approach the desired setpoints. Furthermore, operations close to constraints underline that the alternating partitioning strategies achieve a robust closed-loop behavior that can be validated through simulations with a CSTR where the nominal MPC fails and violates constraints.

In this work, the alternating partitioning schemes are implemented over only two consecutive prediction steps. Further research could investigate the effect of multistep alternating schemes on MPC performance.

The CSTR, being a non-monotone system, permits the construction of an analytical decomposition function based on interval arithmetics as shown in [15]. However, it is observed that this approach tends to produce overly conservative overapproximations, particularly for the temperature variables ϑ and ϑ_K, making it unsuitable for the feedback-based set partitioning strategies discussed here. As a workaround, a relaxed decomposition function is initially employed. This is obtained by replacing ϑ_{min} with ϑ_{max} in 2.52 and 2.54, yielding smaller values for $c_{A,\text{max}}$ and $c_{B,\text{max}}$, such that also ϑ_{max} and $\vartheta_{K,max}$ become smaller and hence enable more practical overapproximations.

Simulations of a the the partitioning strategies across 100 different parameter realizations confirm that partitioning strategies are functional and provide robust performance with the CSTR. However, the relaxed decomposition function does not offer formal bounding guarantees as does the analytical one.

Motivated by the need for tighter and more reliable bounds, especially near constraints, a new decomposition function is extracted from a neural network confined to being monotone with respect to inputs x and p. This monotone neural network is trained on nominal trajectories on the CSTR under various parameter realizations. Training must target entire trajectories rather than isolated single-step predictions, as cumulative prediction errors would otherwise corrupt future steps.

The trajectory-trained monotone neural network is shown to closely replicate nominal trajectories with minor deviations. Leveraging the monotonicity property and the sign structure of the network weights, a decomposition function is extracted that can bound the nominal trajectory under diverse realizations of p. In fact, the lower bounds provided by this approach are often tighter than those of the relaxed interval-based function, which fails to contain many nominal trajectories.

Thus, generating a decomposition function from a monotone-constrained neural network emerges as a promising direction. Future work could aim to establish probabilistic guarantees for this method, ensuring that the decomposition function bounds trajectories with high statistical confidence [46]. Therefore, it should also be investigated which trajectories can be approximated well and which ones not. Furthermore,

exploring other network architectures that enforce monotonicity, beyond the simple constraint of positive weights, may yield further improvements. The monotone neural network consists of a relatively large architecture with six layers, each containing 128 neurons. It remains to be investigated whether a smaller neural network can also achieve satisfactory performance, with the goal of integrating the corresponding decomposition function into a RMPC loop.

Appendix A

plots and figures

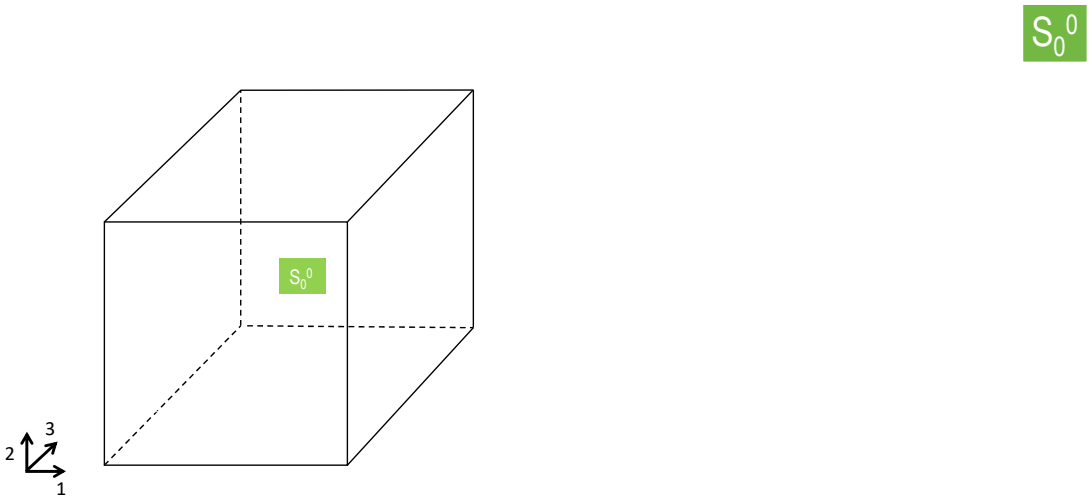

Figure A.1: 3D hyperrectangle representing initial set $\mathcal{S}_0^0$ [15]

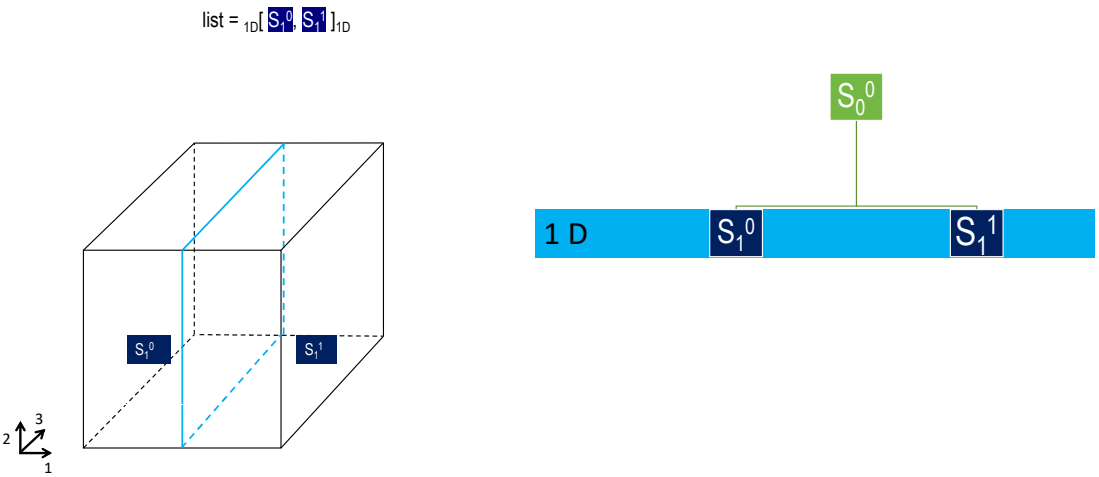

Figure A.2: search-tree-like partitioning in level1 with sets $\mathcal{S}_1^1$ and $\mathcal{S}_1^2$ [15]

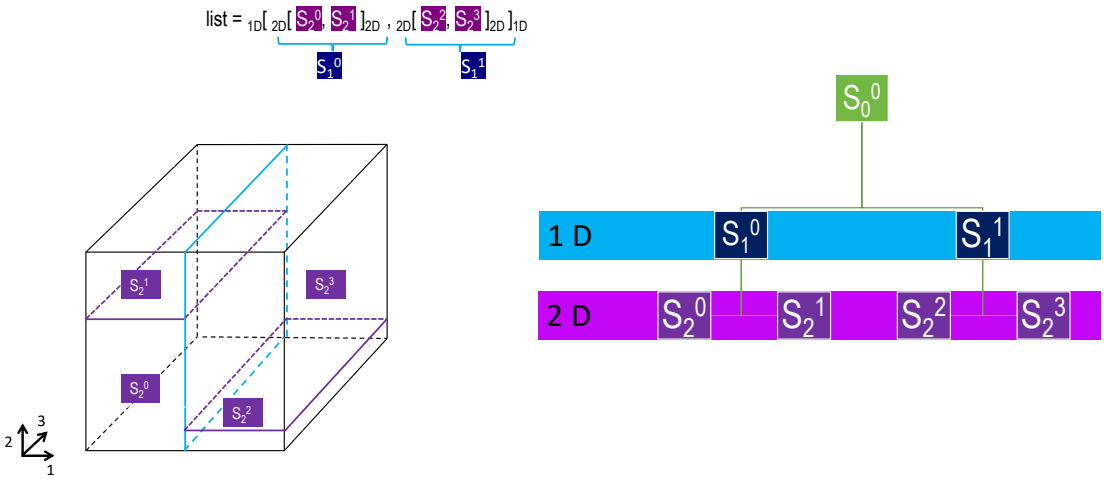

Figure A.3: search-tree-like partitioning in level2 with sets $\mathcal{S}_2^1, \mathcal{S}_2^2, \mathcal{S}_2^3, \mathcal{S}_2^4$ [15]

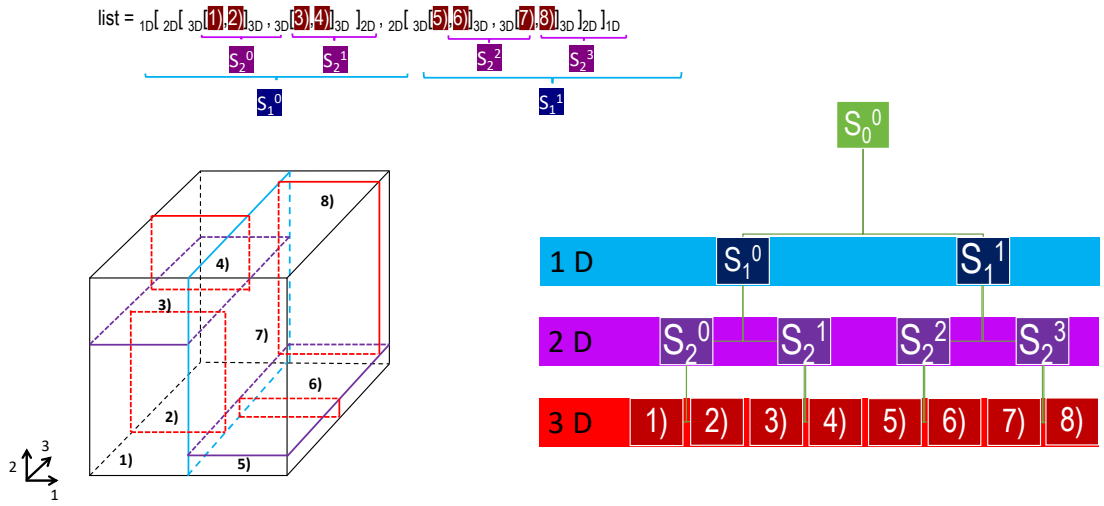

Figure A.4: search-tree-like partitioning in level3 with sets $\mathcal{S}_3^1, \mathcal{S}_3^2, \mathcal{S}_3^3, \mathcal{S}_3^4, \mathcal{S}_3^5, \mathcal{S}_3^6, \mathcal{S}_3^7, \mathcal{S}_3^8$ [15]

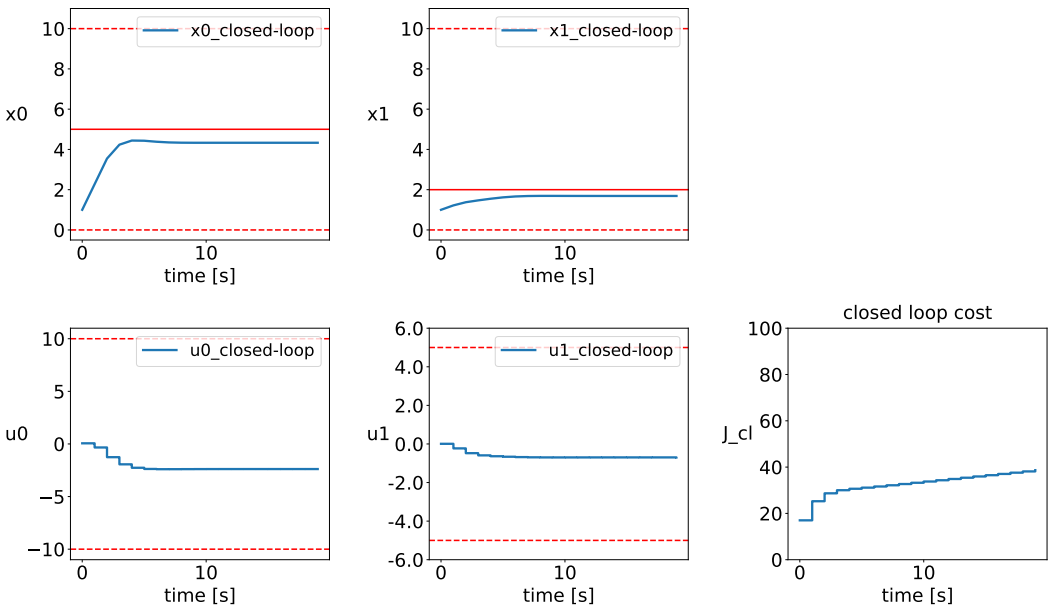

Figure A.5: Full Partitioning MPC for double integrator with no uncertainty for N_s = 9 and ordering [0,1],[0,1]

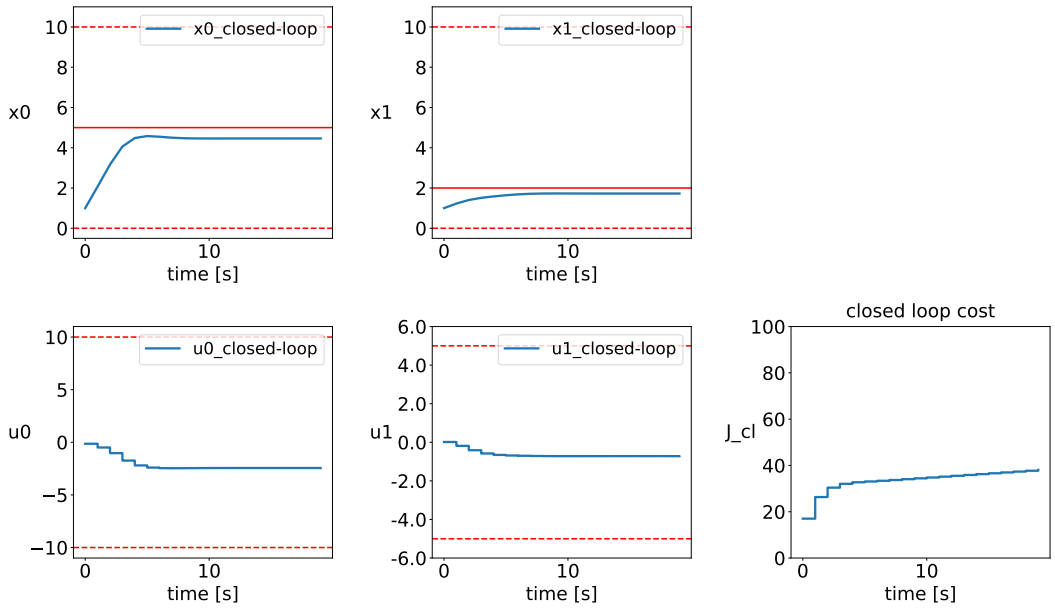

Figure A.6: Full Partitioning MPC for double integrator with no uncertainty for N_s = 16 and ordering [0,1],[0,1]

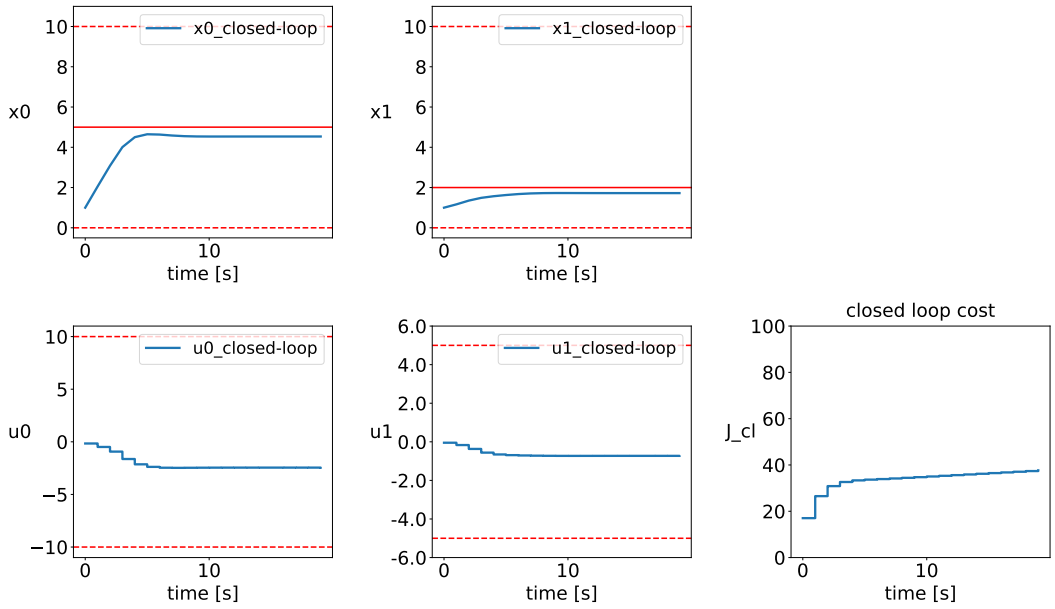

Figure A.7: Full Partitioning MPC for double integrator with no uncertainty for N_s = 24 and ordering [0,1],[0,1]

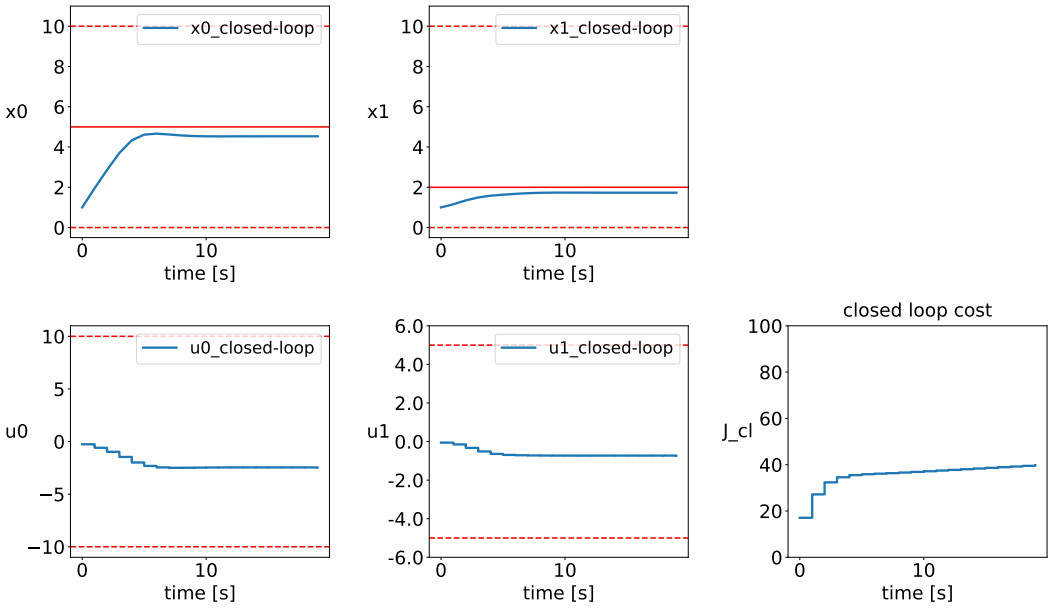

Figure A.8: Full Partitioning MPC for double integrator with no uncertainty for N_s = 30 and ordering [0,1],[0,1]

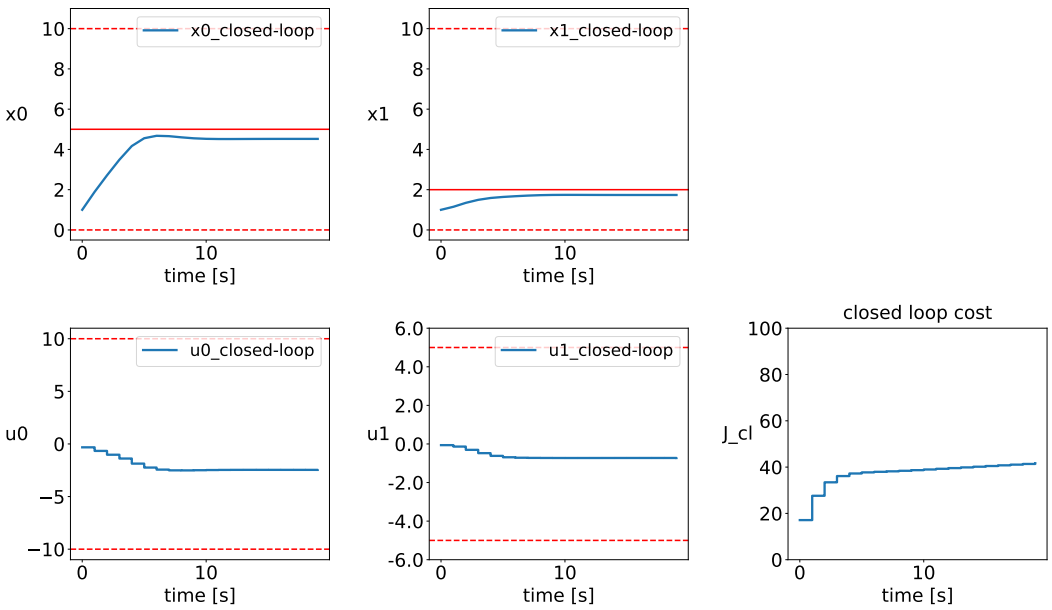

Figure A.9: Full Partitioning MPC for double integrator with no uncertainty for N_s = 36 and ordering [0,1],[0,1]

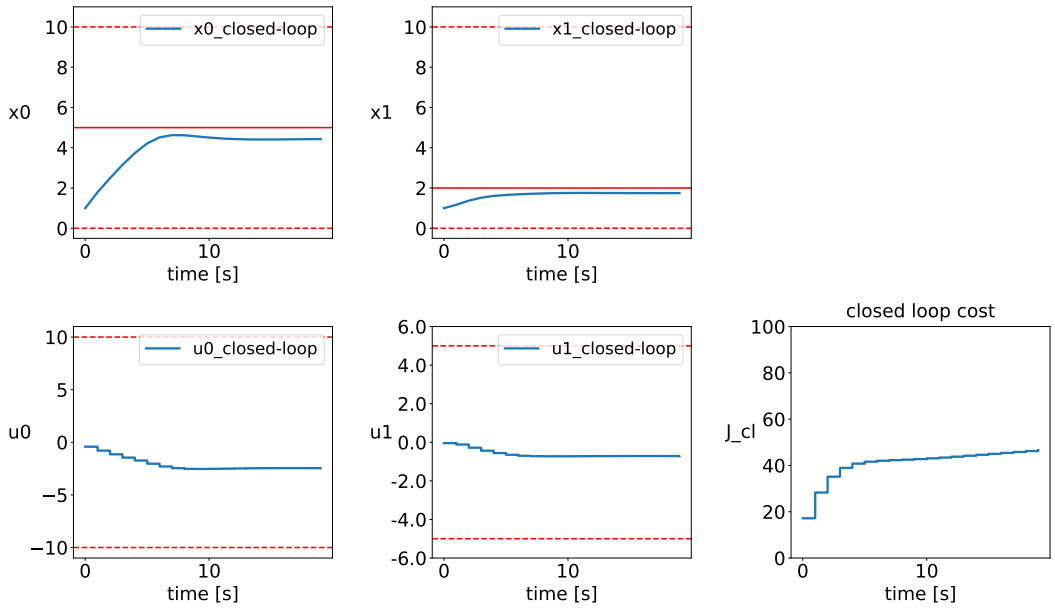

Figure A.10: Full Partitioning MPC for double integrator with no uncertainty for N_s = 45 and ordering [0,1],[0,1]

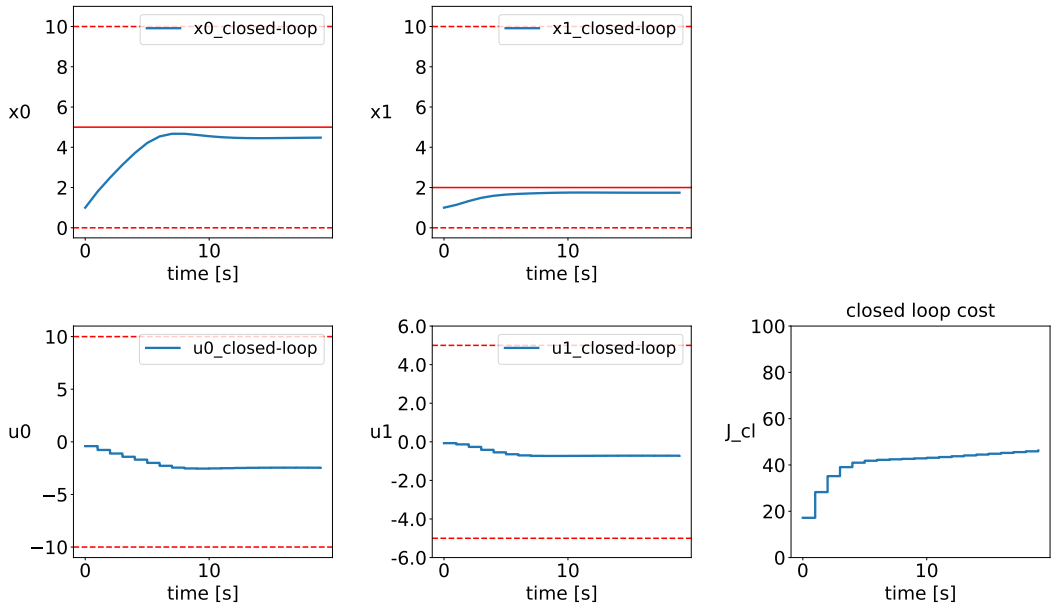

Figure A.11: Full Partitioning MPC for double integrator with no uncertainty for N_s = 54 and ordering [0,1],[0,1]

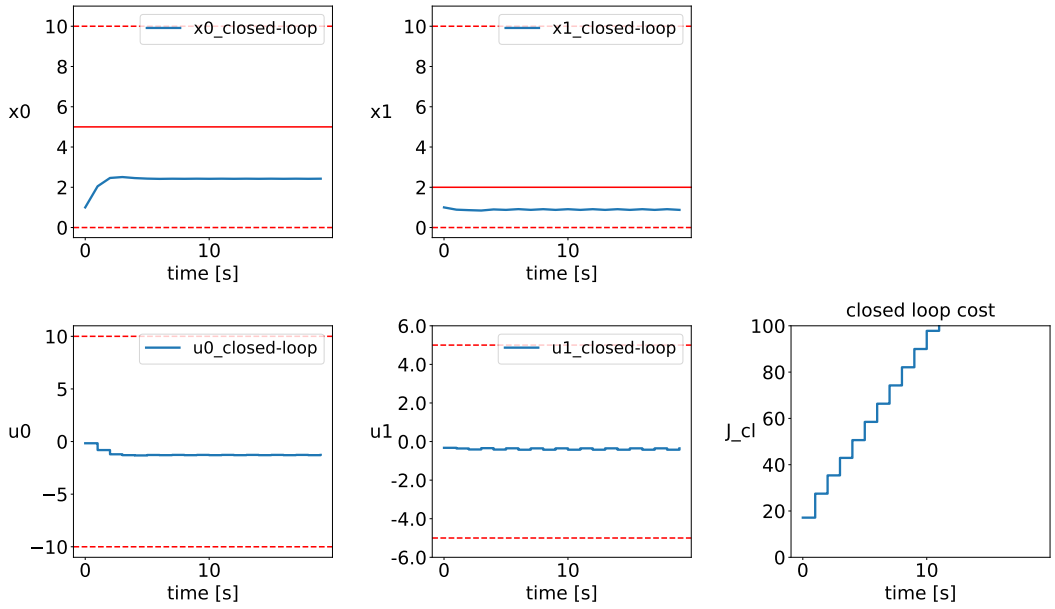

Figure A.12: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 9 and ordering [0,1],[0,1]

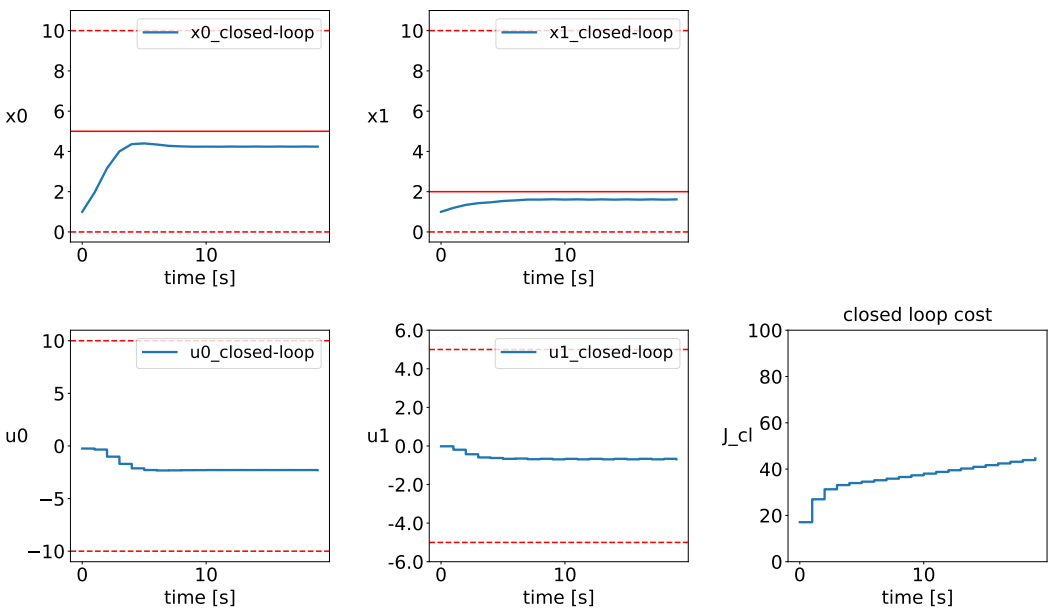

Figure A.13: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 16 and ordering [0,1],[0,1]

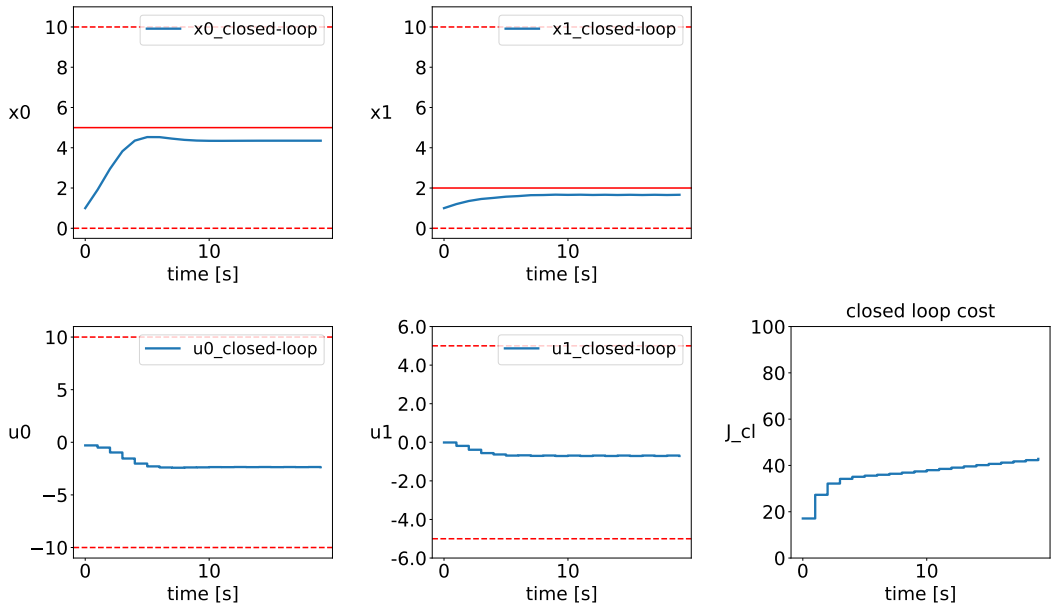

Figure A.14: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 24 and ordering [0,1],[0,1]

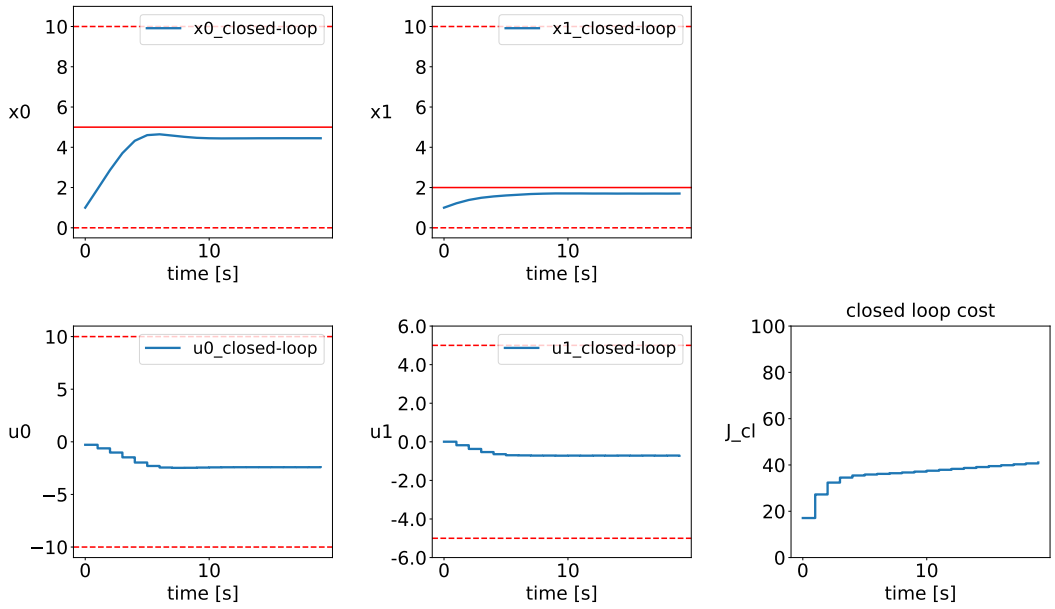

Figure A.15: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 30 and ordering [0,1],[0,1]

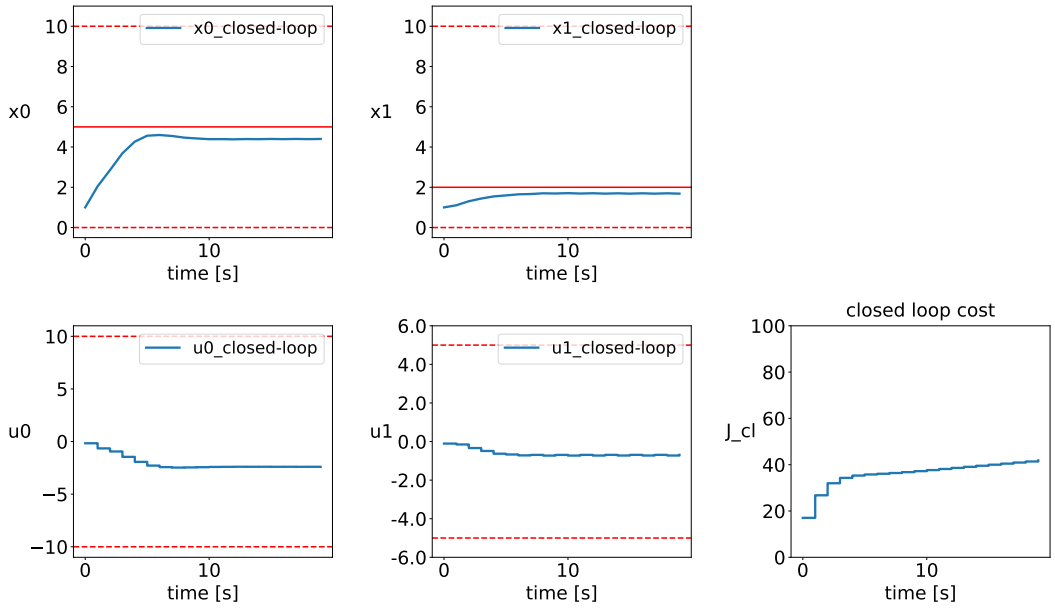

Figure A.16: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 36 and ordering [0,1],[0,1]

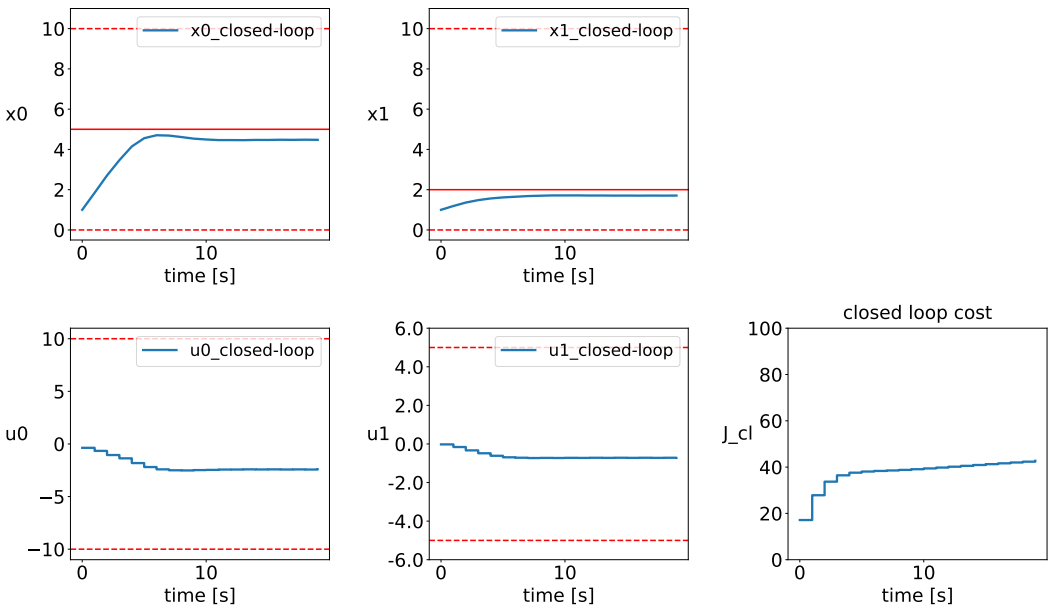

Figure A.17: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 45 and ordering [0,1],[0,1]

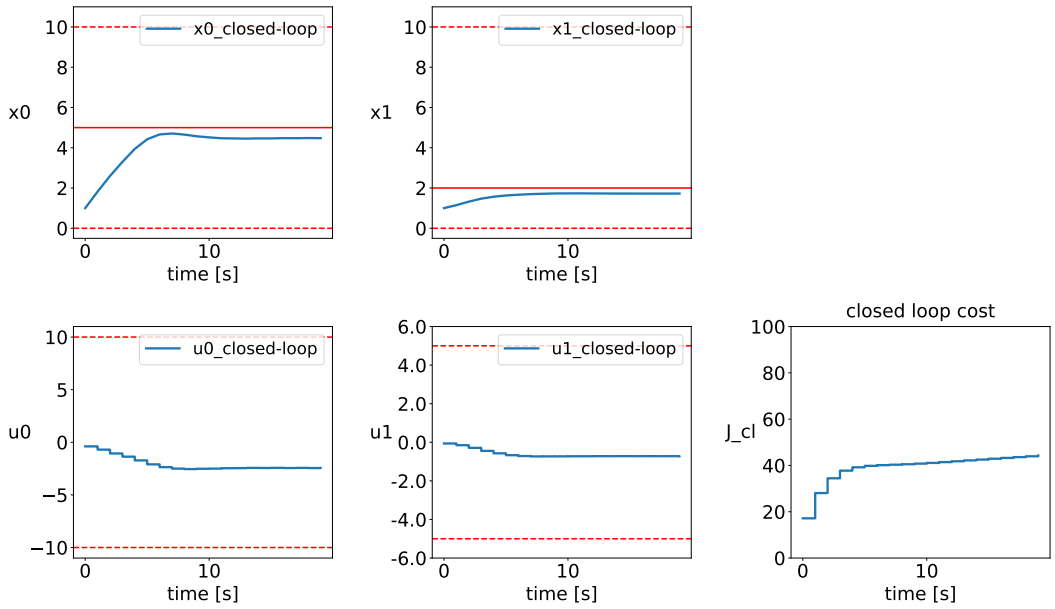

Figure A.18: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 54 and ordering [0,1],[0,1]

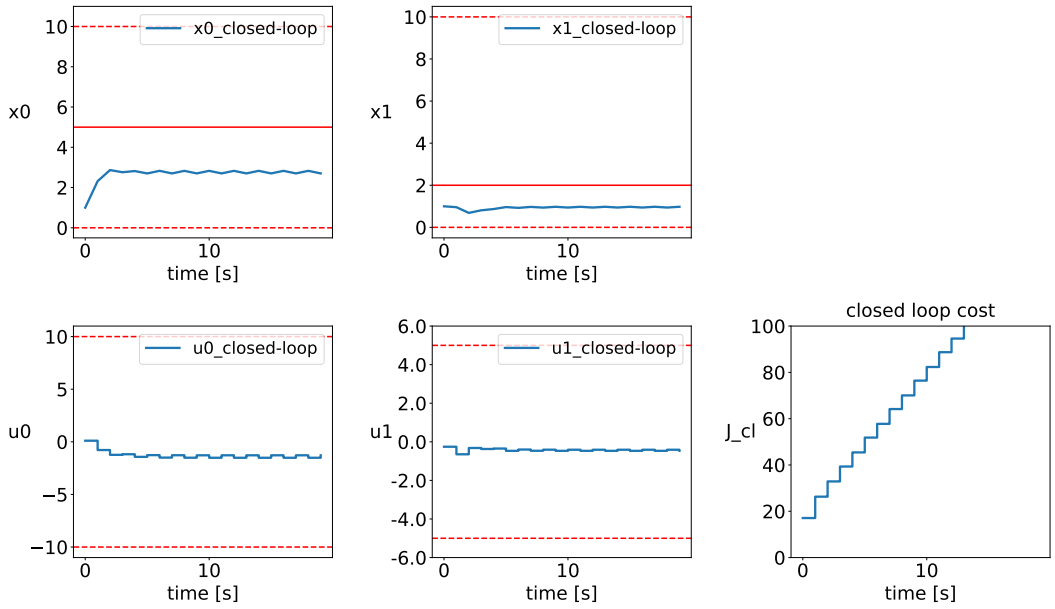

Figure A.19: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 9 and ordering [0,1],[0,1]

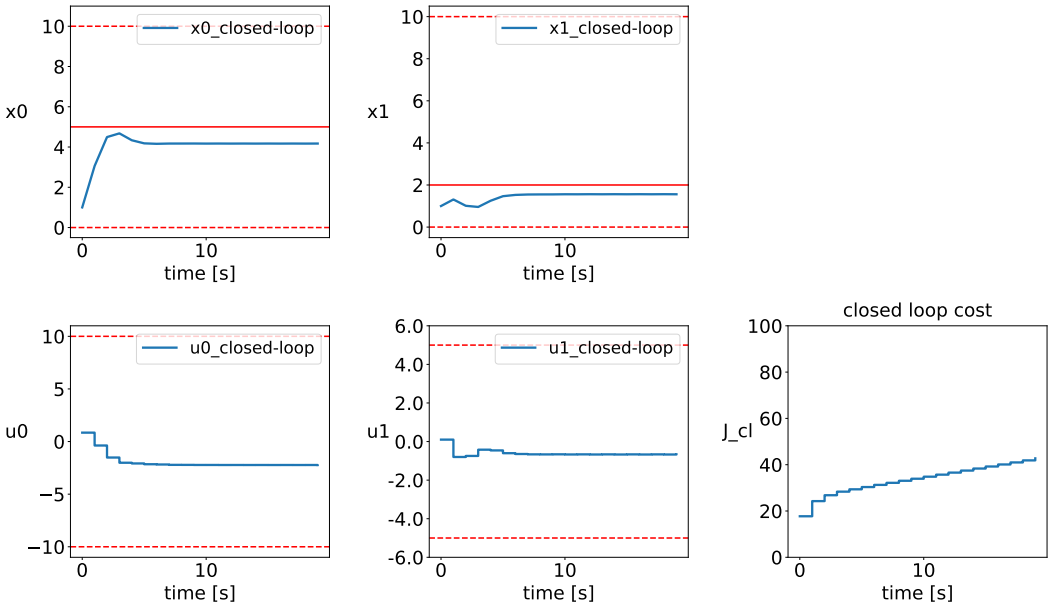

Figure A.20: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 16 and ordering [0,1],[0,1]

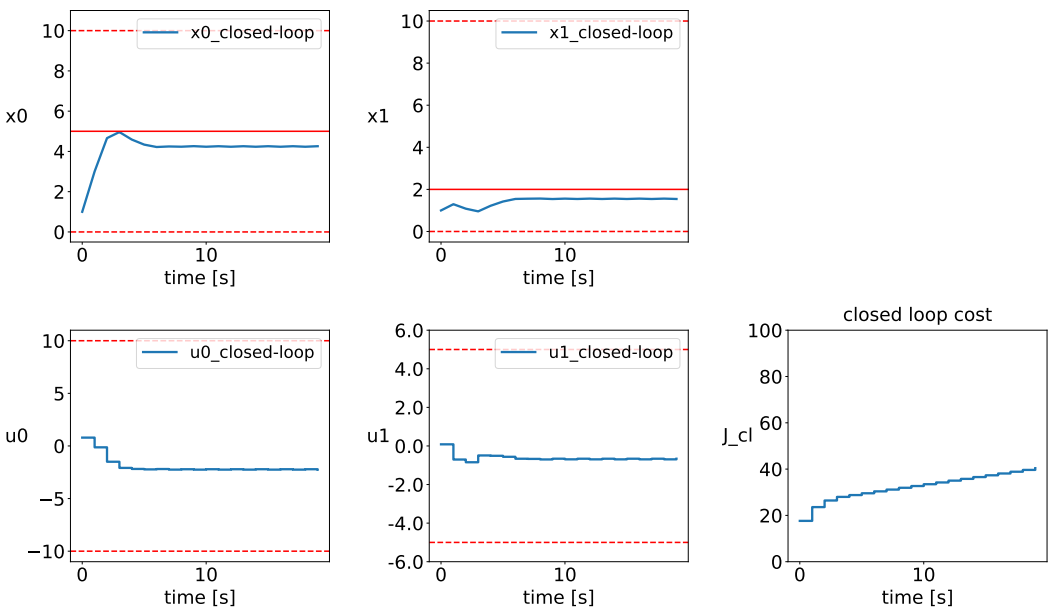

Figure A.21: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 24 and ordering [0,1],[0,1]

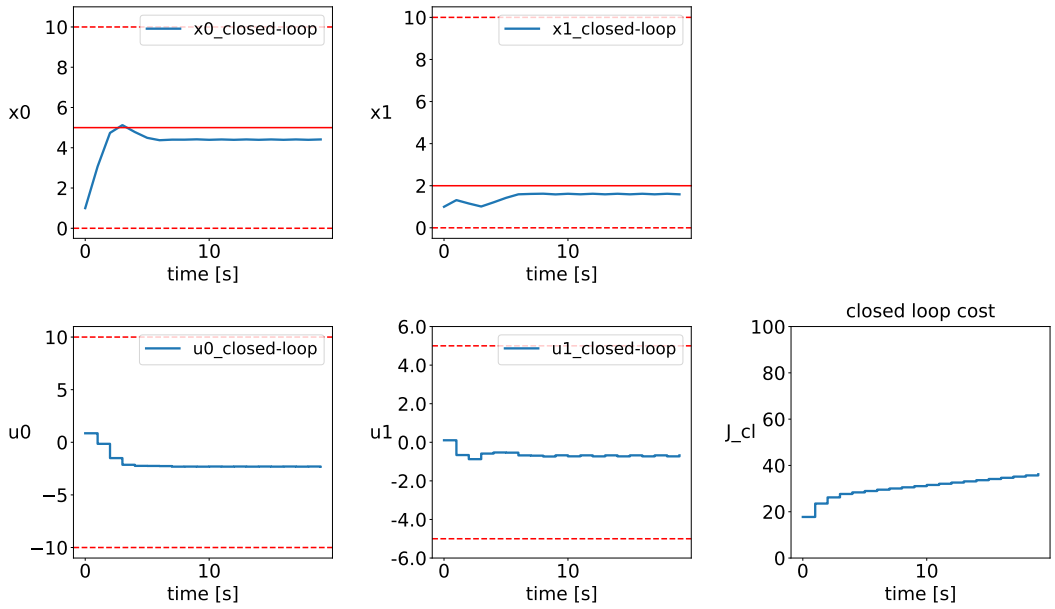

Figure A.22: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 30 and ordering [0,1],[0,1]

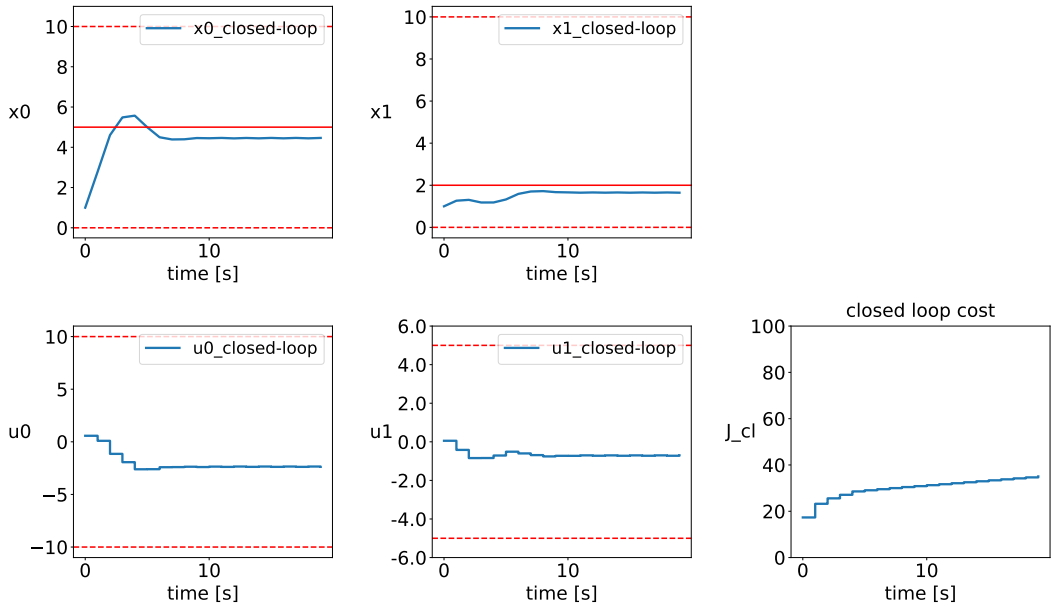

Figure A.23: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 36 and ordering [0,1],[0,1]

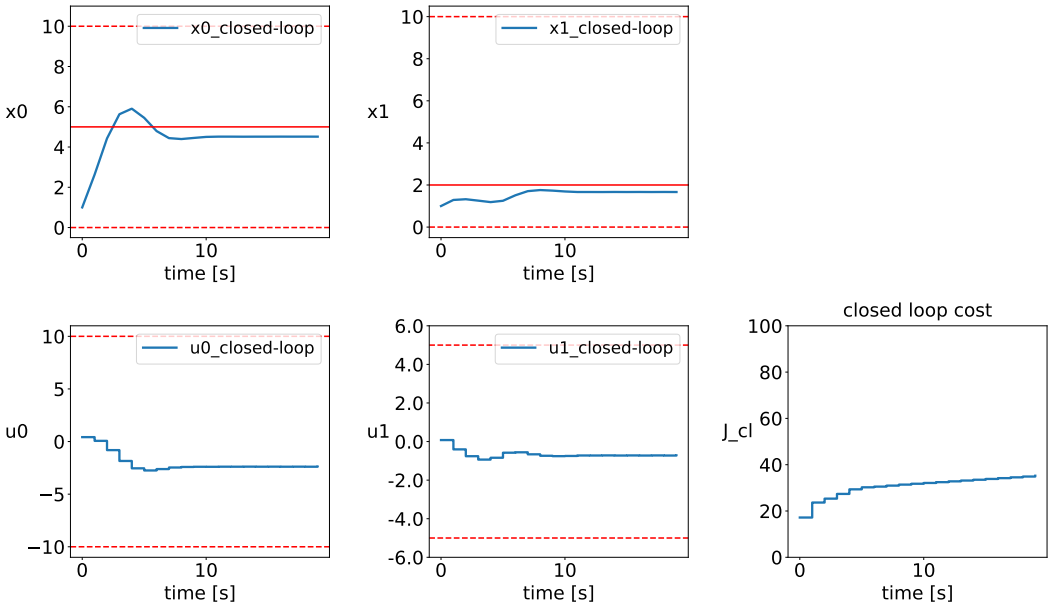

Figure A.24: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 45 and ordering [0,1],[0,1]

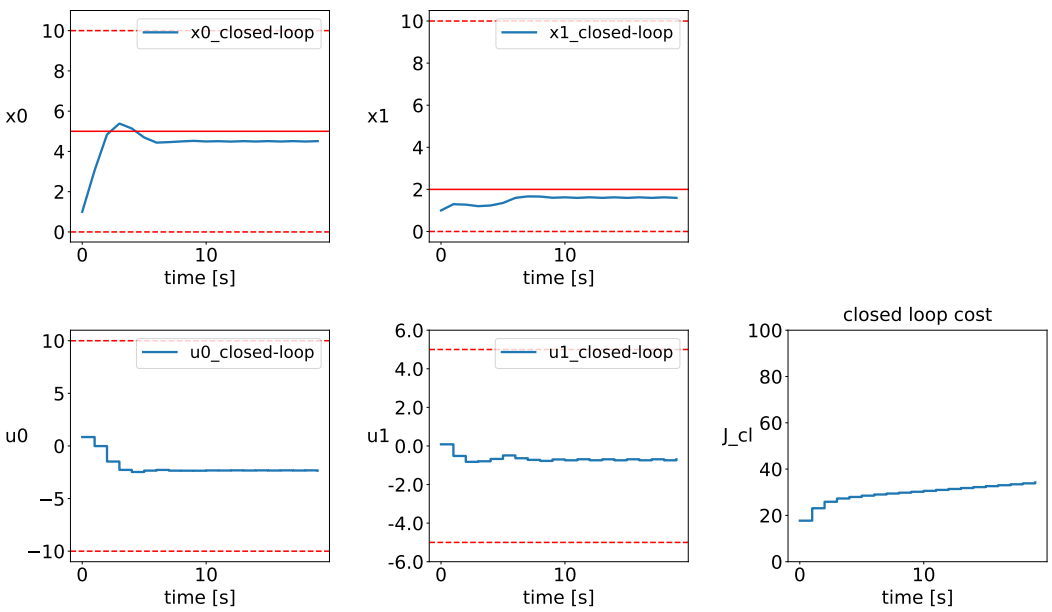

Figure A.25: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 54 and ordering [0,1],[0,1]

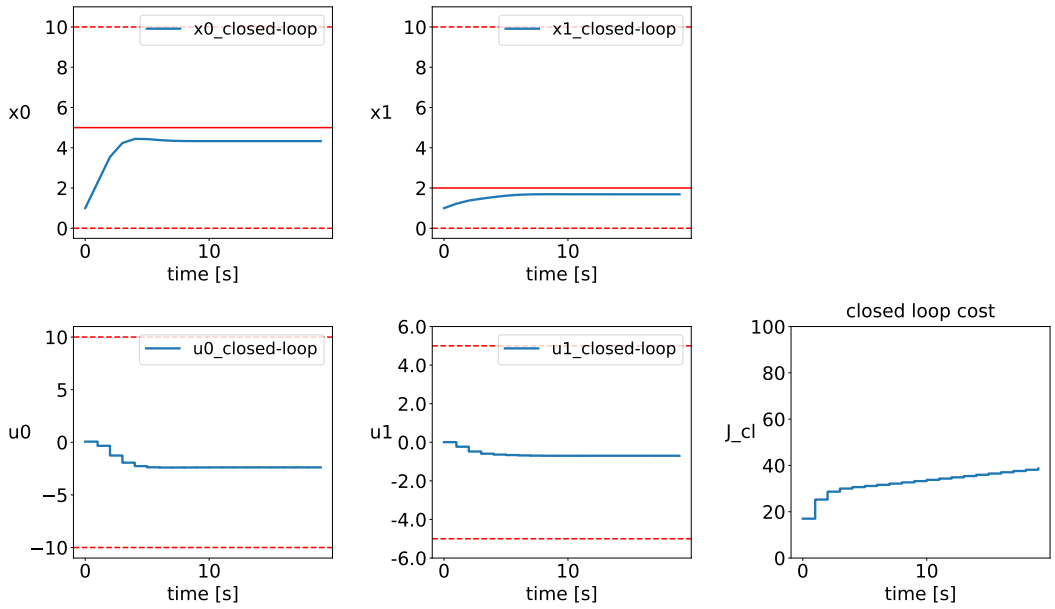

Figure A.26: Full Partitioning MPC for double integrator with no uncertainty for N_s = 9 and ordering [0,1],[1,0]

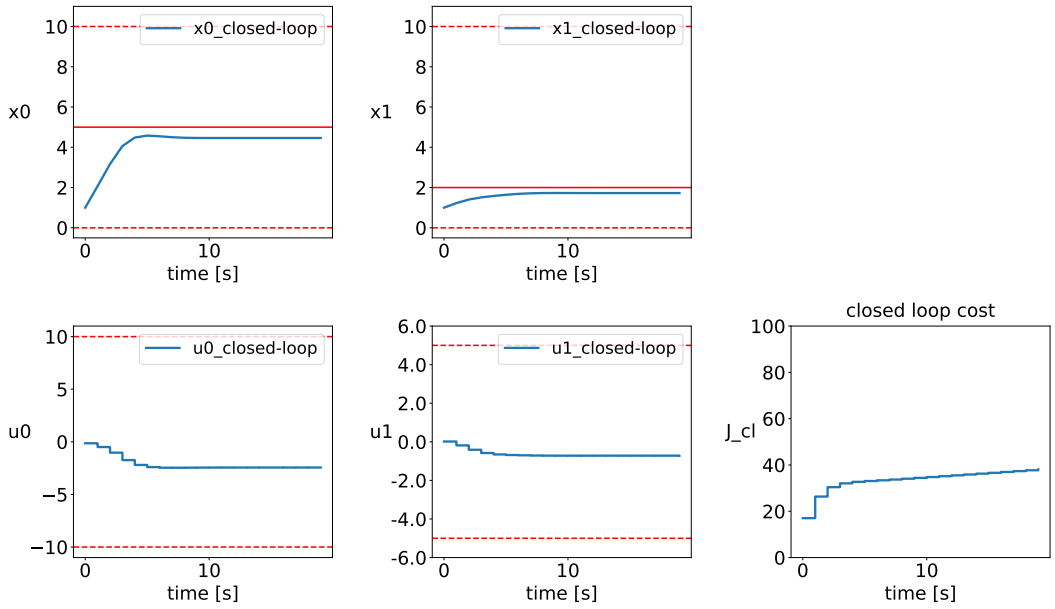

Figure A.27: Full Partitioning MPC for double integrator with no uncertainty for N_s = 16 and ordering [0,1],[1,0]

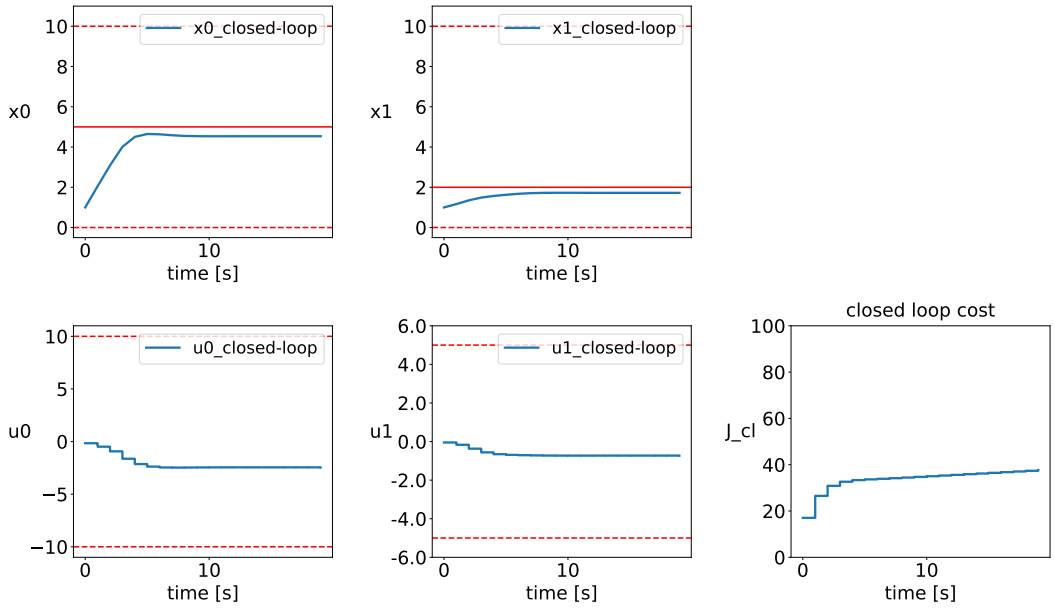

Figure A.28: Full Partitioning MPC for double integrator with no uncertainty for N_s = 24 and ordering [0,1],[1,0]

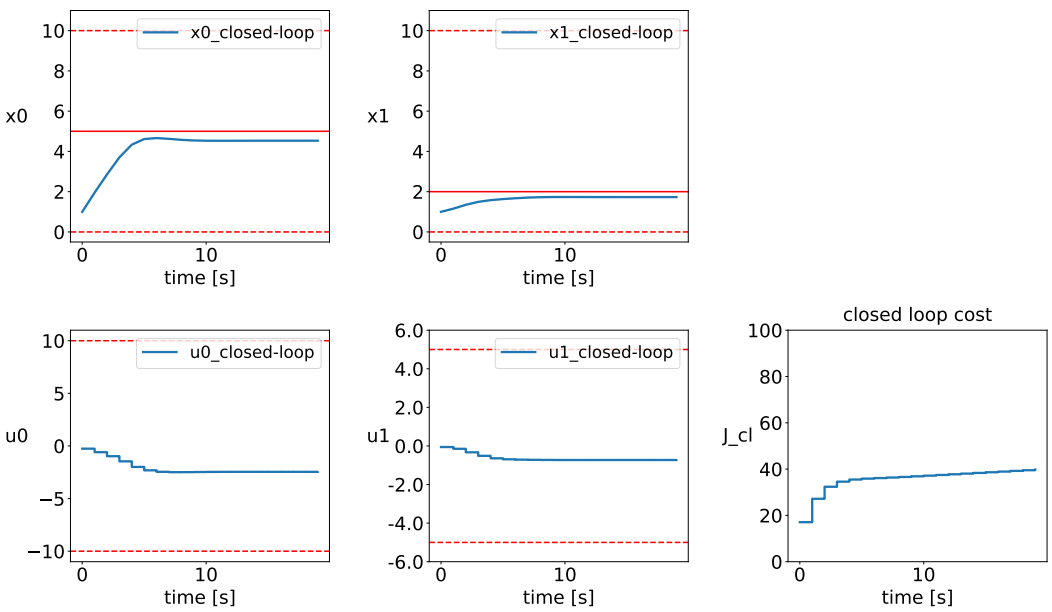

Figure A.29: Full Partitioning MPC for double integrator with no uncertainty for N_s = 30 and ordering [0,1],[1,0]

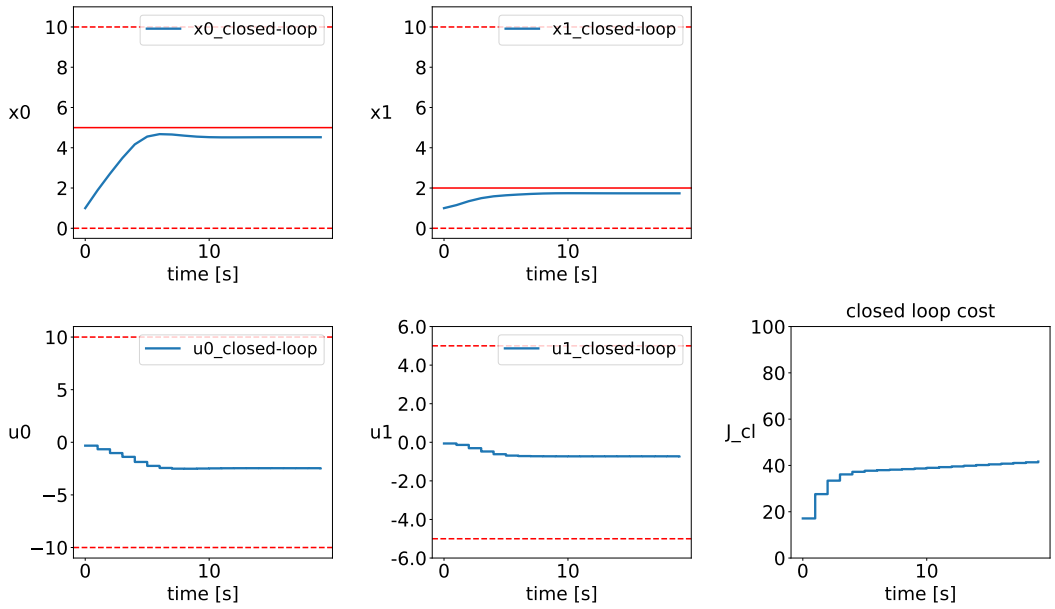

Figure A.30: Full Partitioning MPC for double integrator with no uncertainty for N_s = 36 and ordering [0,1],[1,0]

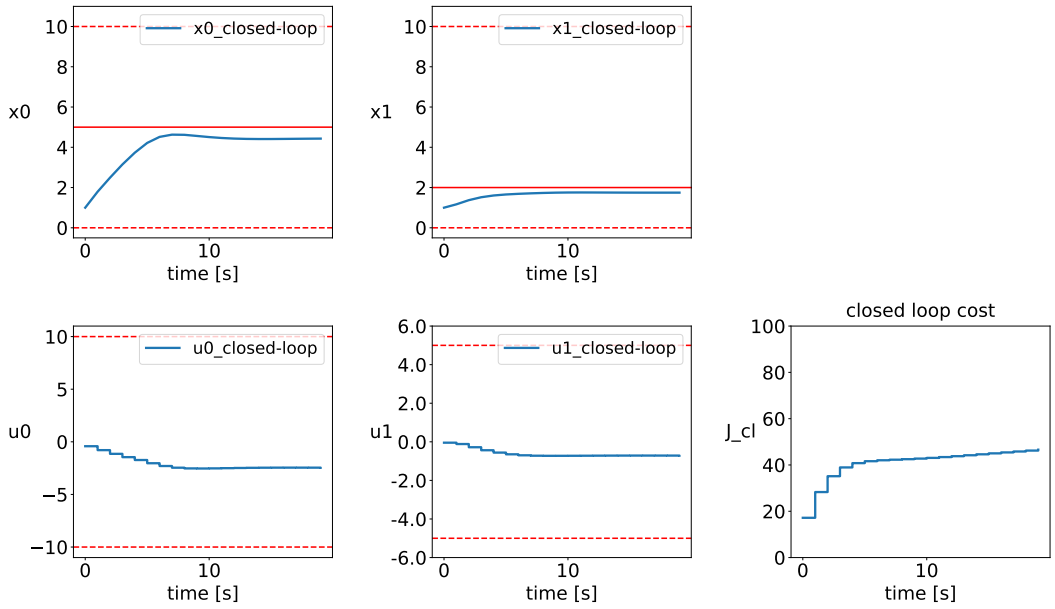

Figure A.31: Full Partitioning MPC for double integrator with no uncertainty for N_s = 45 and ordering [0,1],[1,0]

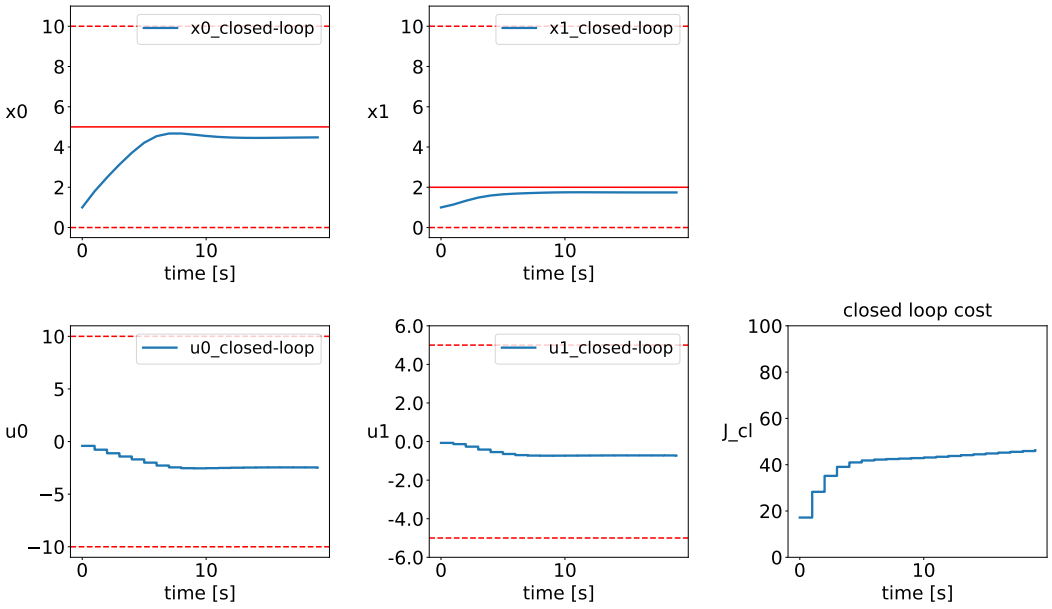

Figure A.32: Full Partitioning MPC for double integrator with no uncertainty for N_s = 54 and ordering [0,1],[1,0]

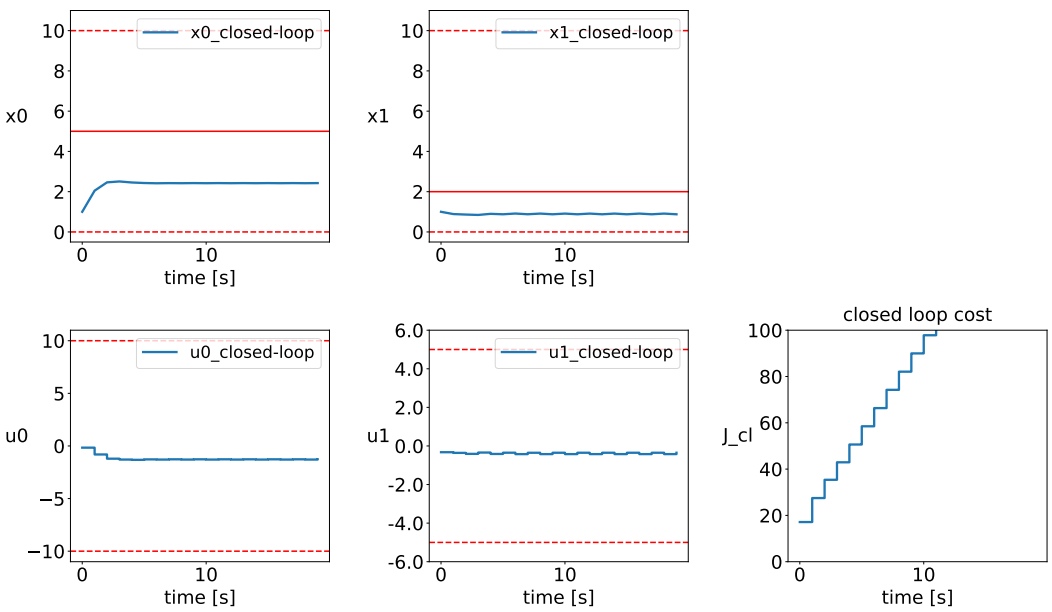

Figure A.33: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 9 and ordering [0,1],[1,0]

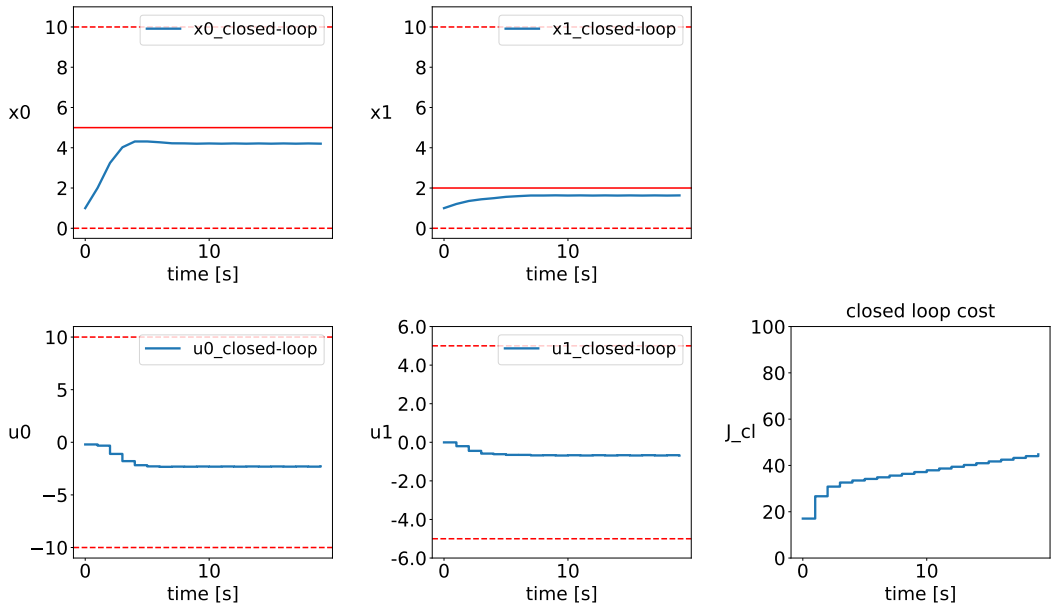

Figure A.34: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 16 and ordering [0,1],[1,0]

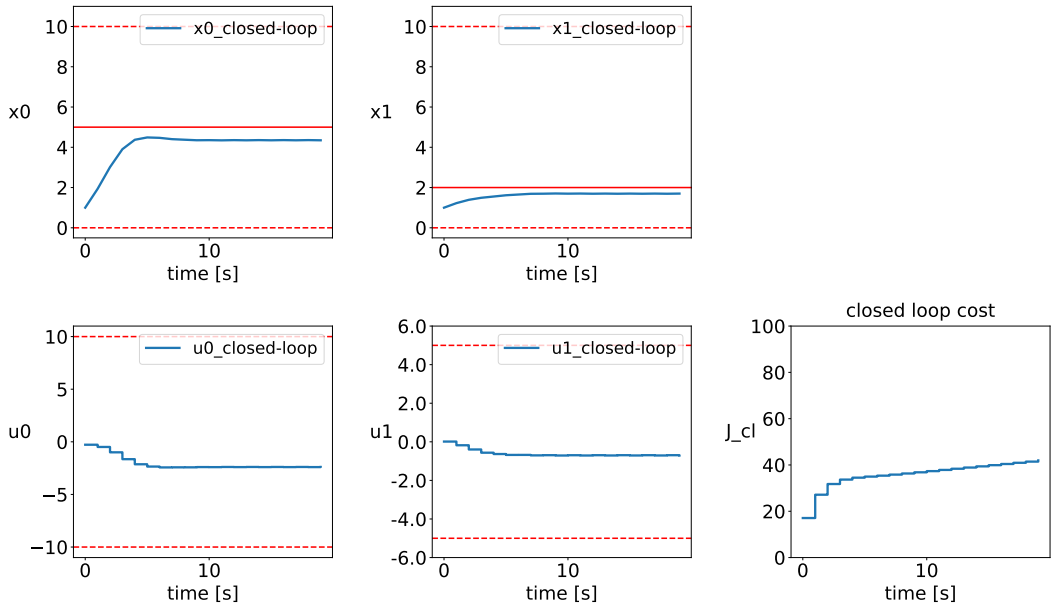

Figure A.35: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 24 and ordering [0,1],[1,0]

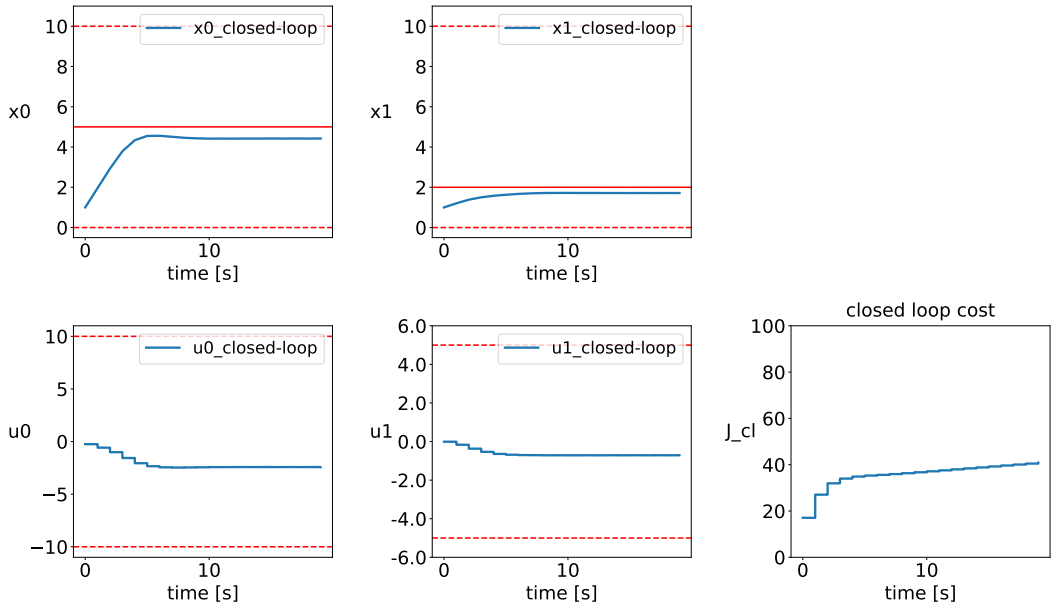

Figure A.36: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 30 and ordering [0,1],[1,0]

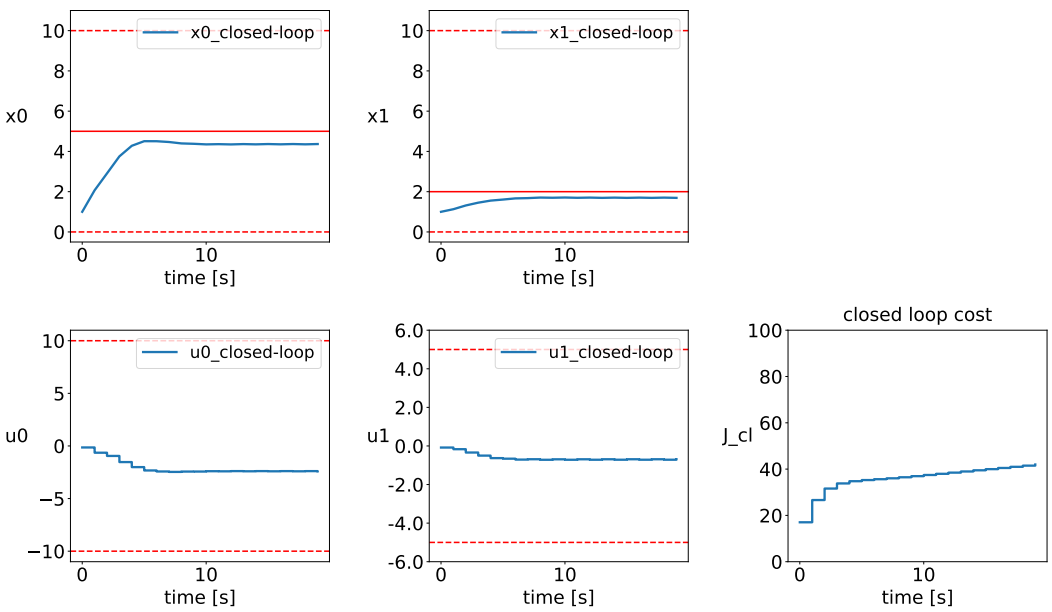

Figure A.37: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 36 and ordering [0,1],[1,0]

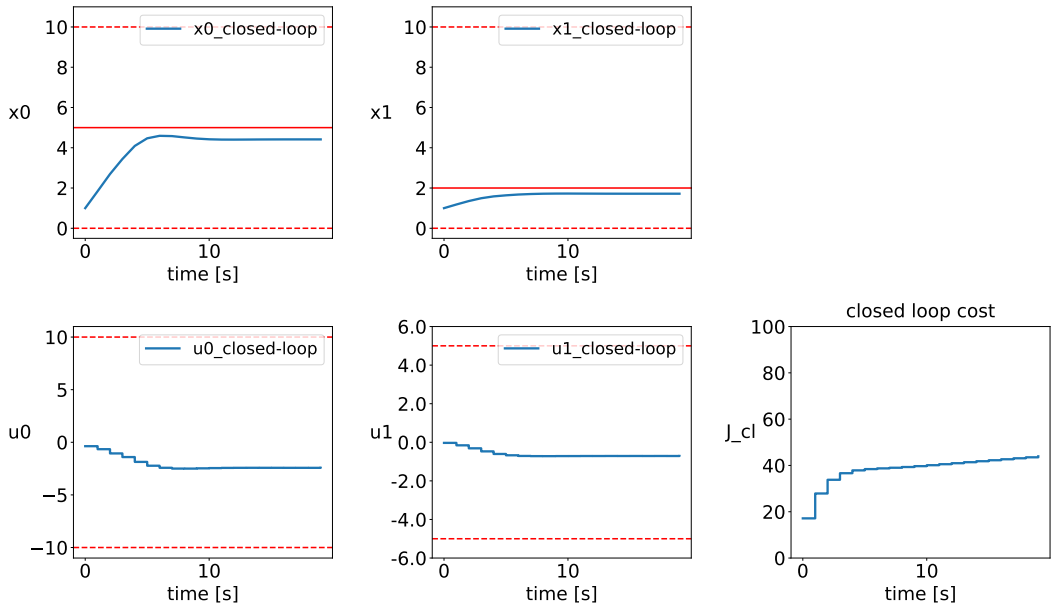

Figure A.38: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 45 and ordering [0,1],[1,0]

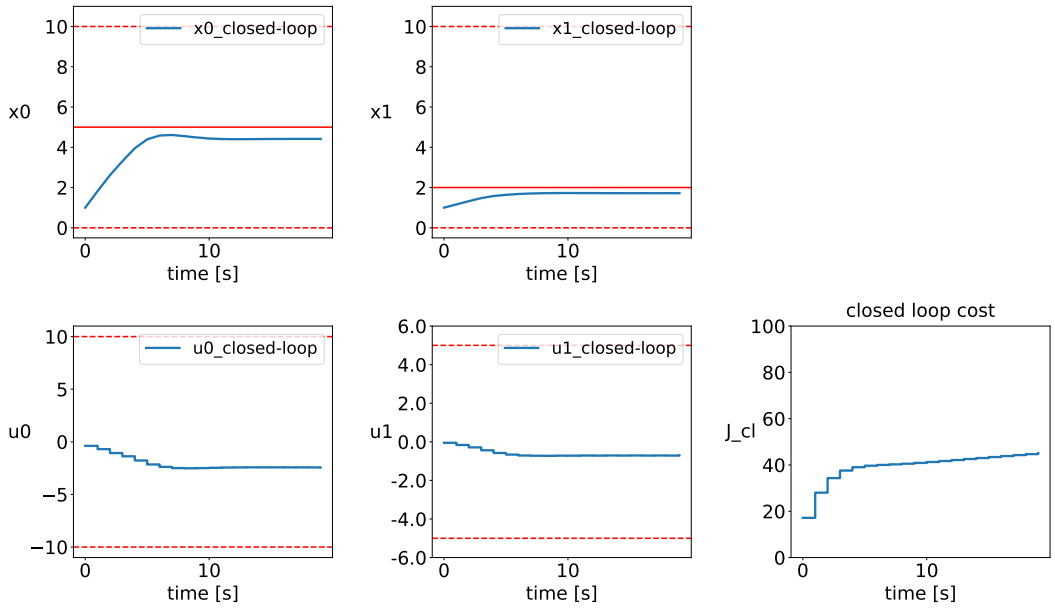

Figure A.39: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 54 and ordering [0,1],[1,0]

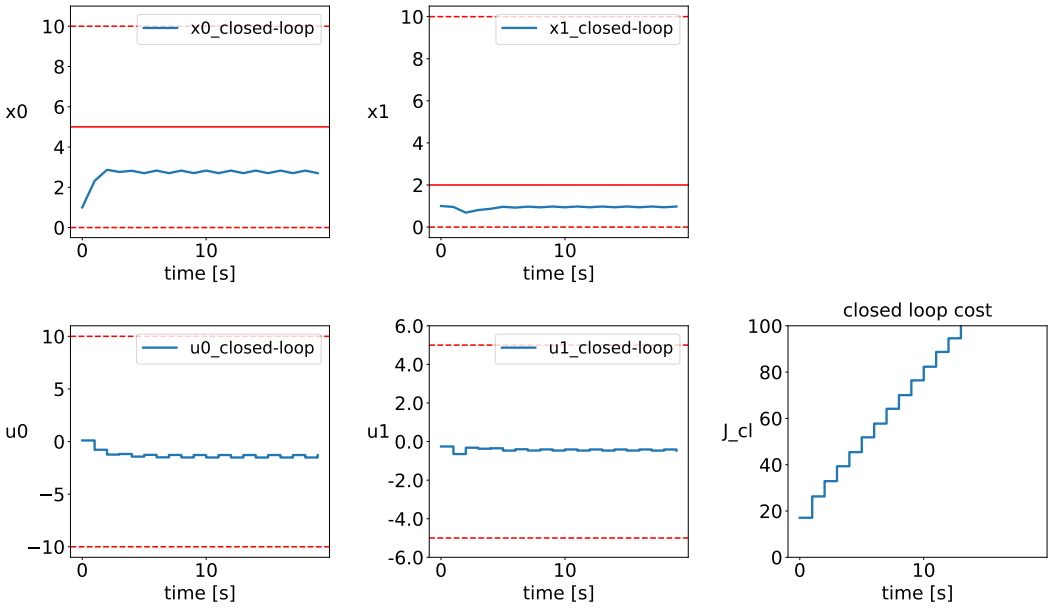

Figure A.40: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 9 and ordering [0,1],[1,0]

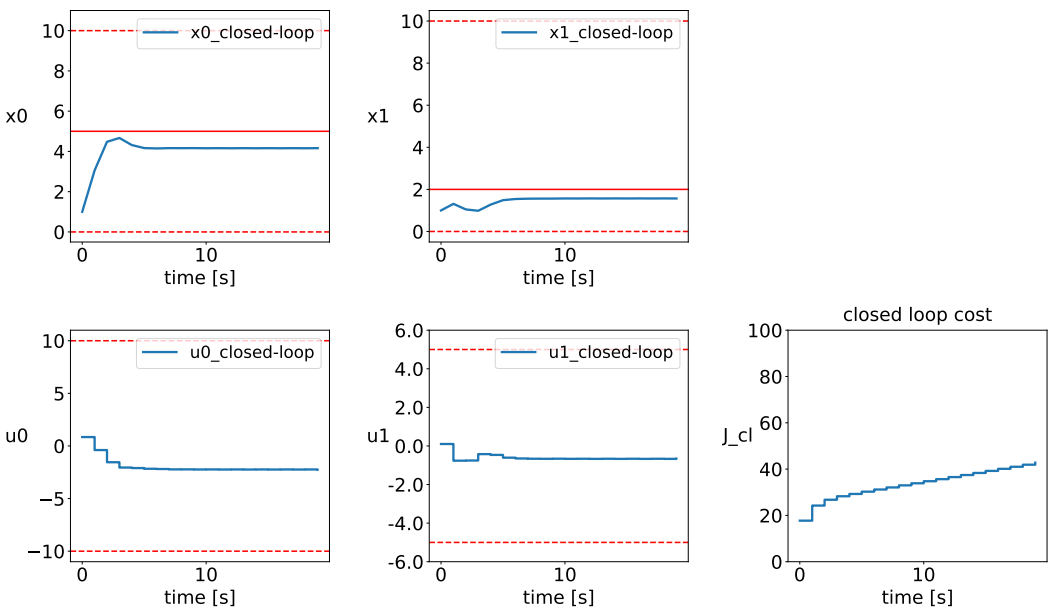

Figure A.41: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 16 and ordering [0,1],[1,0]

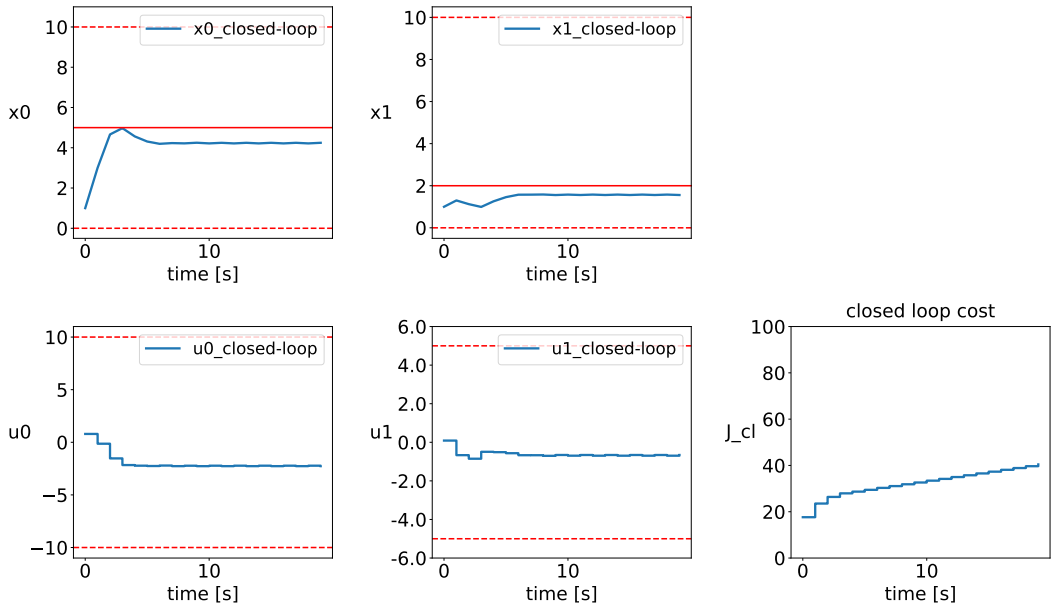

Figure A.42: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 24 and ordering [0,1],[1,0]

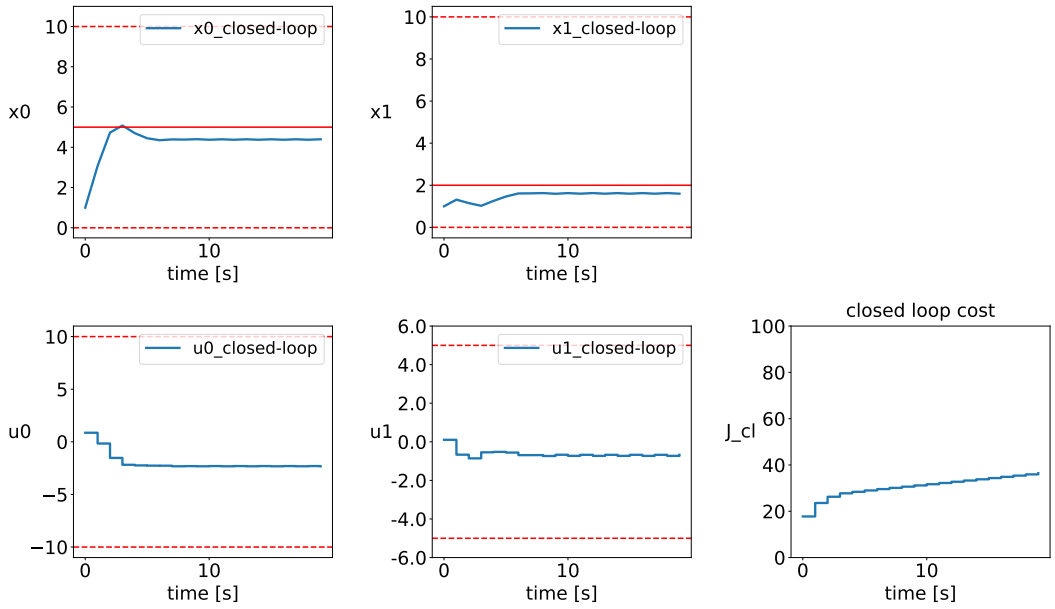

Figure A.43: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 30 and ordering [0,1],[1,0]

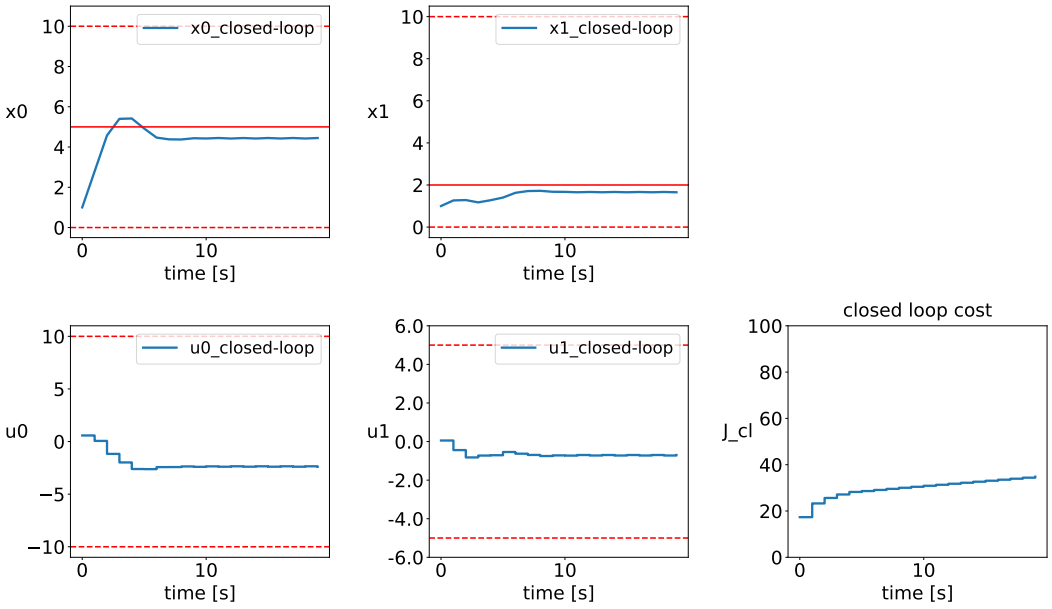

Figure A.44: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 36 and ordering [0,1],[1,0]

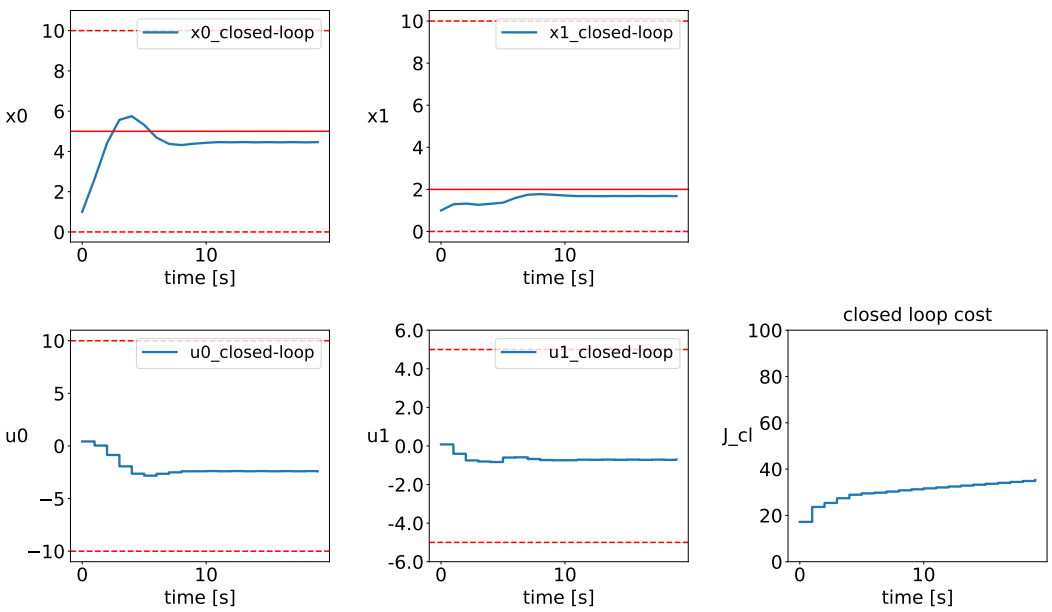

Figure A.45: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 45 and ordering [0,1],[[1,0]

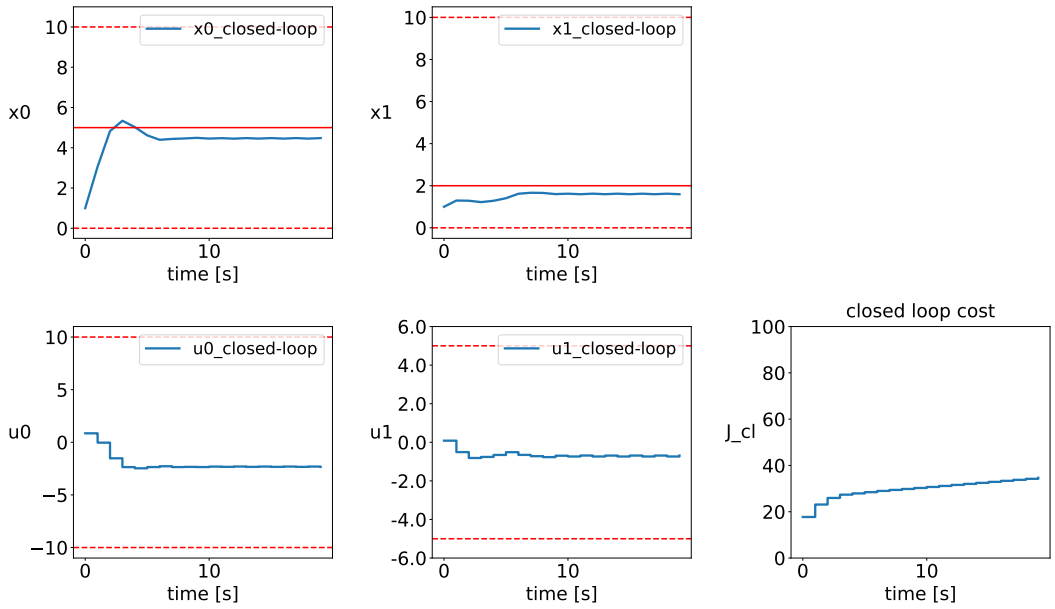

Figure A.46: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 54 and ordering [0,1],[1,0]

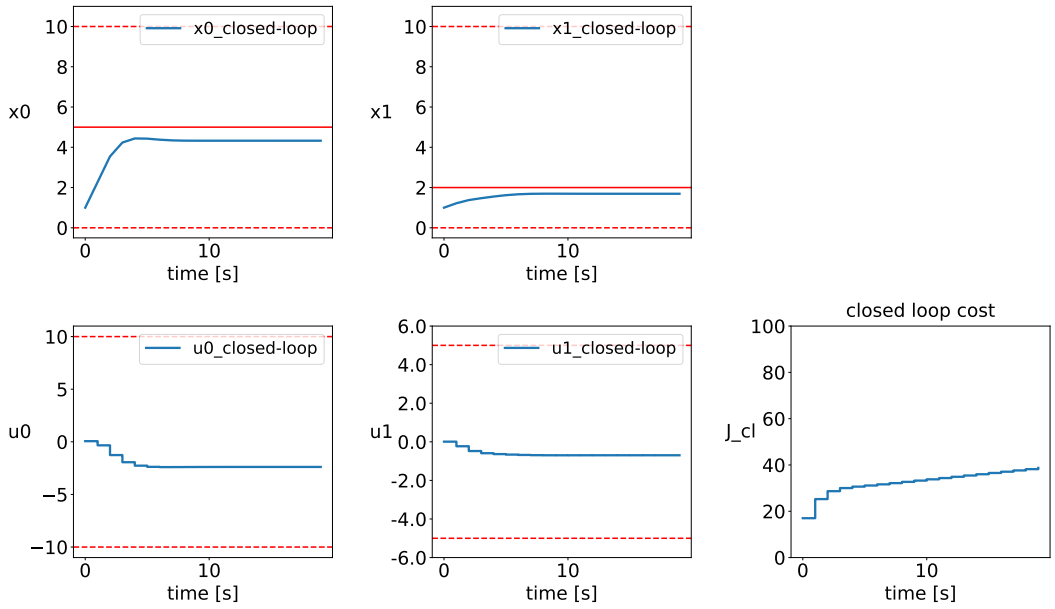

Figure A.47: Full Partitioning MPC for double integrator with no uncertainty for N_s = 9 and ordering [1,0],[0,1]

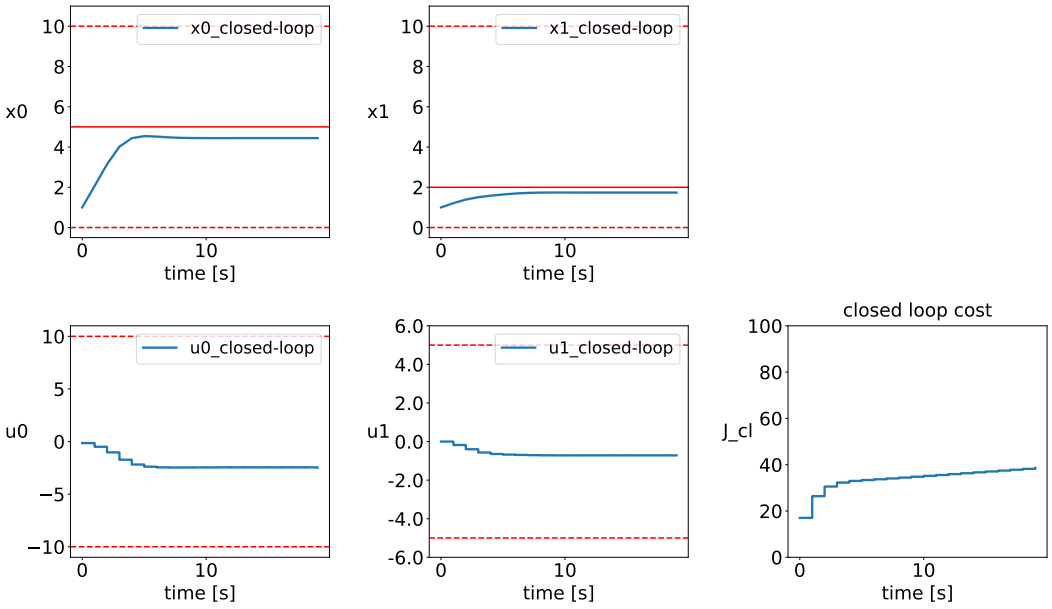

Figure A.48: Full Partitioning MPC for double integrator with no uncertainty for N_s = 16 and ordering [1,0],[0,1]

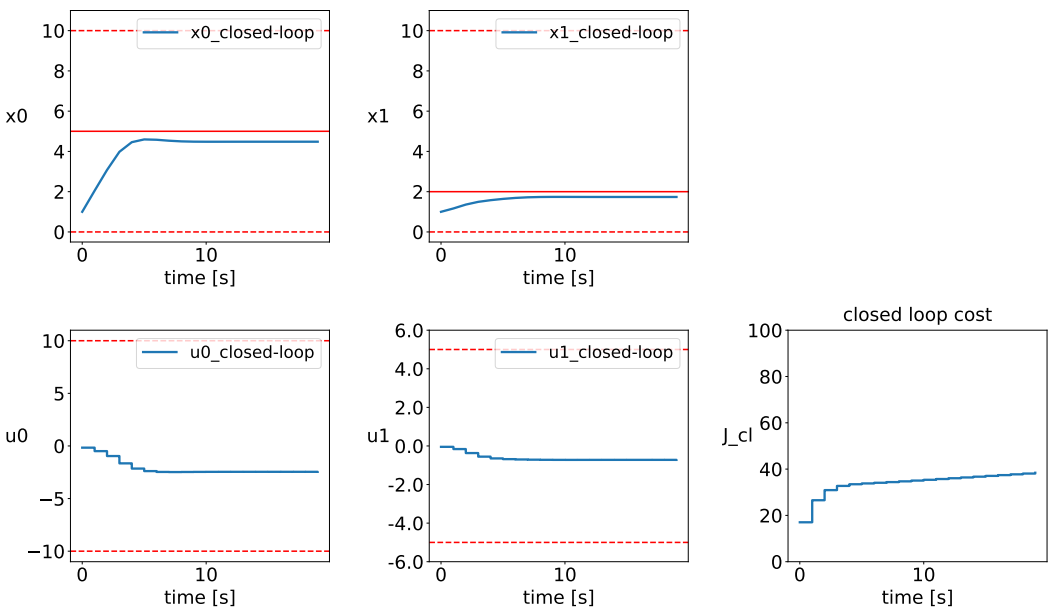

Figure A.49: Full Partitioning MPC for double integrator with no uncertainty for N_s = 24 and ordering [[1,0],[0,1]

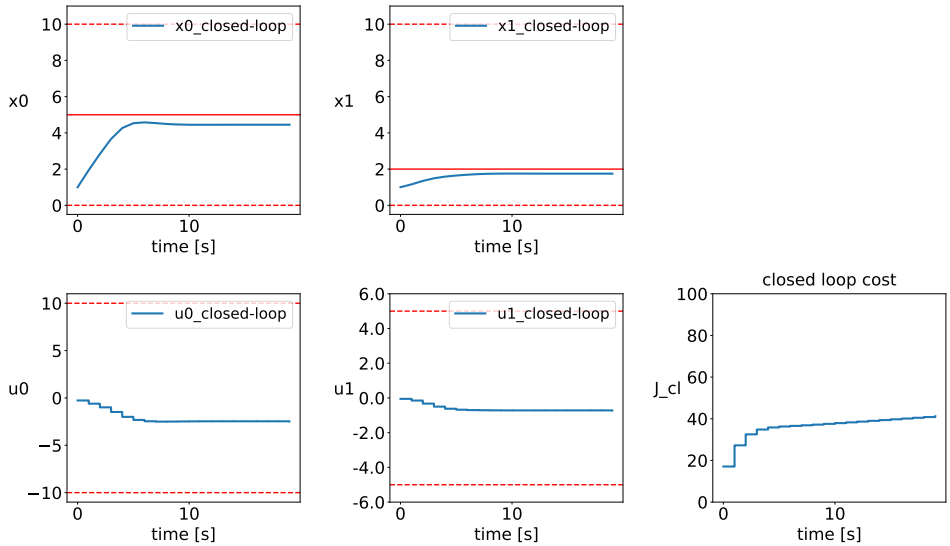

Figure A.50: Full Partitioning MPC for double integrator with no uncertainty for N_s = 30 and ordering [1,0],[0,1]

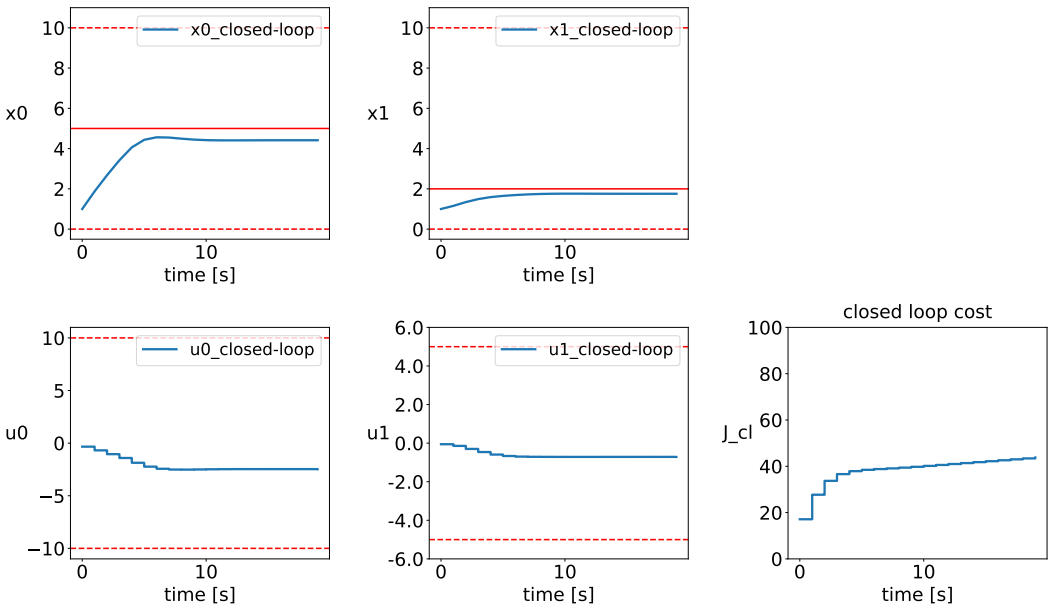

Figure A.51: Full Partitioning MPC for double integrator with no uncertainty for N_s = 36 and ordering [1,0],[0,1]

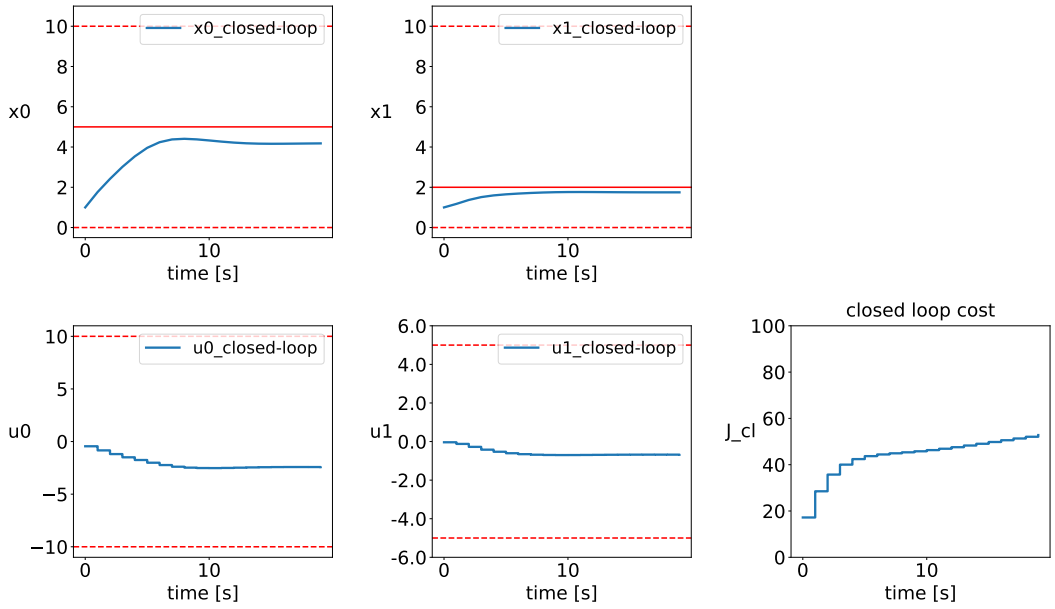

Figure A.52: Full Partitioning MPC for double integrator with no uncertainty for N_s = 45 and ordering [1,0],[0,1]

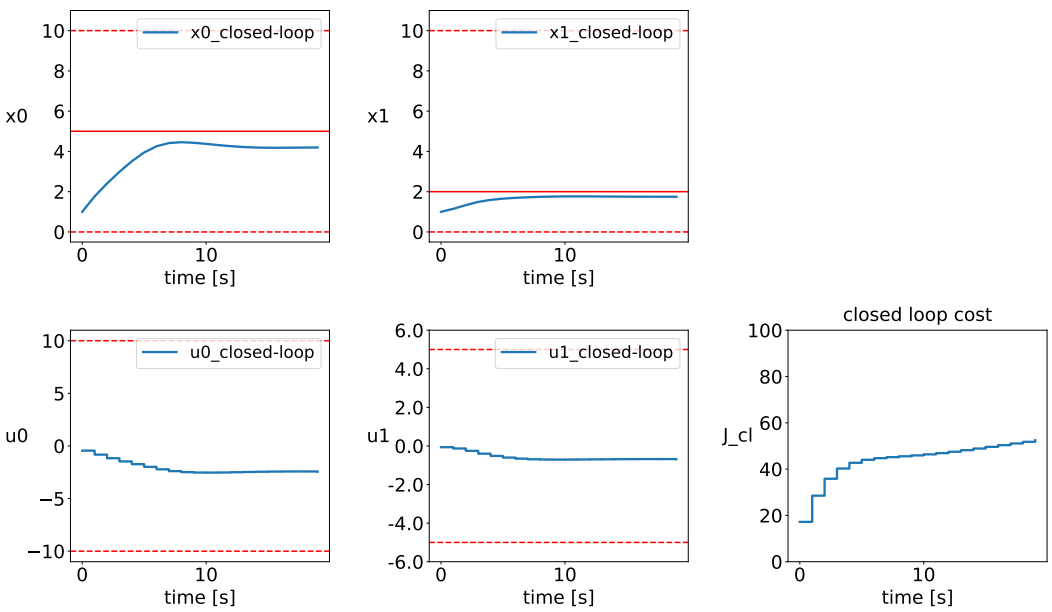

Figure A.53: Full Partitioning MPC for double integrator with no uncertainty for N_s = 54 and ordering [1,0],[0,1]

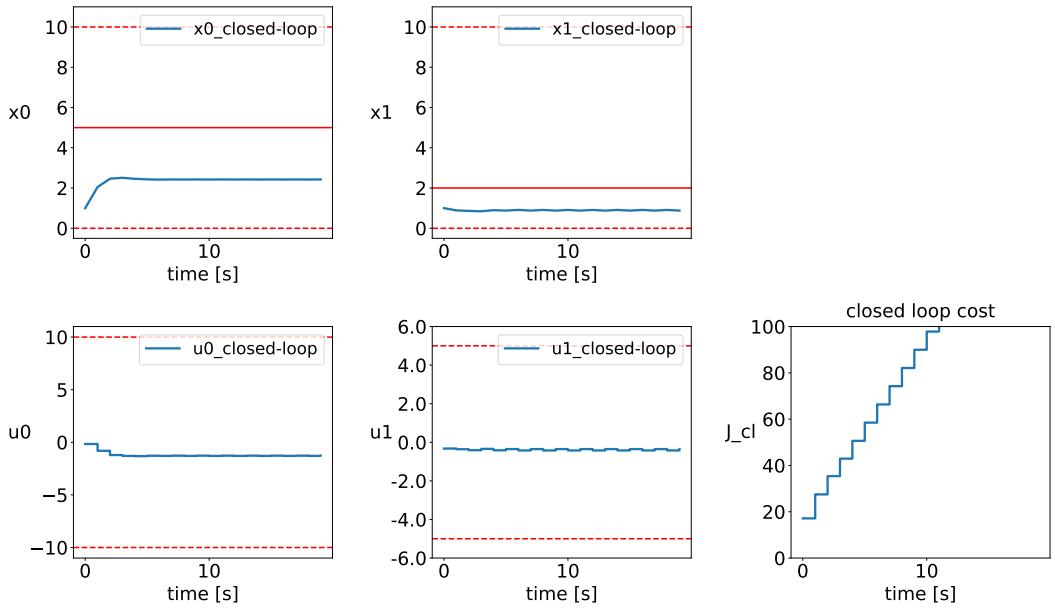

Figure A.54: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 9 and ordering [[1,0],[0,1]

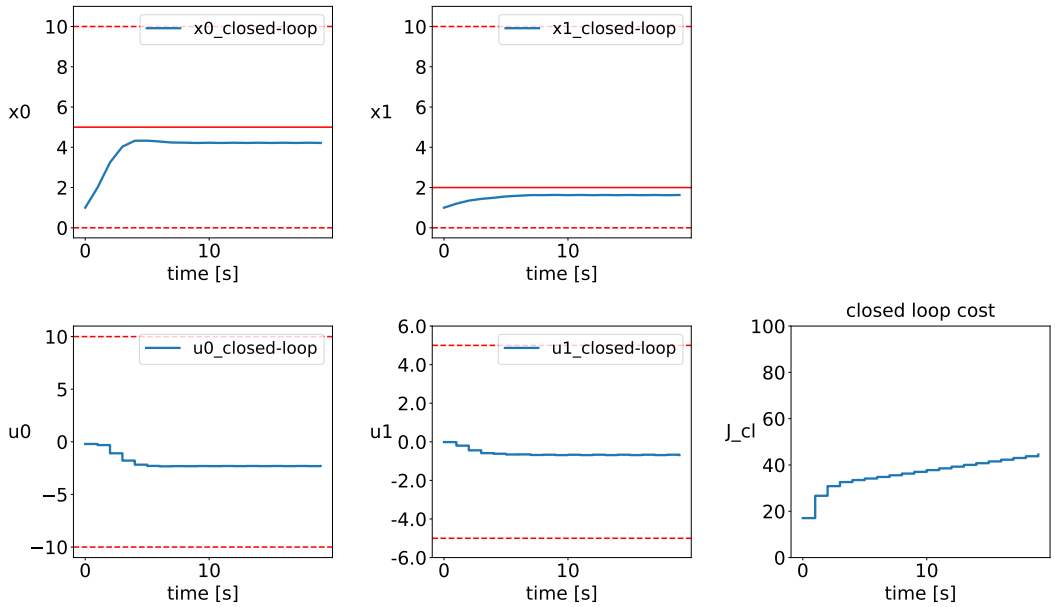

Figure A.55: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 16 and ordering [1,0],[0,1]

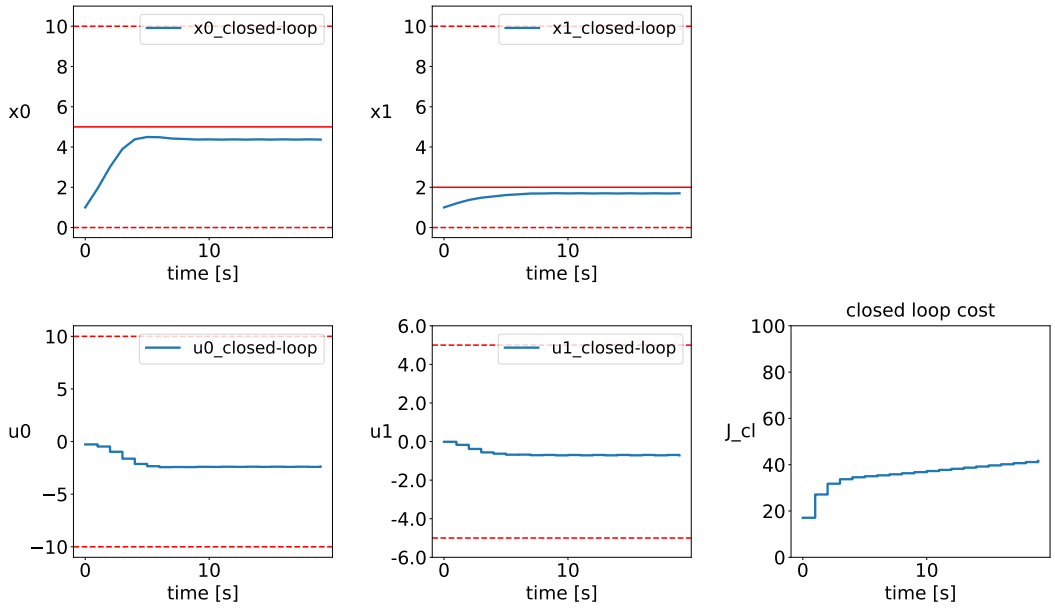

Figure A.56: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 24 and ordering [1,0],[0,1]

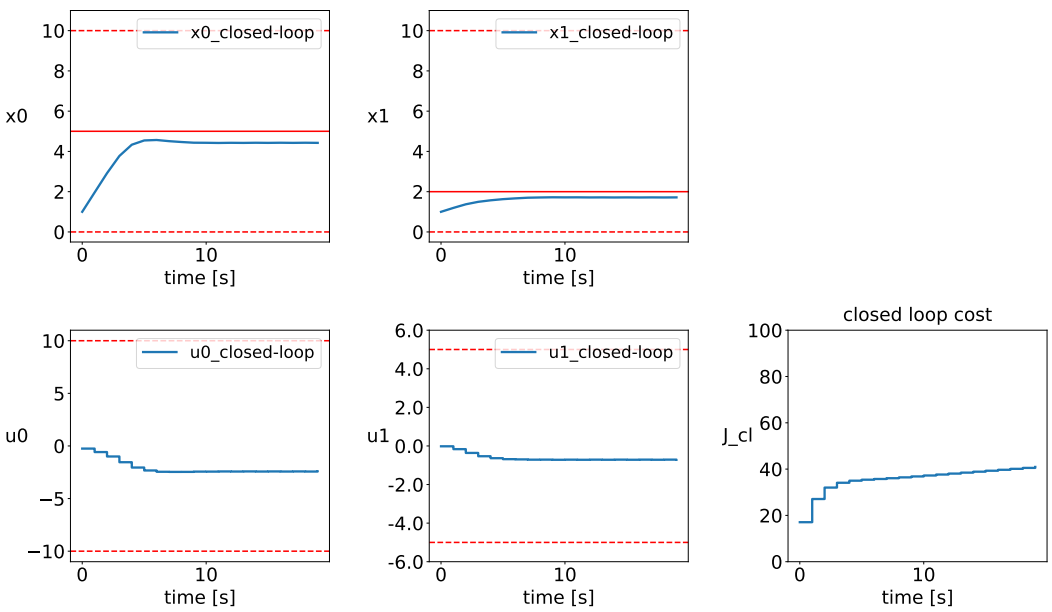

Figure A.57: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 30 and ordering [1,0],[0,1]

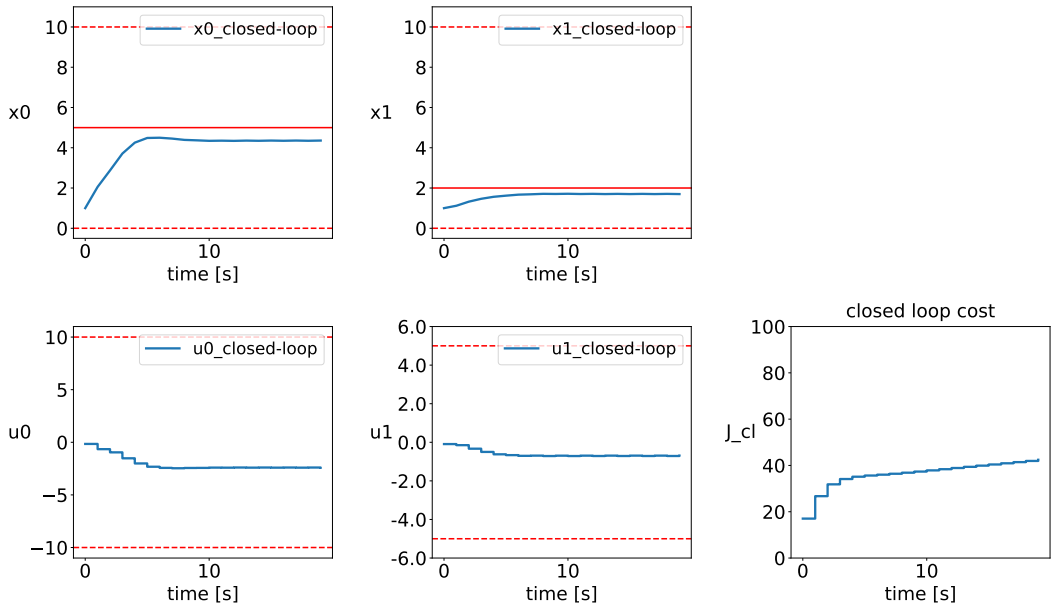

Figure A.58: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 36 and ordering [1,0],[0,1]

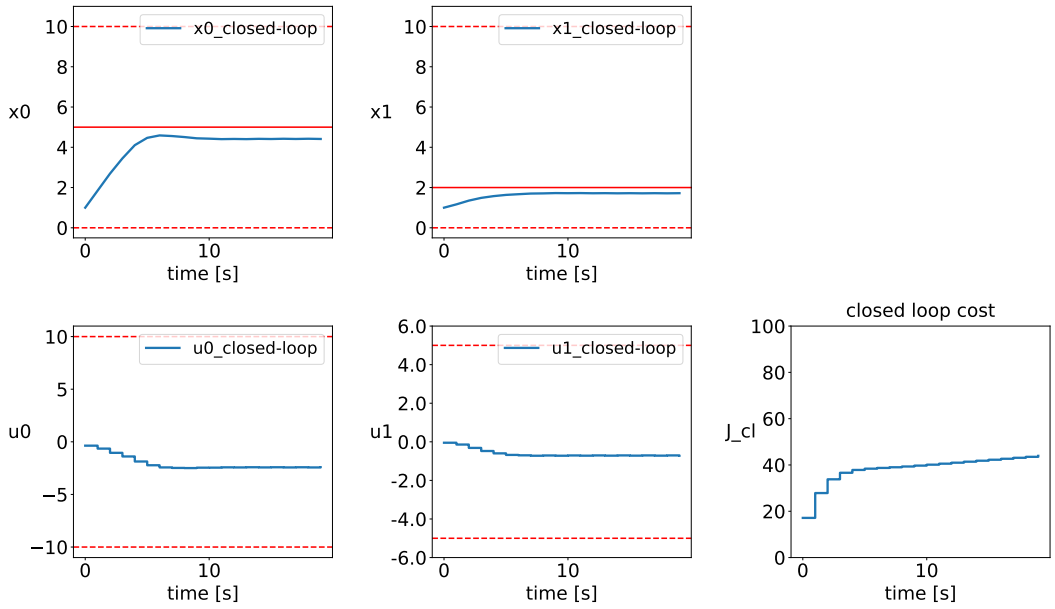

Figure A.59: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 45 and ordering [1,0],[0,1]

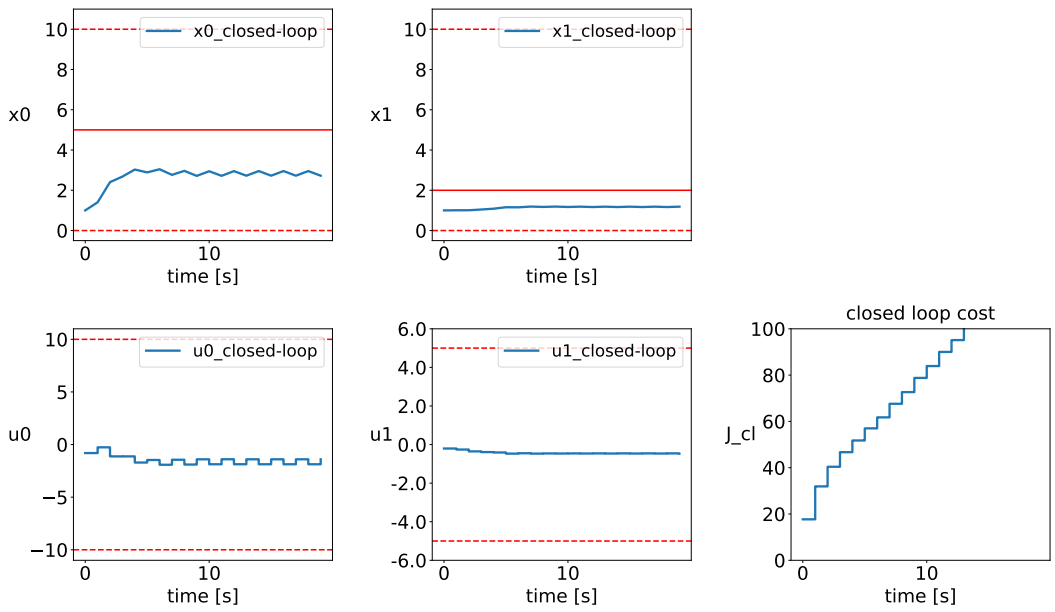

Figure A.60: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 54 and ordering [1,0],[0,1]

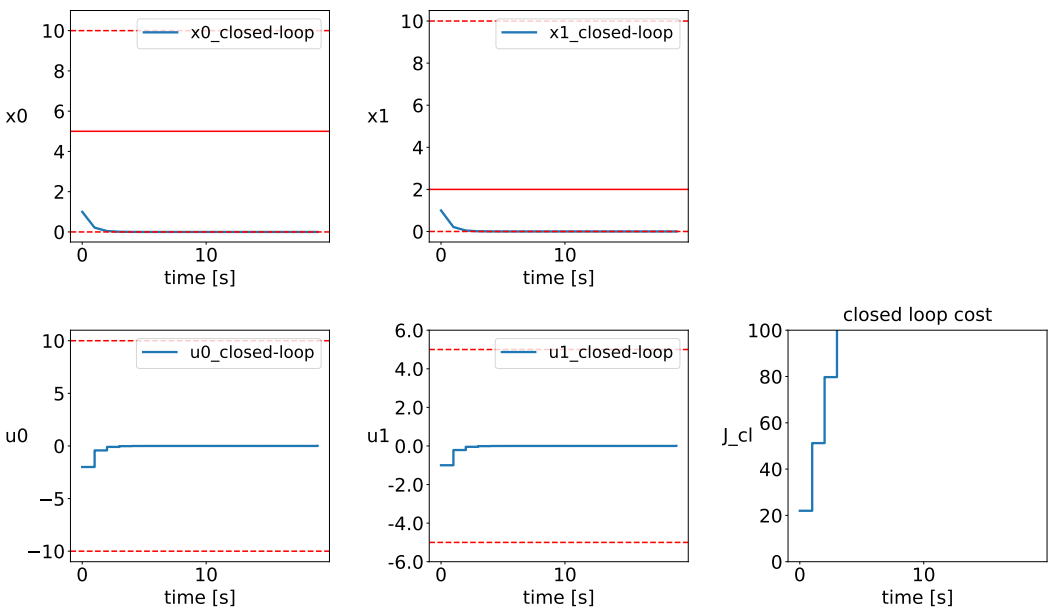

Figure A.61: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 9 and ordering [1,0],[0,1]

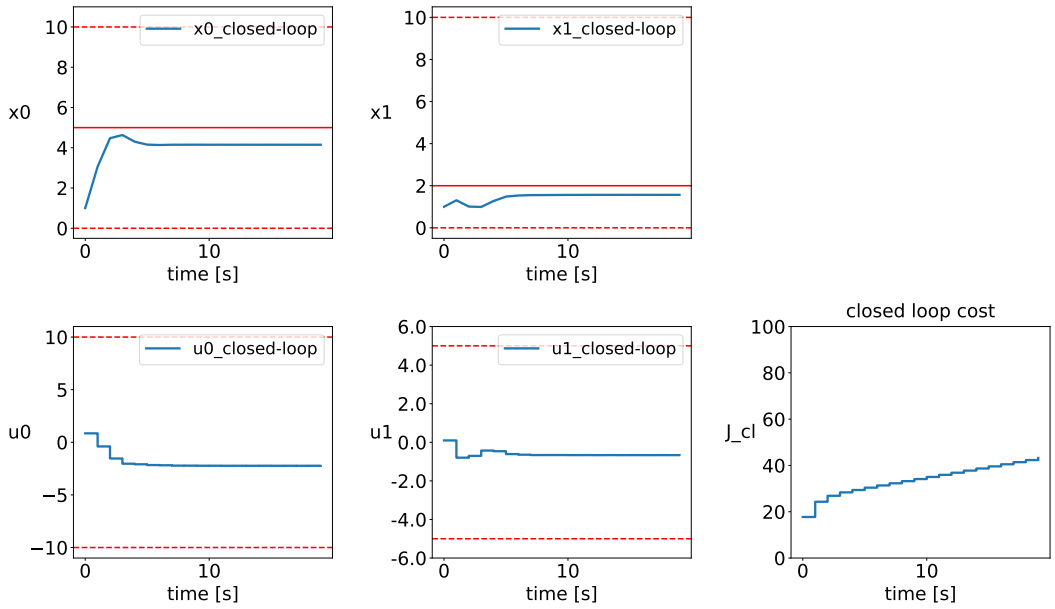

Figure A.62: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 16 and ordering [1,0],[0,1]

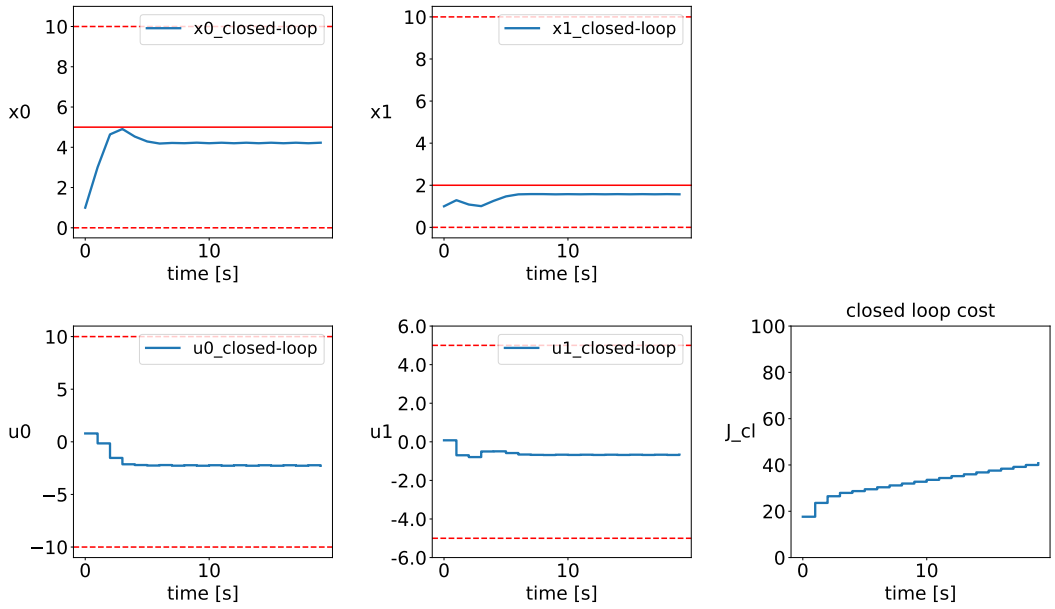

Figure A.63: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 24 and ordering [1,0],[0,1]

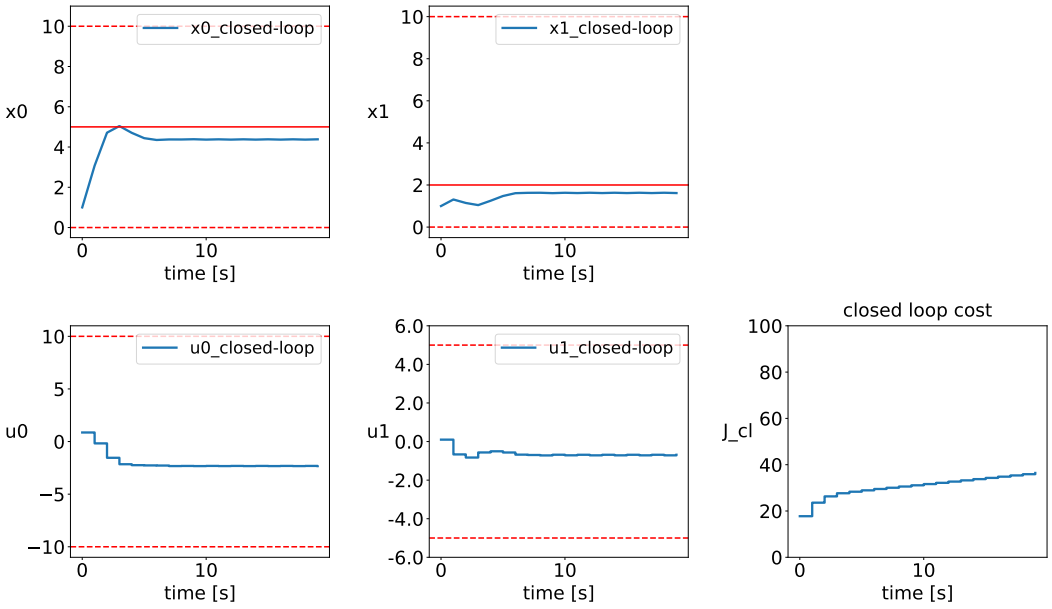

Figure A.64: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 30 and ordering [1,0],[0,1]

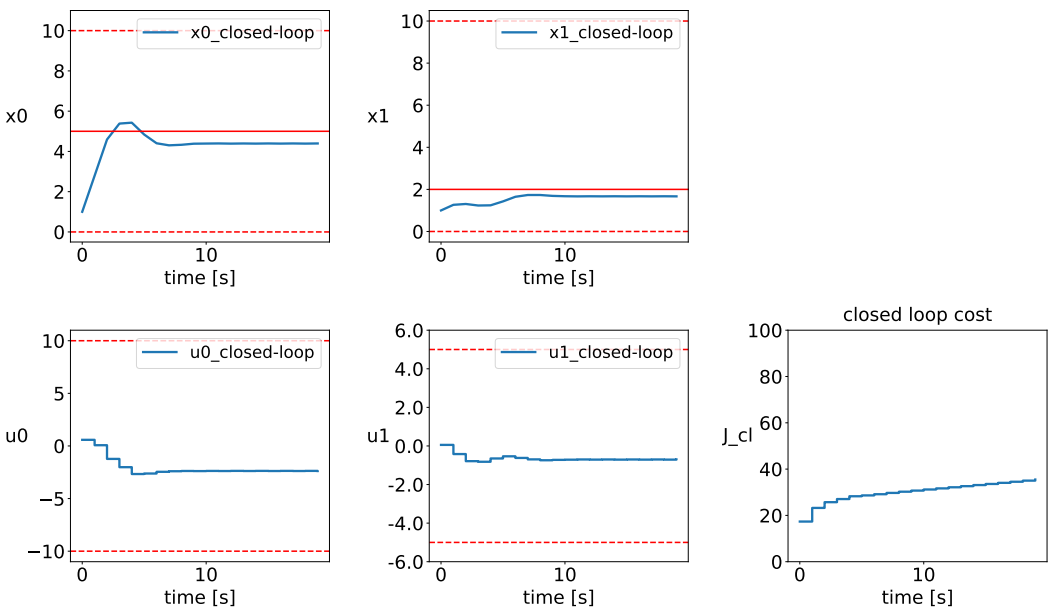

Figure A.65: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 36 and ordering [1,0],[0,1]

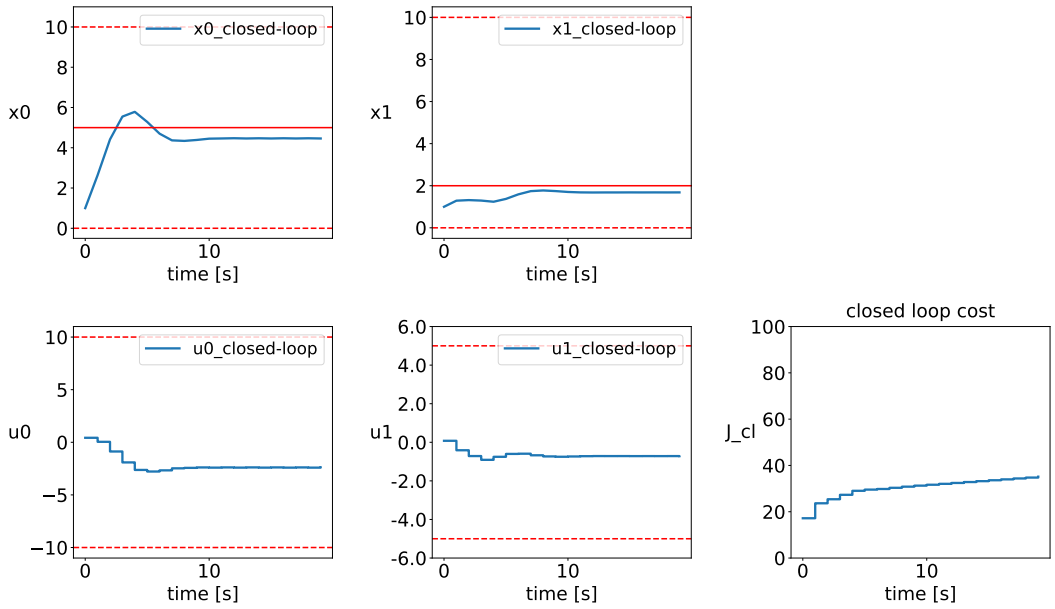

Figure A.66: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 45 and ordering [1,0],[0,1]

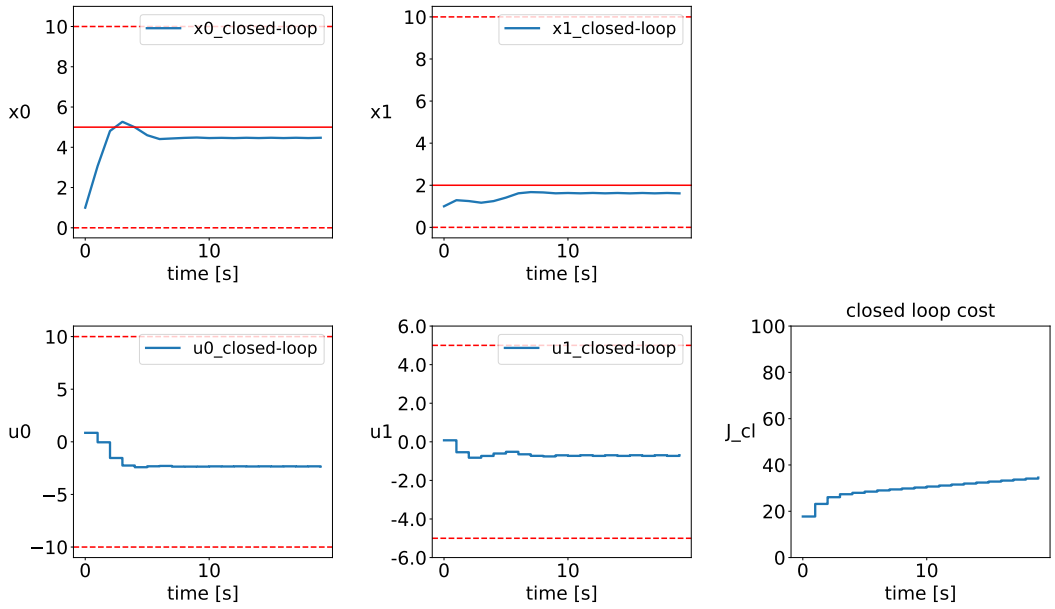

Figure A.67: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 54 and ordering [1,0],[0,1]

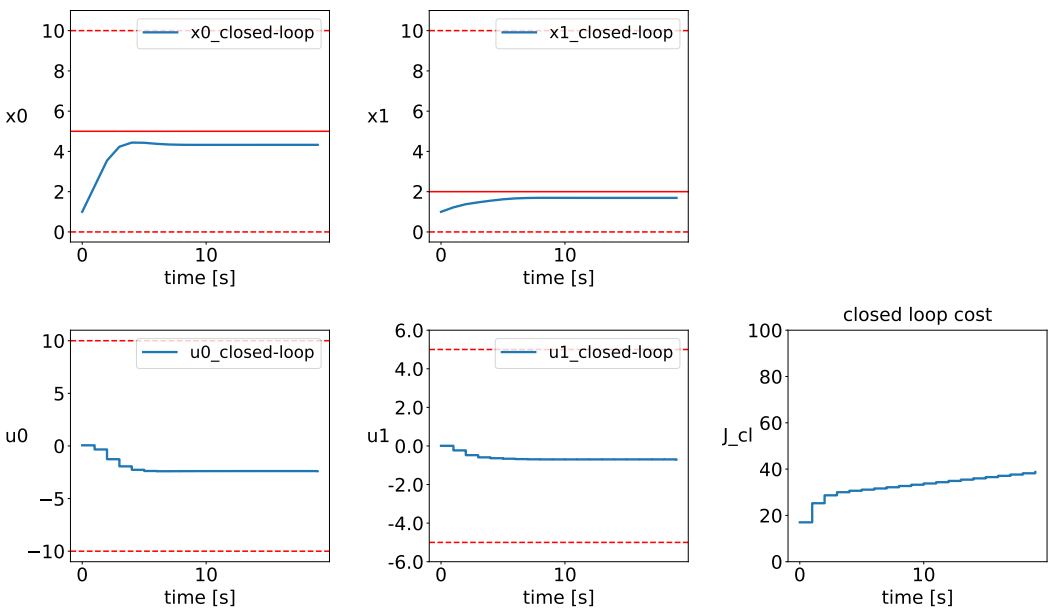

Figure A.68: Full Partitioning MPC for double integrator with no uncertainty for N_s = 9 and ordering [1,0],[1,0]

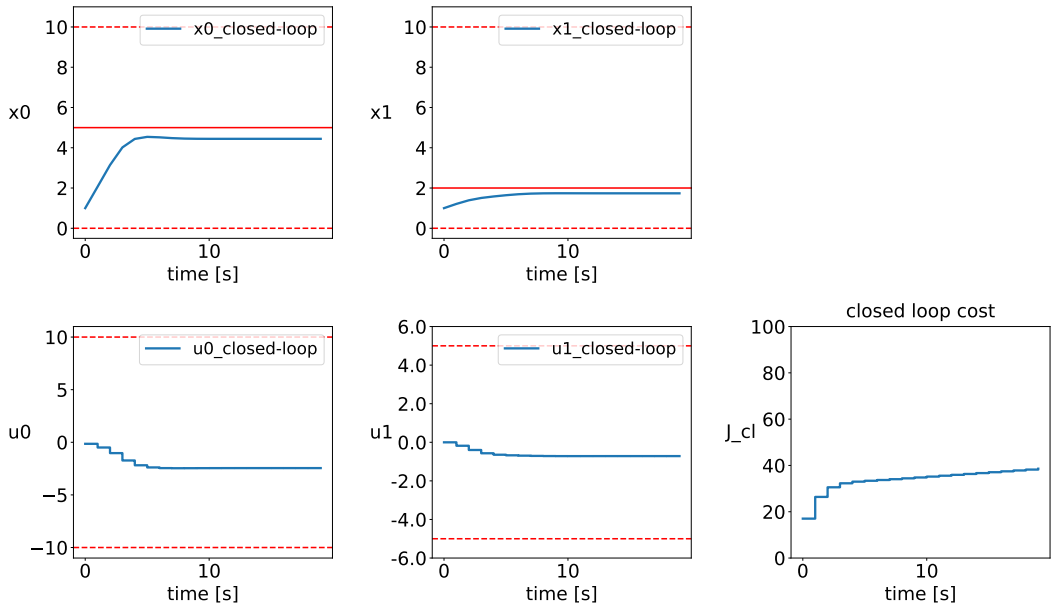

Figure A.69: Full Partitioning MPC for double integrator with no uncertainty for N_s = 16 and ordering [1,0],[1,0]

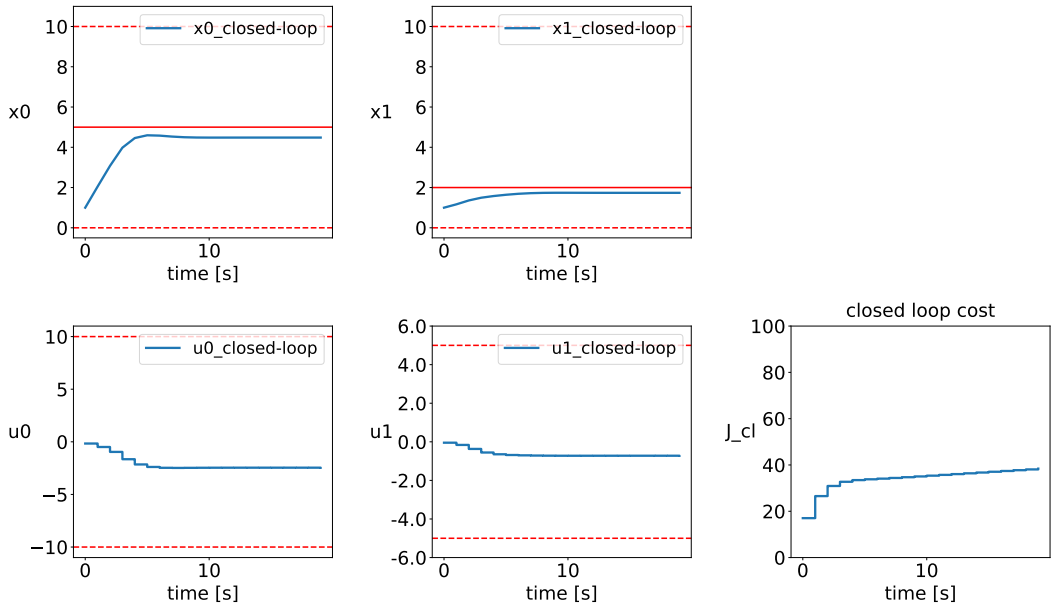

Figure A.70: Full Partitioning MPC for double integrator with no uncertainty for N_s = 24 and ordering [[1,0],[1,0]

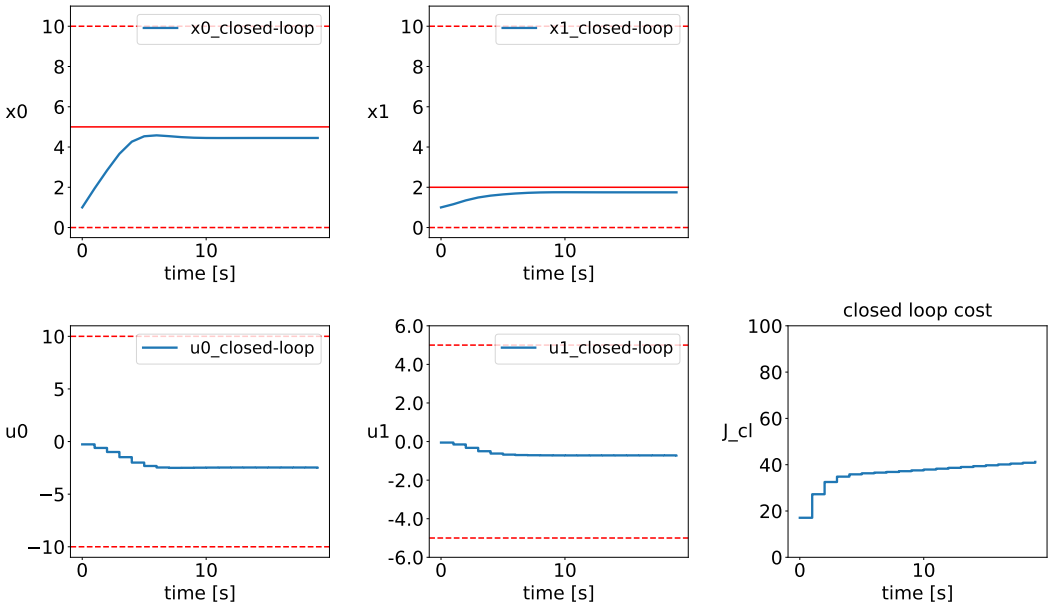

Figure A.71: Full Partitioning MPC for double integrator with no uncertainty for N_s = 30 and ordering [1,0],[1,0]

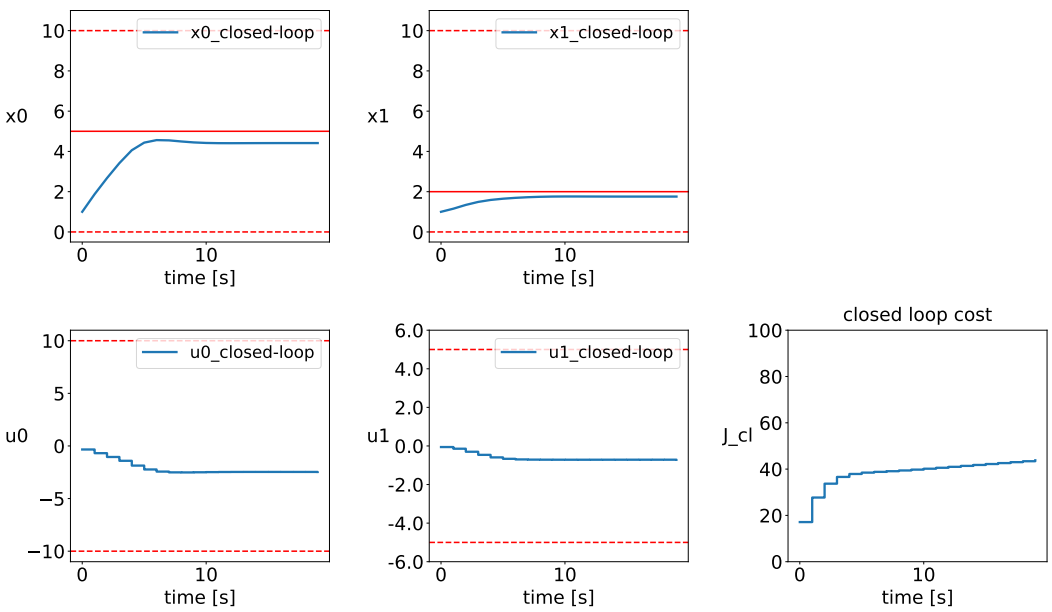

Figure A.72: Full Partitioning MPC for double integrator with no uncertainty for N_s = 36 and ordering [1,0],[1,0]

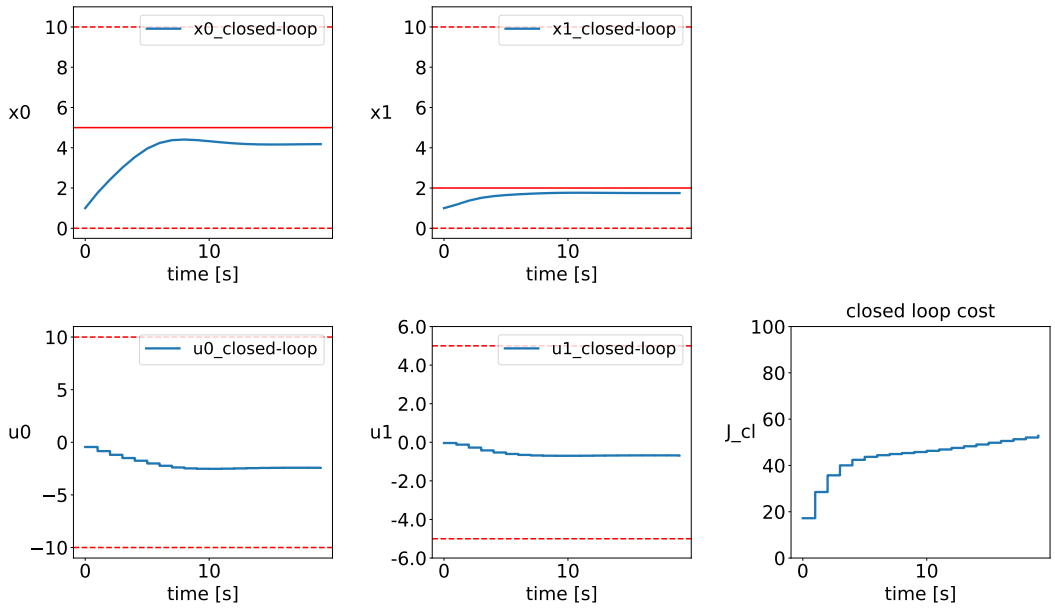

Figure A.73: Full Partitioning MPC for double integrator with no uncertainty for N_s = 45 and ordering [1,0],[1,0]

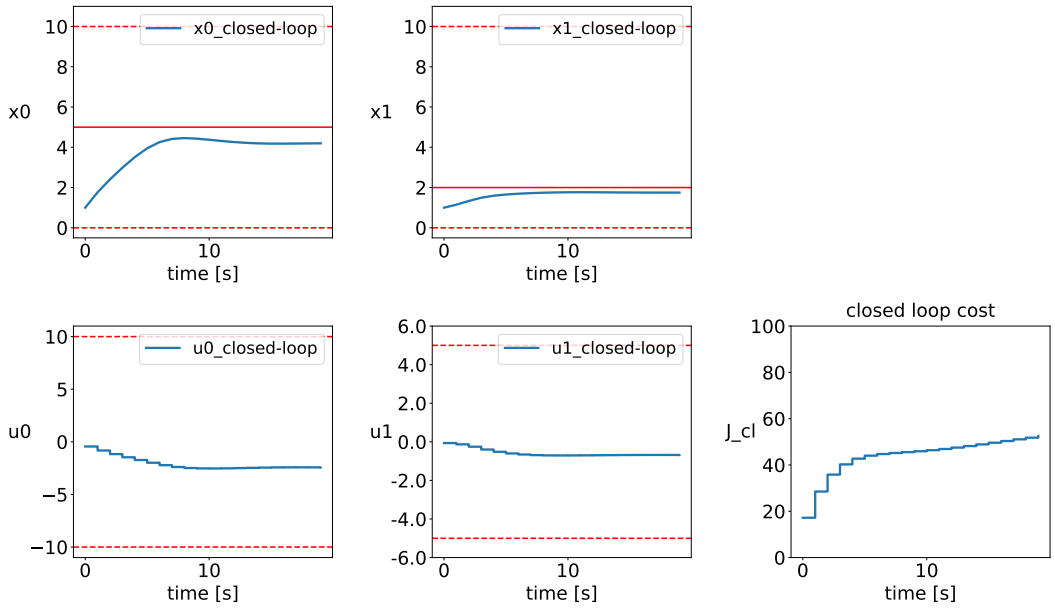

Figure A.74: Full Partitioning MPC for double integrator with no uncertainty for N_s = 54 and ordering [1,0],[1,0]

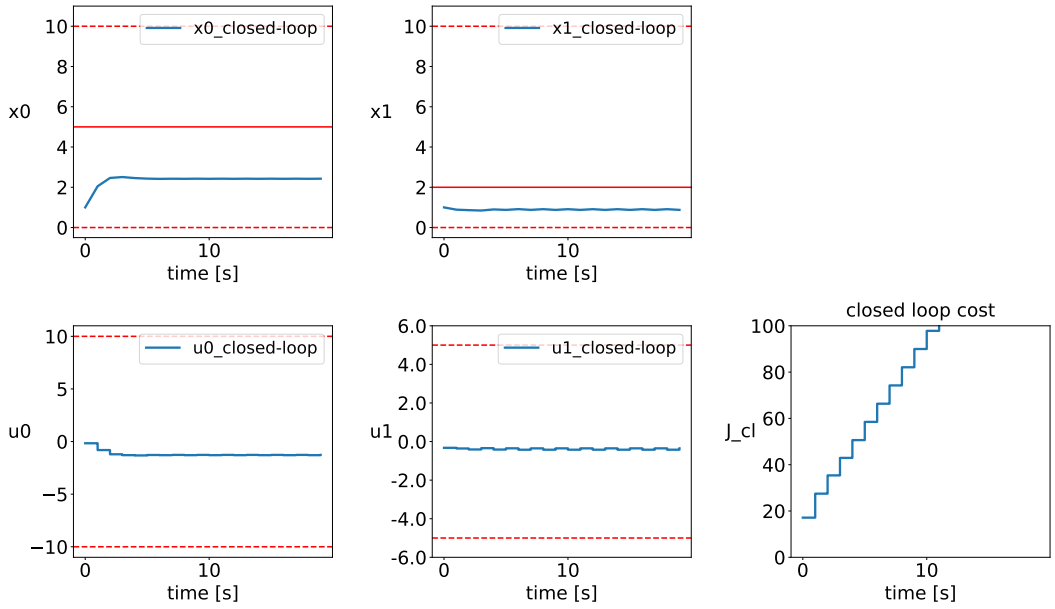

Figure A.75: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 9 and ordering [[1,0],[1,0]

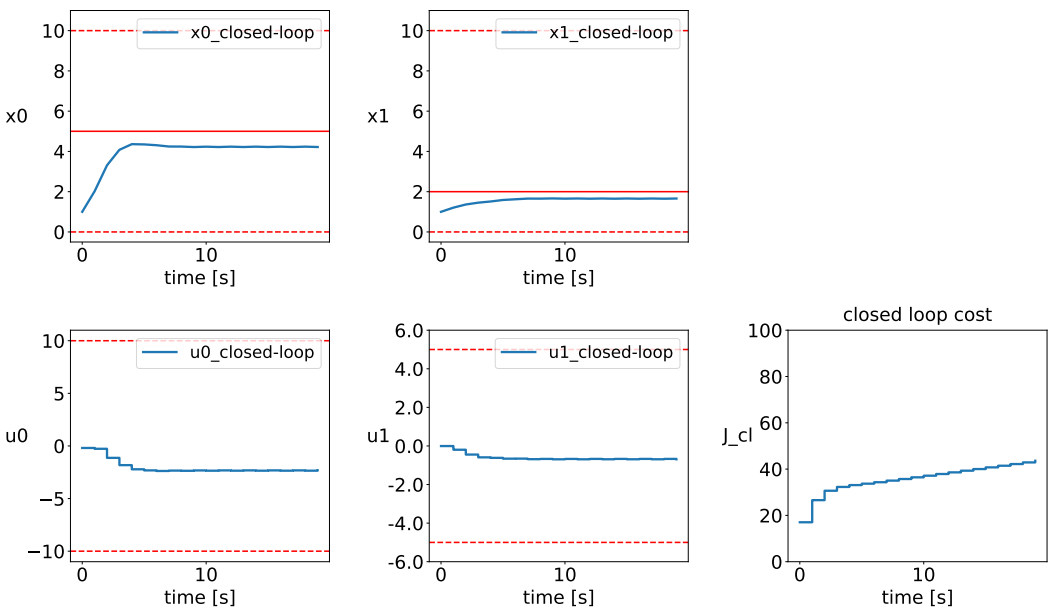

Figure A.76: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 16 and ordering [1,0],[1,0]

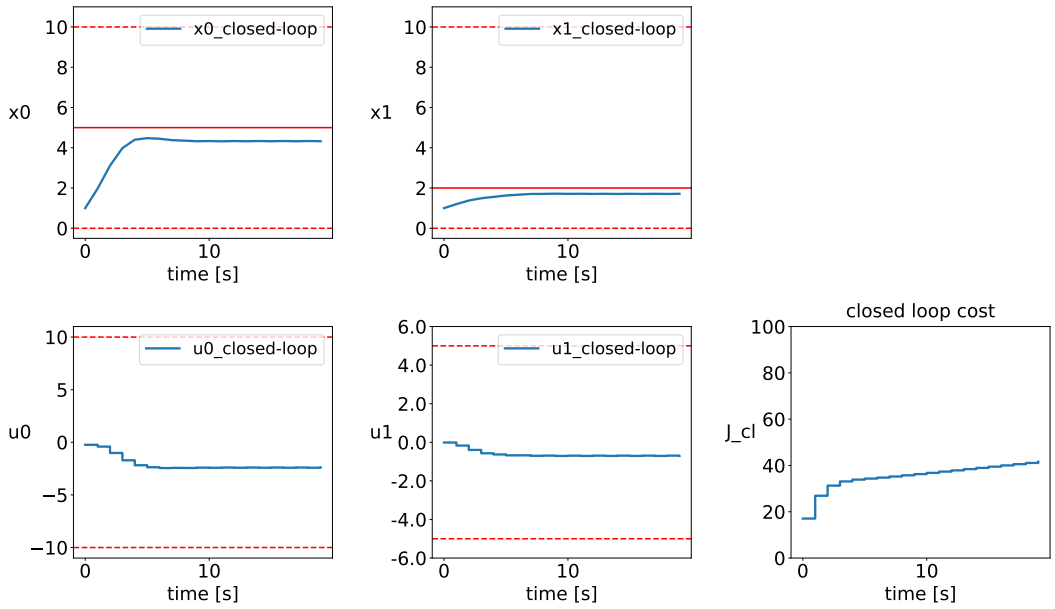

Figure A.77: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 24 and ordering [1,0],[1,0]

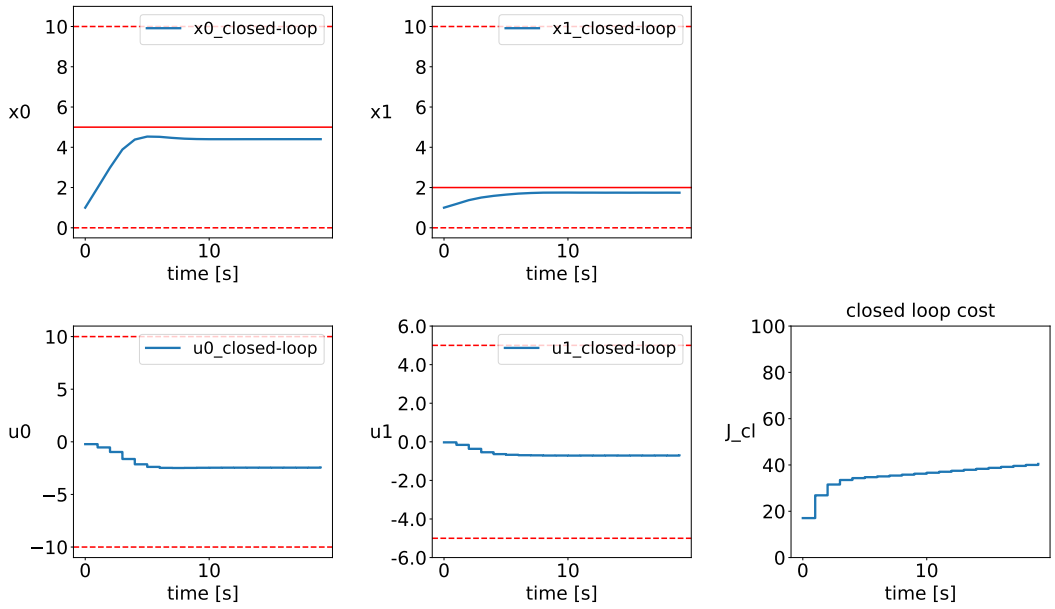

Figure A.78: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 30 and ordering [1,0],[1,0]

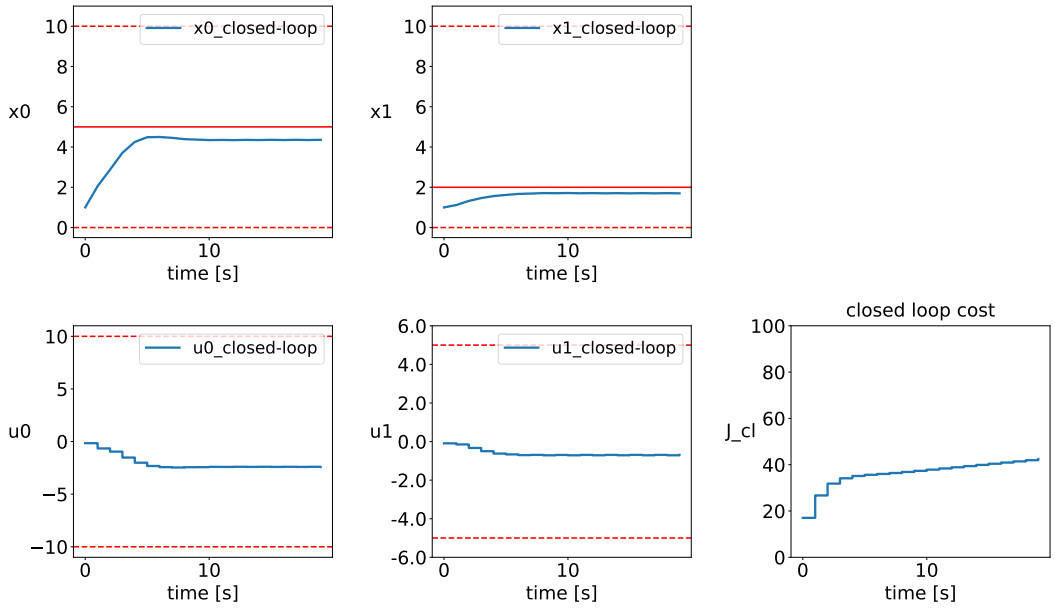

Figure A.79: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 36 and ordering [1,0],[1,0]

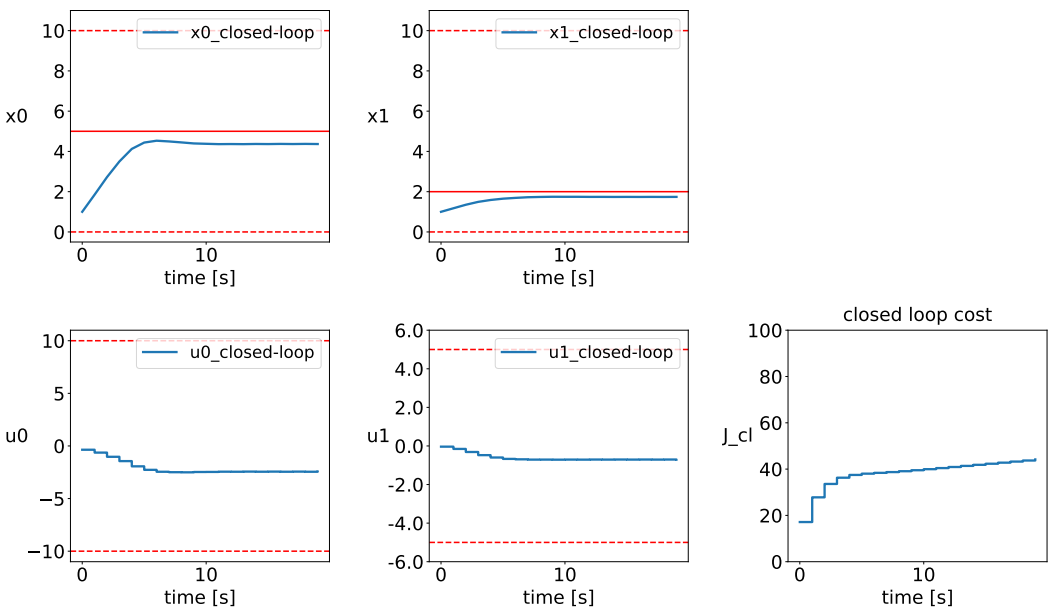

Figure A.80: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 45 and ordering [1,0],[1,0]

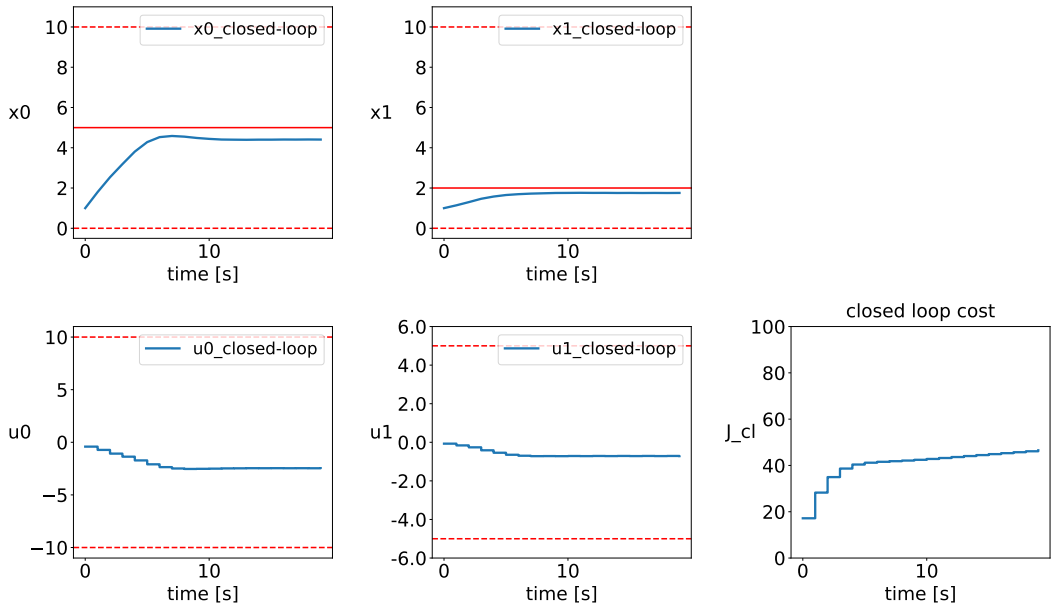

Figure A.81: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 54 and ordering [1,0],[1,0]

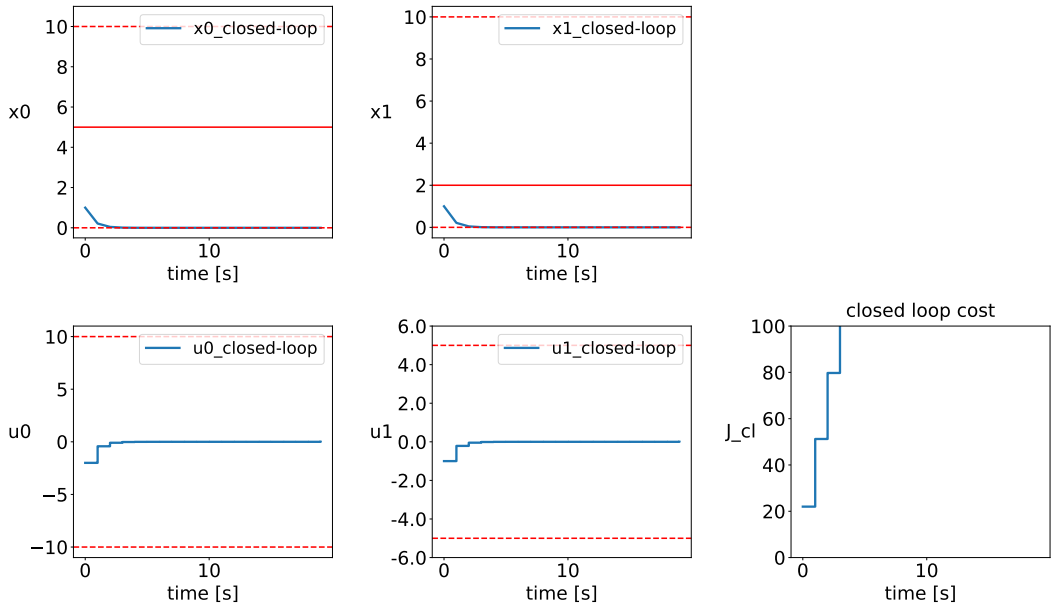

Figure A.82: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 9 and ordering [1,0],[1,0]

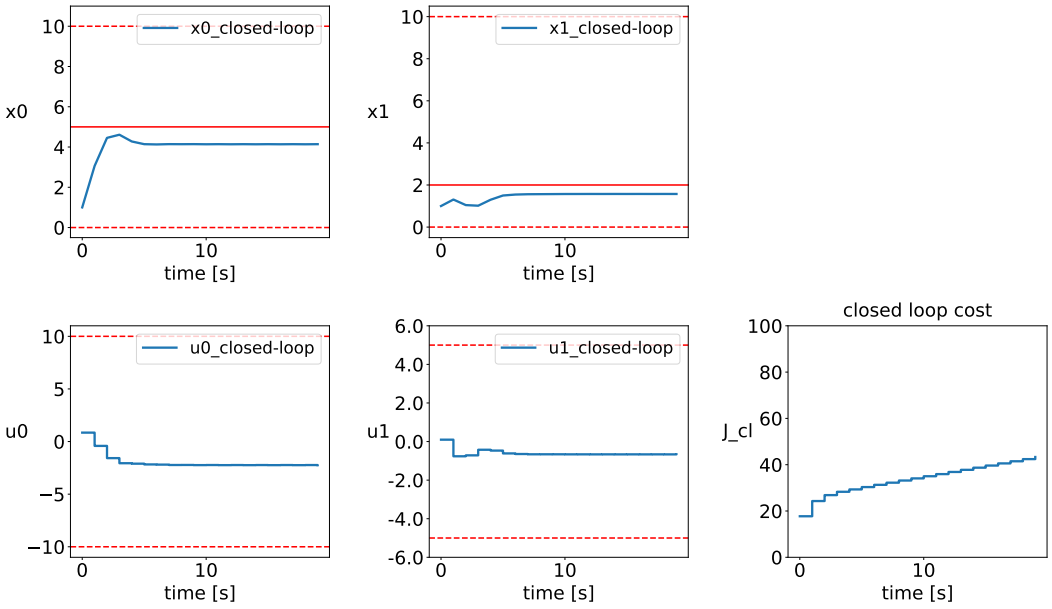

Figure A.83: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 16 and ordering [1,0],[1,0]

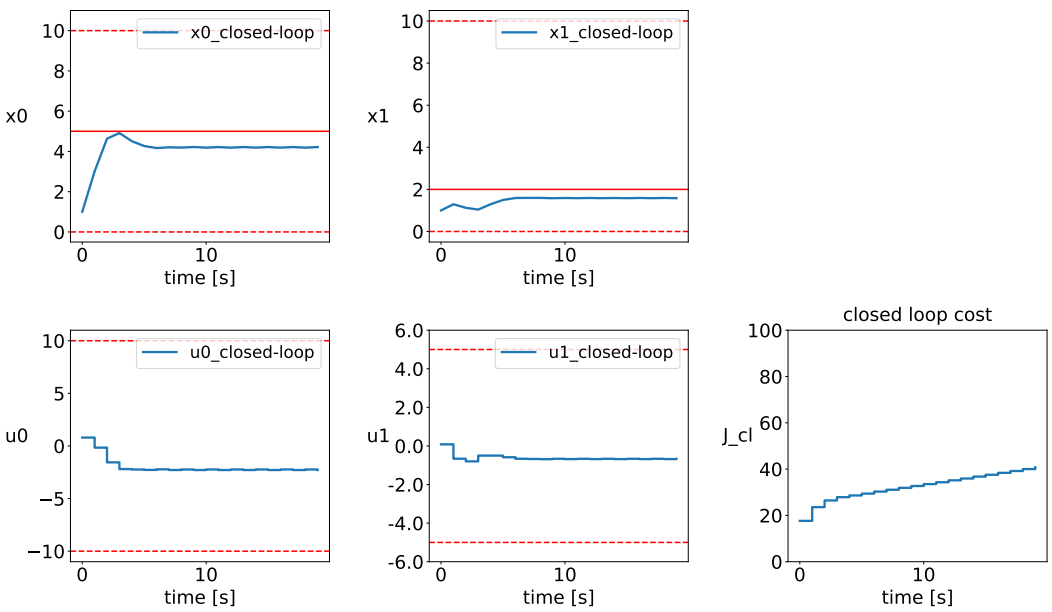

Figure A.84: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 24 and ordering [1,0],[1,0]

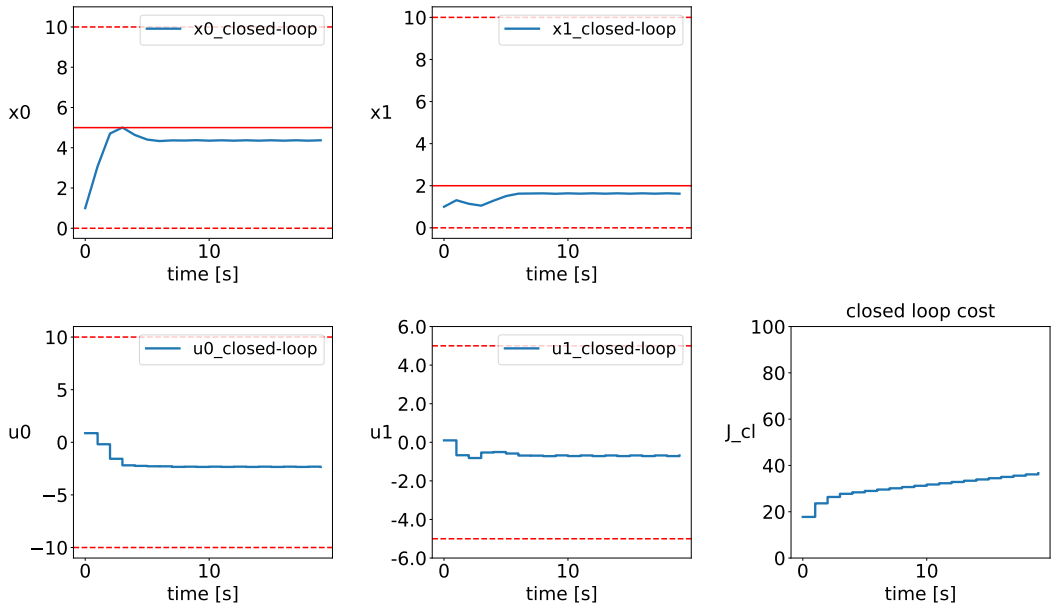

Figure A.85: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 30 and ordering [1,0],[1,0]

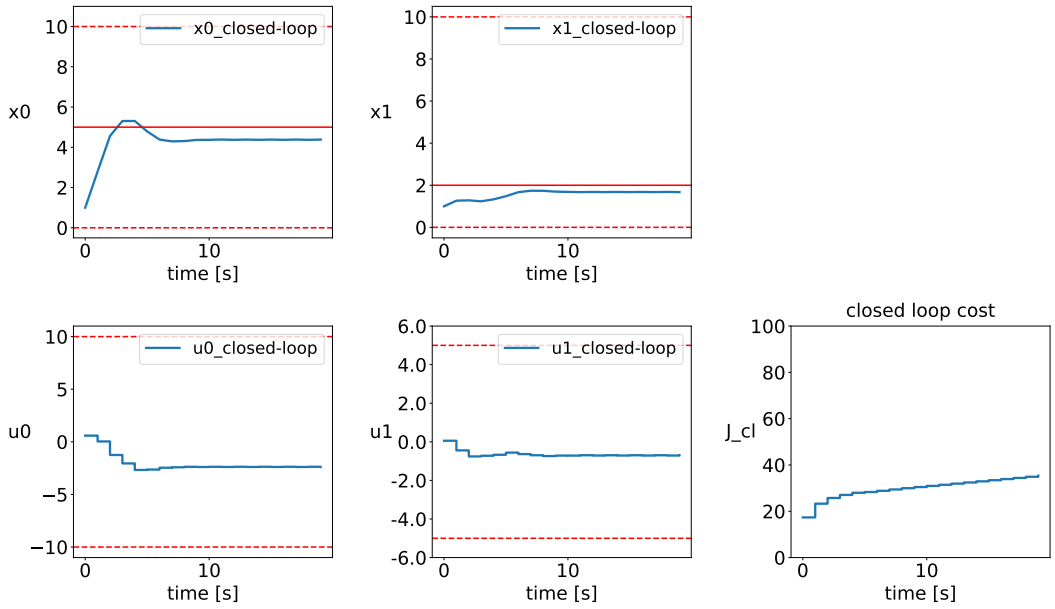

Figure A.86: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 36 and ordering [1,0],[1,0]

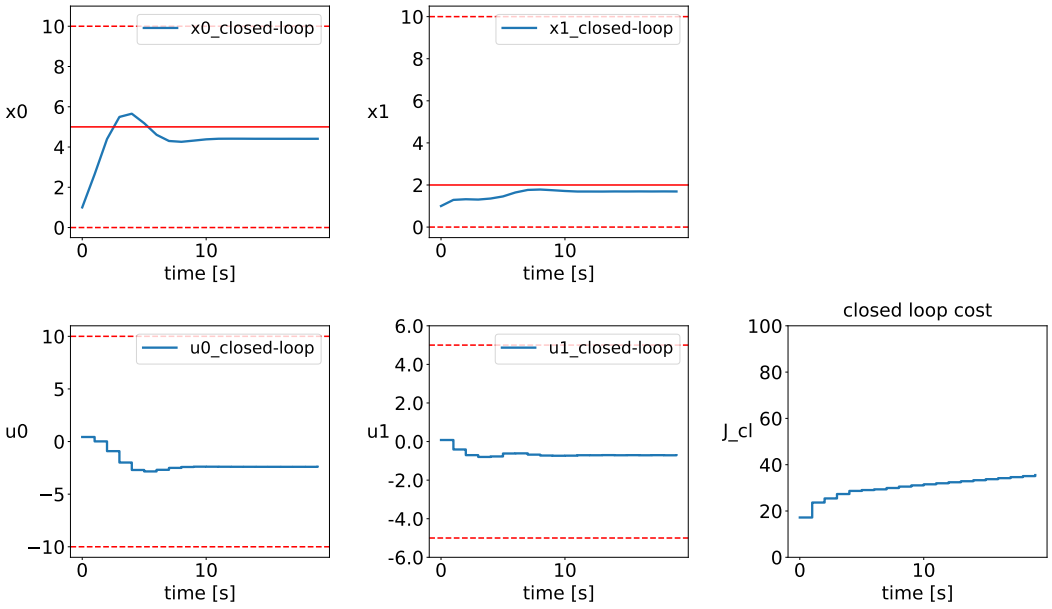

Figure A.87: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 45 and ordering [1,0],[1,0]

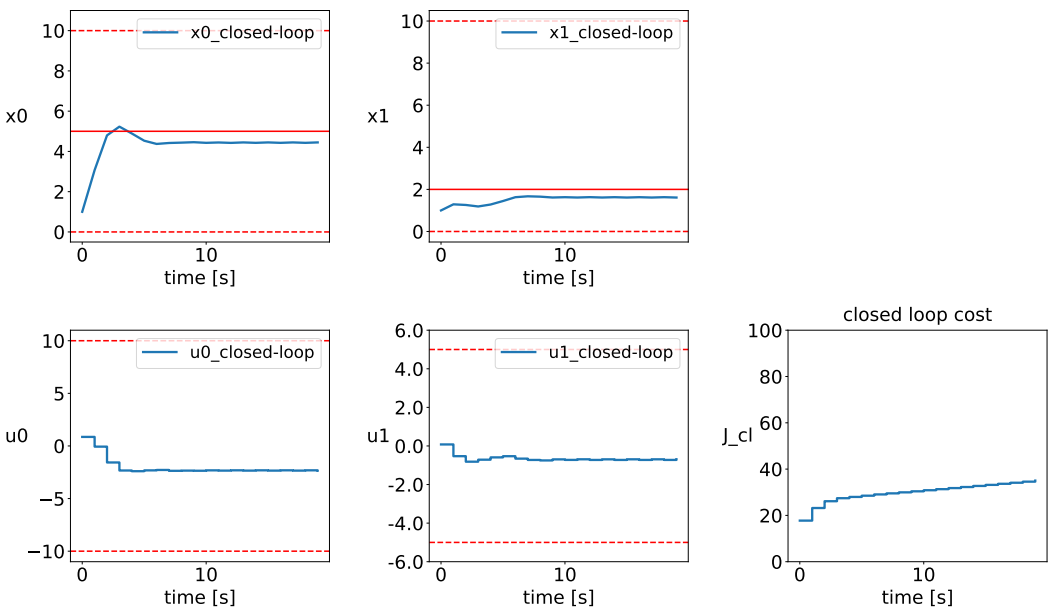

Figure A.88: Alternating-Variable MPC for double integrator with no uncertainty for N_s = 54 and ordering [1,0],[1,0]

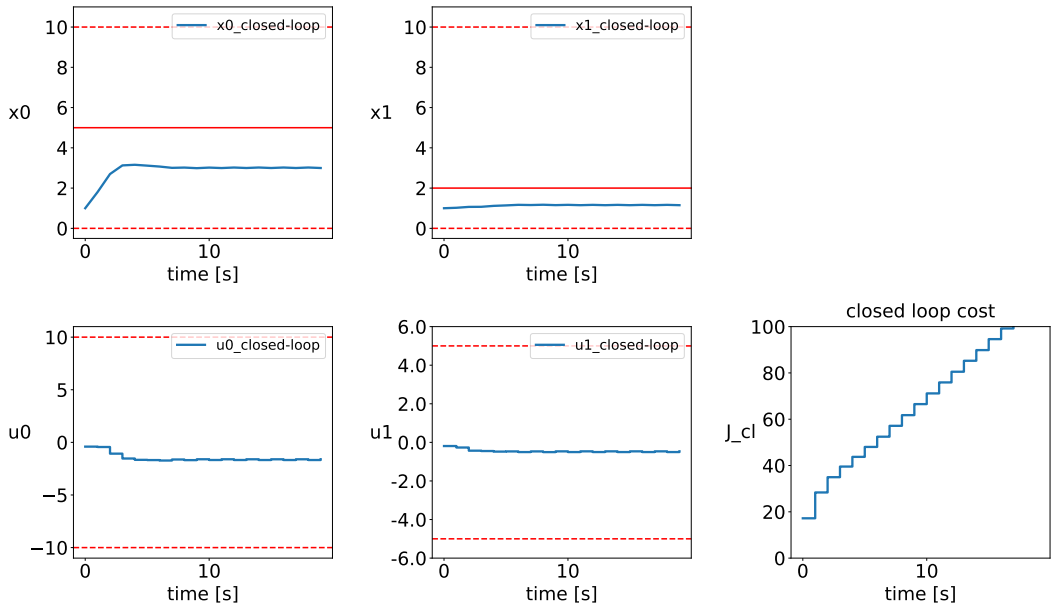

Figure A.89: Alternating-Constant MPC for double integrator with no uncertainty for N_s = 16 and ordering [1,0],[1,0] and only partitioning of one dimension in subsequent steps

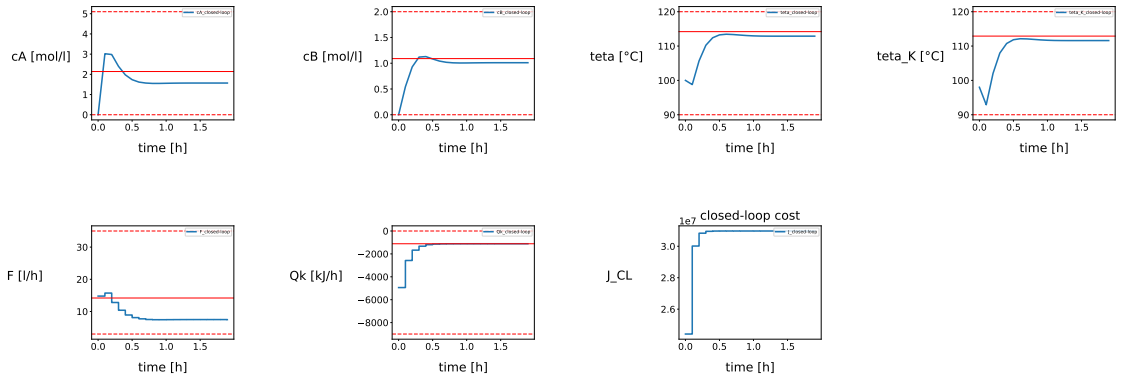

Figure A.90: Full Partitioning MPC with no uncertainty for CSTR with N_s = 4 and ordering [1,2,3,4]

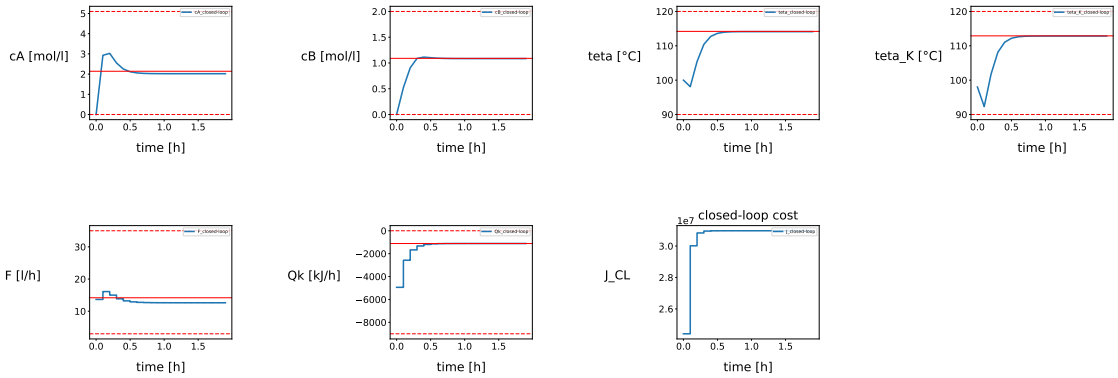

Figure A.91: Full Partitioning MPC with no uncertainty for N_s = 8 and ordering [1,2,3,4]

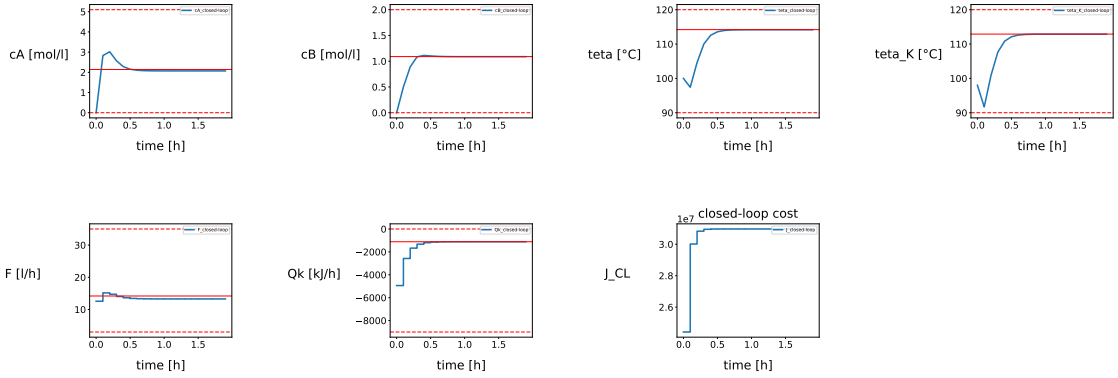

Figure A.92: Full Partitioning MPC with no uncertainty for CSTR with N_s = 16 and ordering [1,2,3,4]

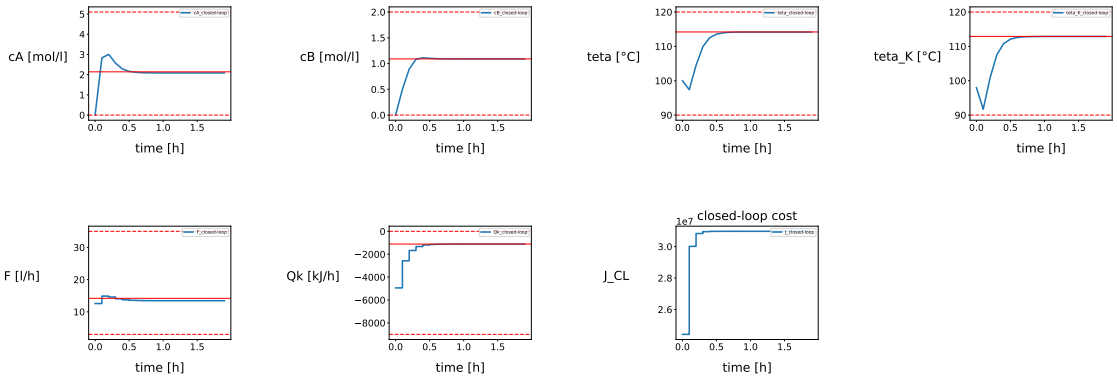

Figure A.93: Full Partitioning MPC with no uncertainty for CSTR with N_s = 24 and ordering [1,2,3,4]

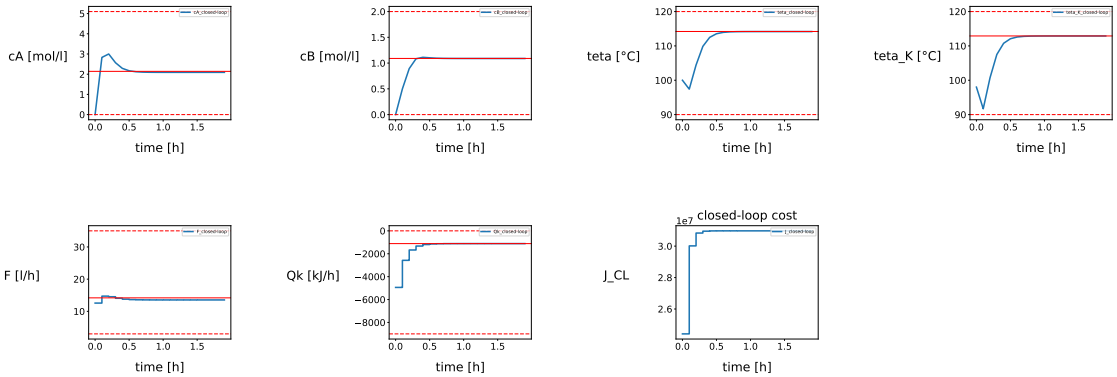

Figure A.94: Full Partitioning MPC with no uncertainty for CSTR with N_s = 32 and ordering [1,2,3,4]

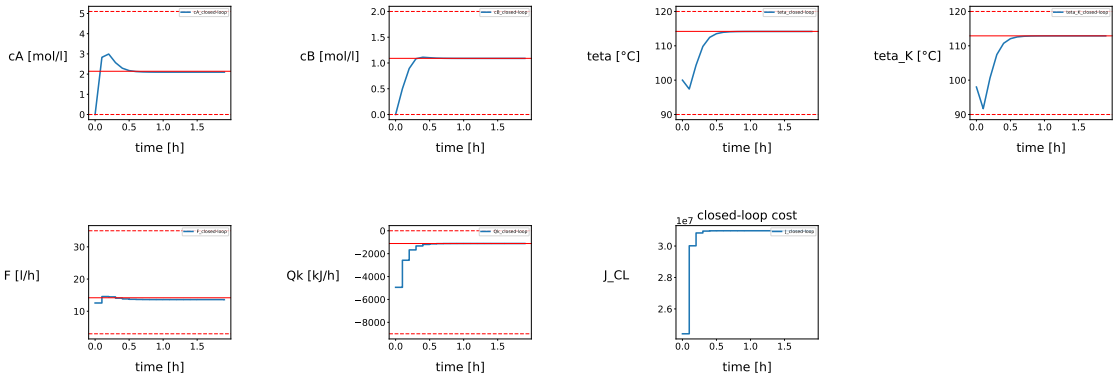

Figure A.95: Full Partitioning MPC with no uncertainty for CSTR with N_s = 40 and ordering [1,2,3,4]

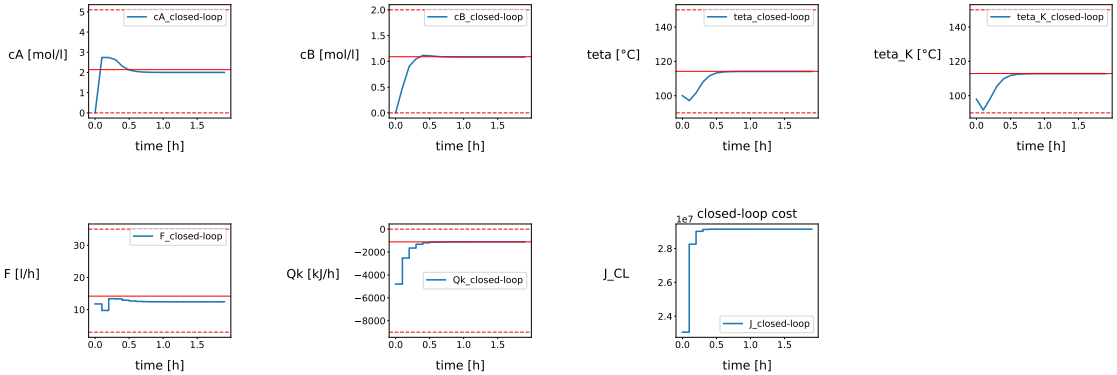

Figure A.96: Alternating Constant Partitioning MPC with no uncertainty for CSTR with N_s = 4 and ordering [1,2,3,4],[1,2,3,4]

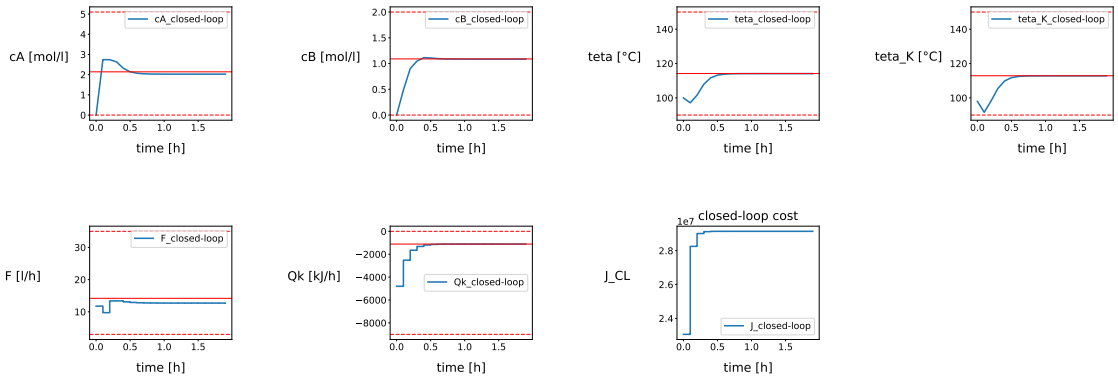

Figure A.97: Alternating Constant Partitioning MPC with no uncertainty for CSTR with N_s = 8 and ordering [1,2,3,4],[1,2,3,4]

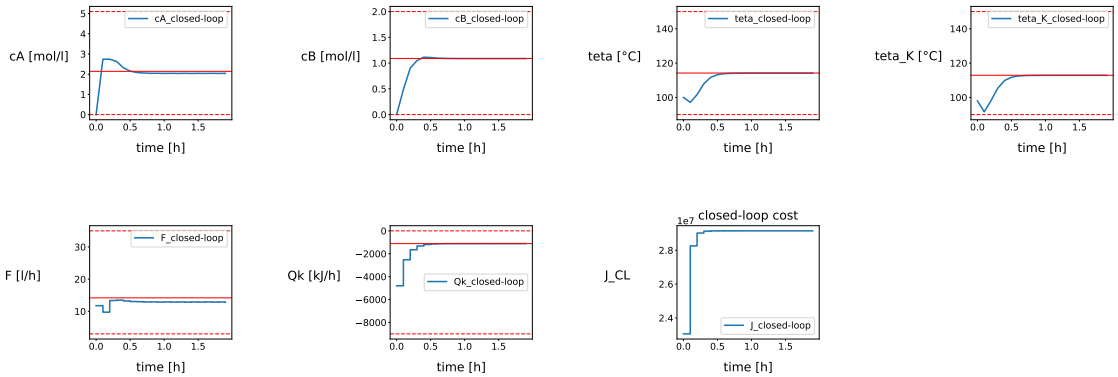

Figure A.98: Alternating Constant Partitioning MPC with no uncertainty for CSTR with N_s = 16 and ordering [1,2,3,4],[1,2,3,4]

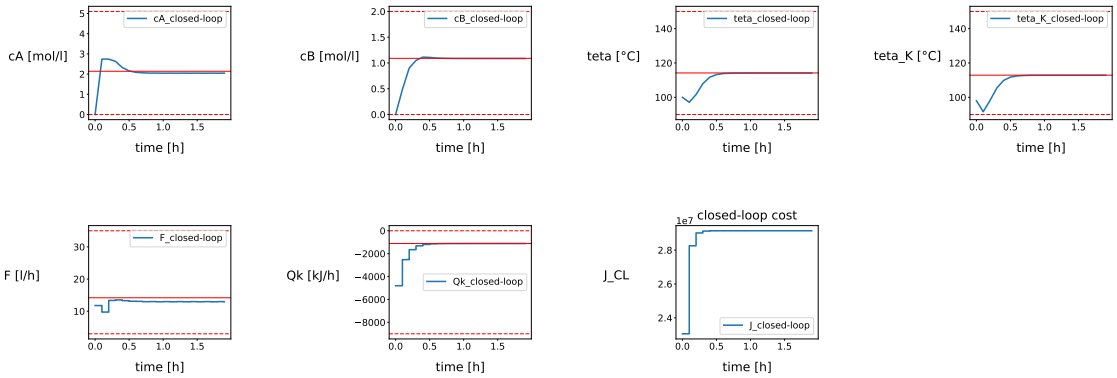

Figure A.99: Alternating Constant Partitioning MPC with no uncertainty for CSTR with N_s = 24 and ordering [1,2,3,4],[1,2,3,4]

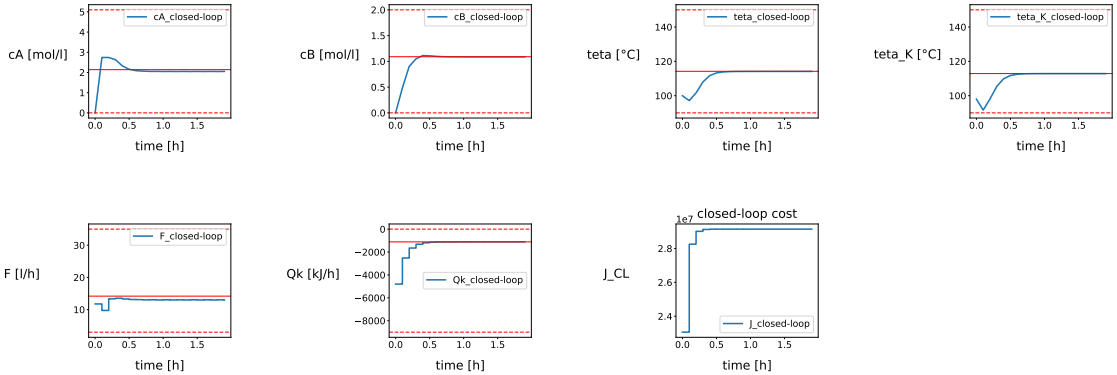

Figure A.100: Alternating Constant Partitioning MPC with no uncertainty for CSTR with N_s = 32 and ordering [1,2,3,4],[1,2,3,4]

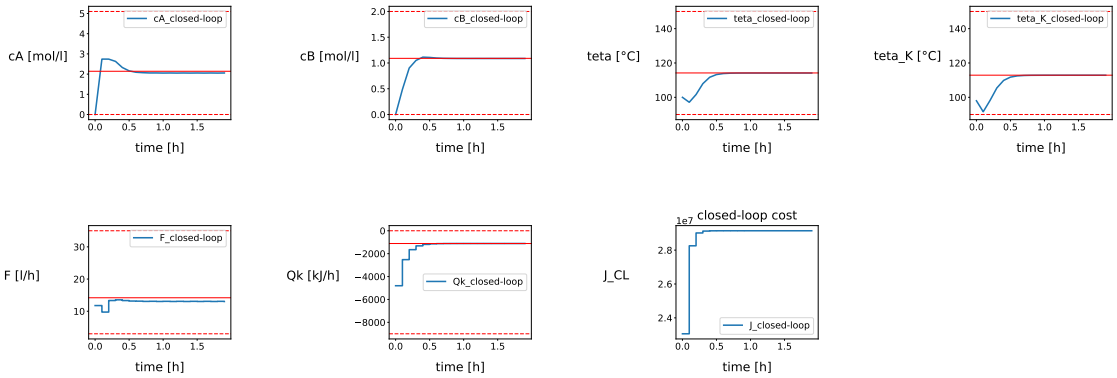

Figure A.101: Alternating Constant Partitioning MPC with no uncertainty for CSTR with N_s = 40 and ordering [1,2,3,4],[1,2,3,4]

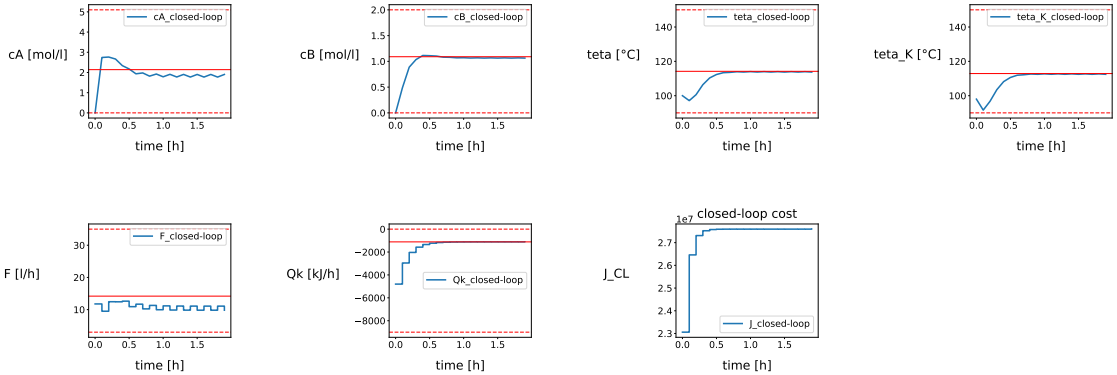

Figure A.102: Alternating Variable Partitioning MPC with no uncertainty for CSTR with N_s = 4(N1=2,N2=6) and ordering [1,2,3,4],[1,2,3,4]

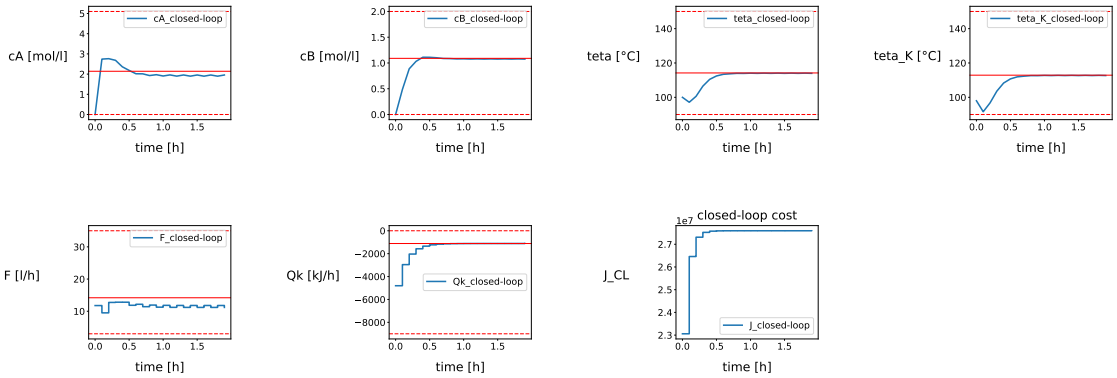

Figure A.103: Alternating Variable Partitioning MPC with no uncertainty for CSTR with N_s = 8(N1=10,N2=6) and ordering [1,2,3,4],[1,2,3,4]

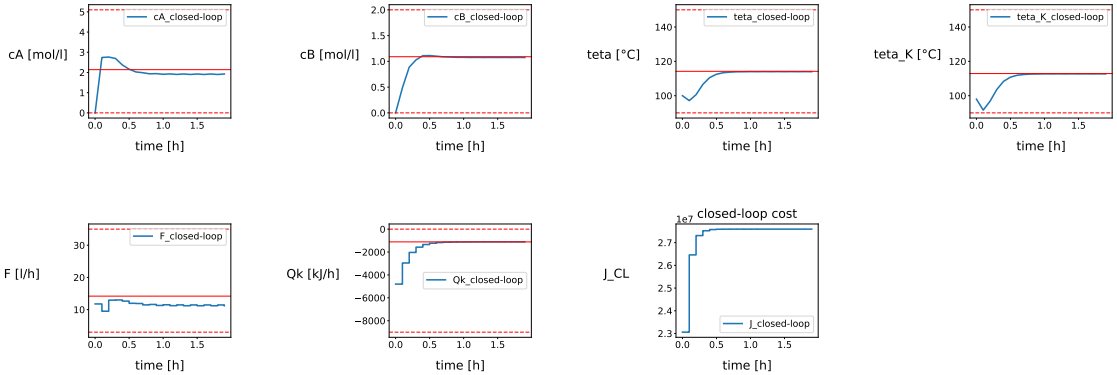

Figure A.104: Alternating Variable Partitioning MPC with no uncertainty for CSTR with N_s = 16(N1=14,N2=18) and ordering [1,2,3,4],[1,2,3,4]

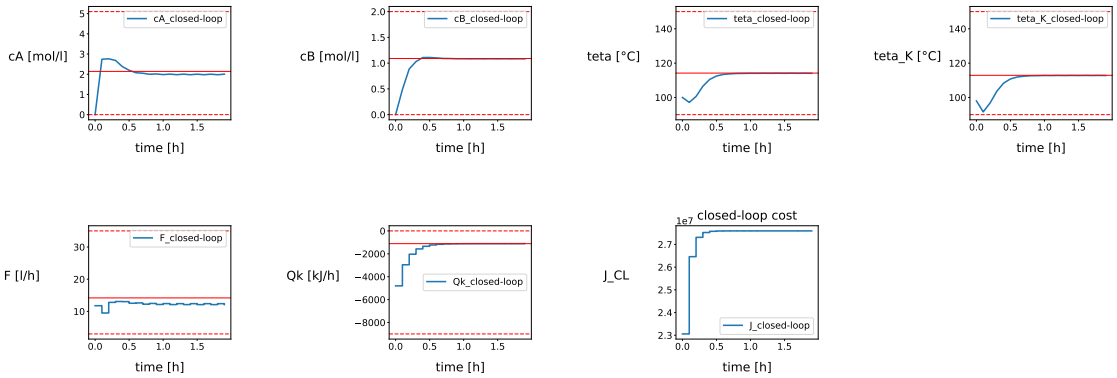

Figure A.105: Alternating Variable Partitioning MPC with no uncertainty for CSTR with N_s = 24(N1=20,N2=28) and ordering [1,2,3,4],[1,2,3,4]

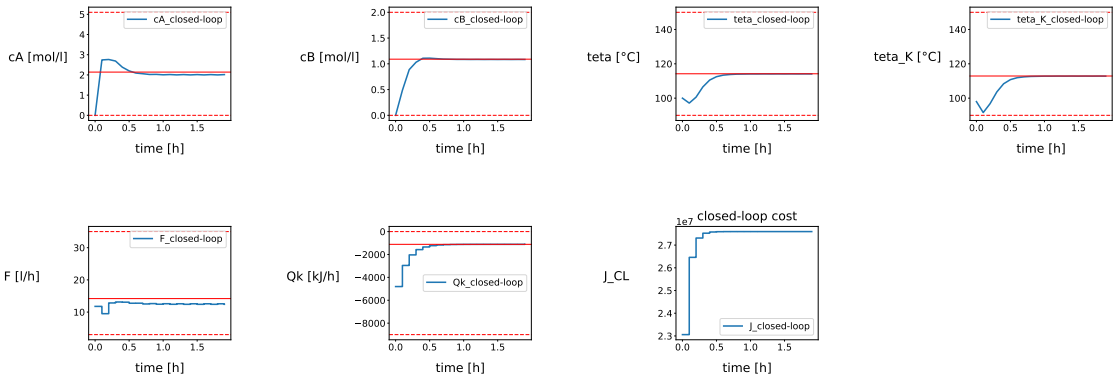

Figure A.106: Alternating Variable Partitioning MPC with no uncertainty for CSTR with N_s = 32(N1=30,N2=34) and ordering [1,2,3,4],[1,2,3,4]

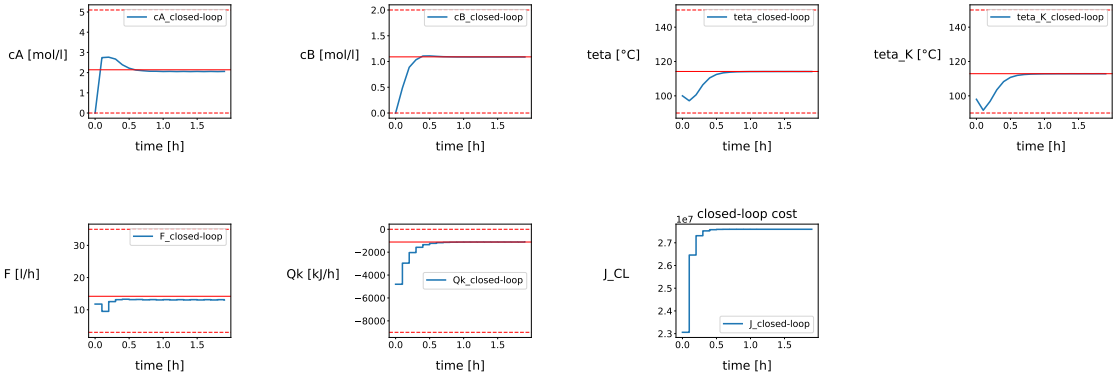

Figure A.107: Alternating Variable Partitioning MPC with no uncertainty for CSTR with N_s = 40(N1=36,N2=44) and ordering [1,2,3,4],[1,2,3,4]

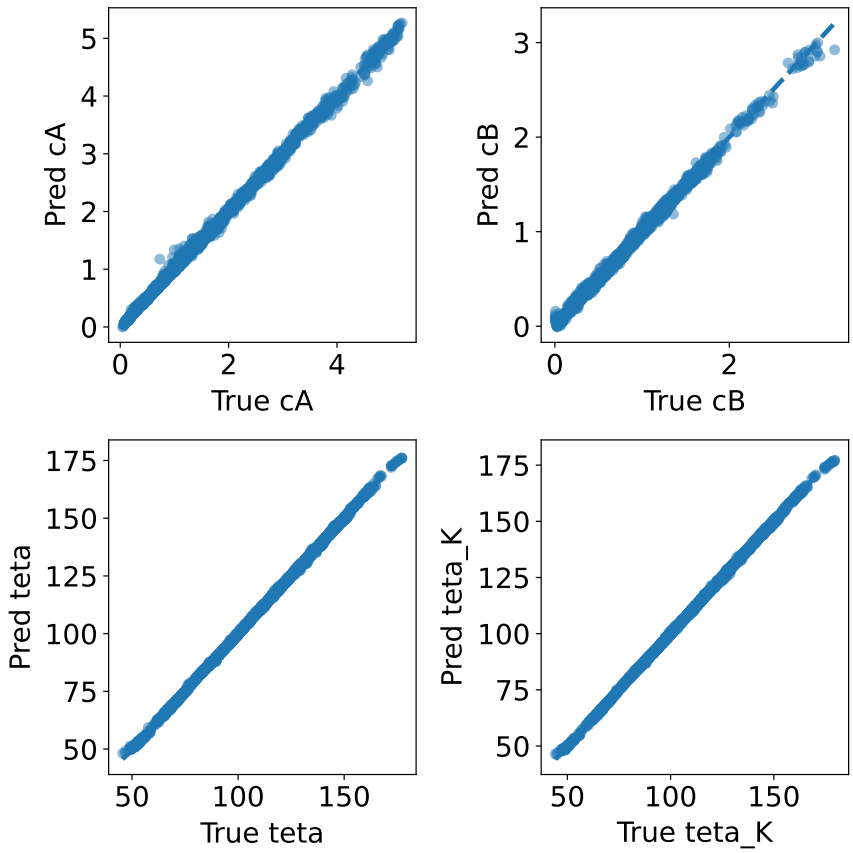

Figure A.108: pairplot for one-step NN

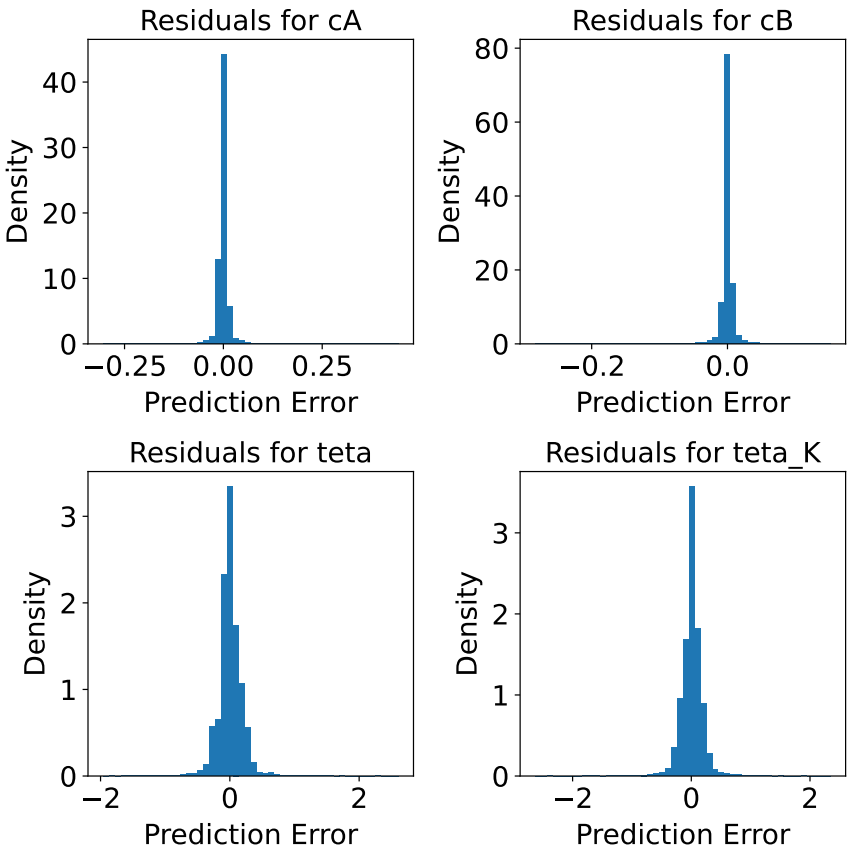

Figure A.109: error-residualplot for one-step NN

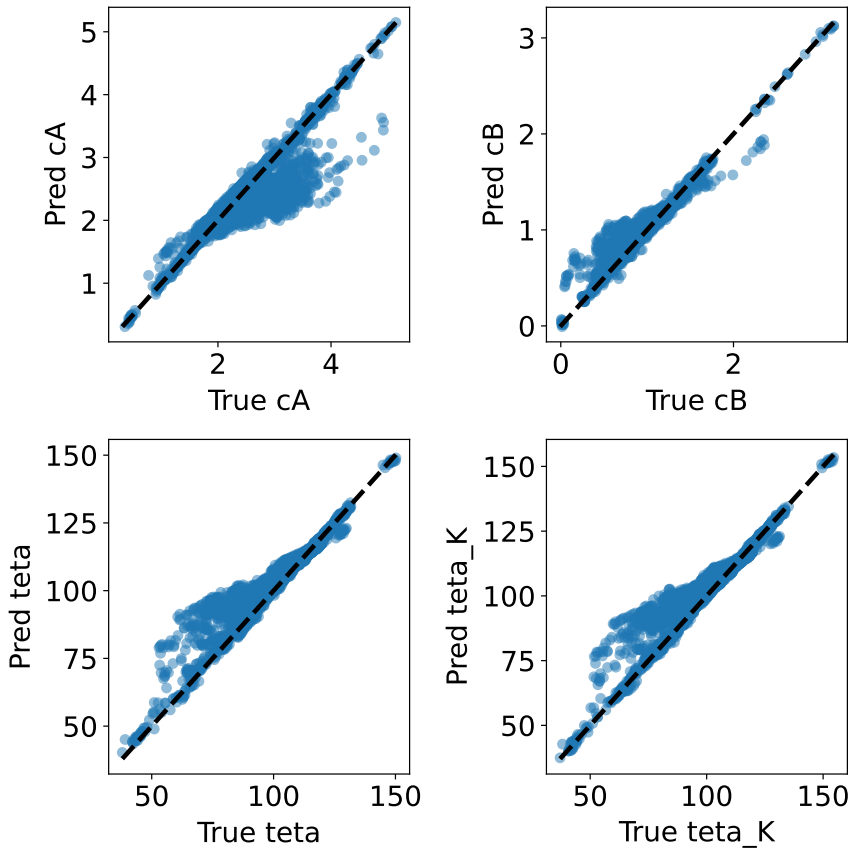

Figure A.110: pairplot for multi-step NN

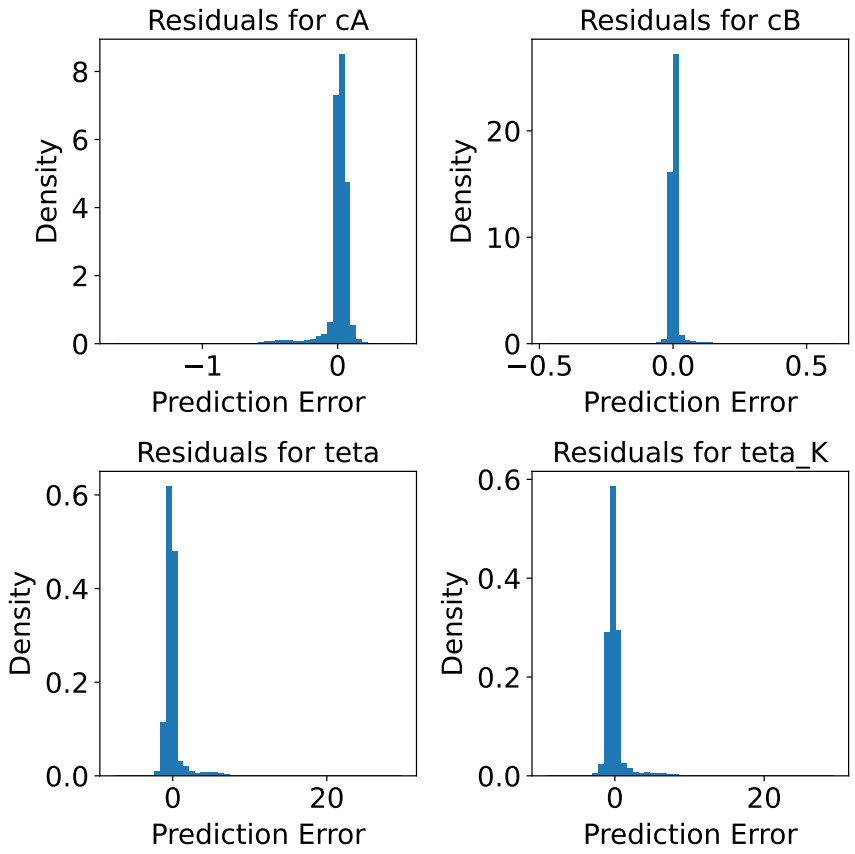

Figure A.111: error-residualplot for multi-step NN

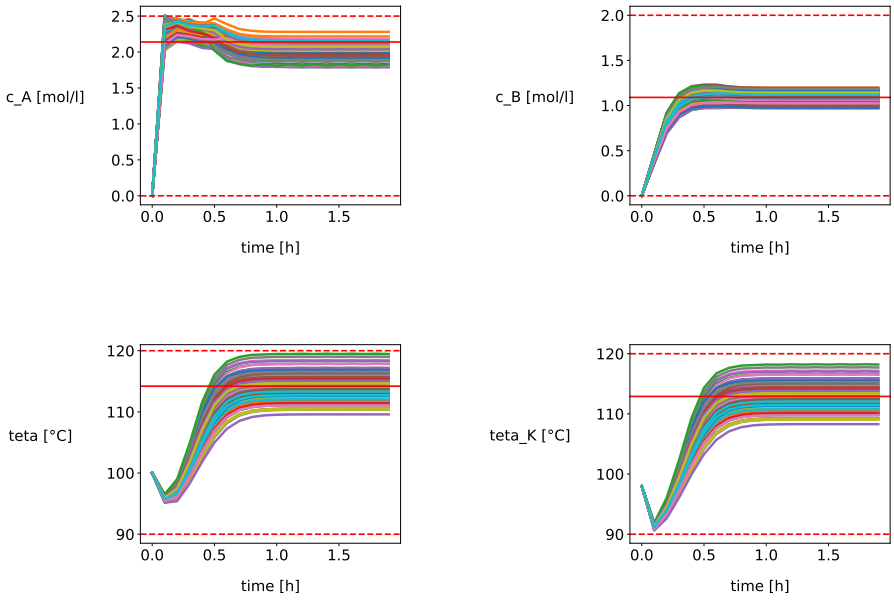

Figure A.112: Alternating Constant Partitioning MPC-Trajectories for 100 random combinations of 11 uncertainty parameters of the CSTR

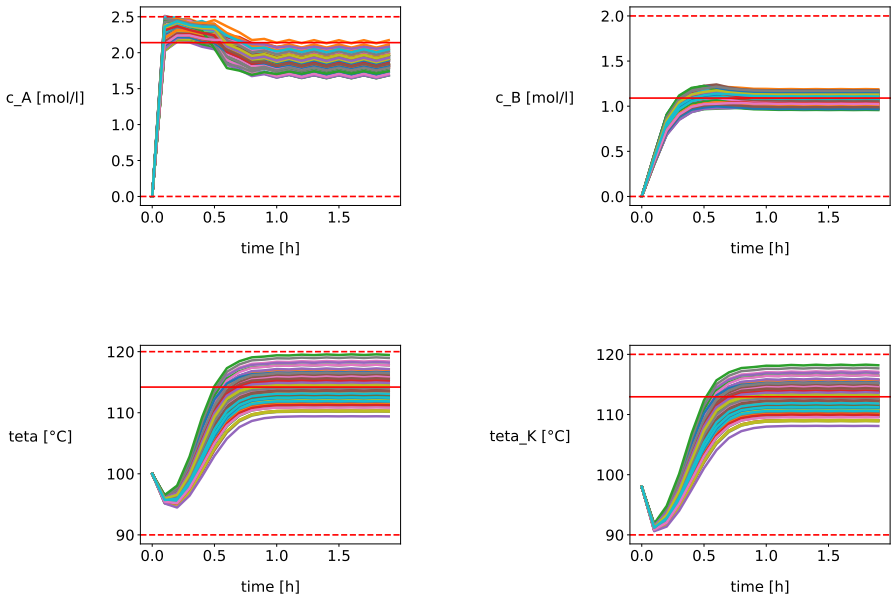

Figure A.113: Alternating Variable Partitioning MPC-Trajectories for 100 random combinations of 11 uncertainty parameters of the CSTR

Appendix B

Formulas and Algorithms

Algorithm 1 Constr_Func$(l, a, h, x_k^{[1:N_s]\pm})$ [15]

for $k = 1$ **to** len(l) **do**
  idx = get_num$(l[k])$
  dim = a[-depth(l)]
  for $s =$ idx **do**
    if $s =$ idx$[1]$ **and** $k = 1$ **then**
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{s^-} - x_{\text{dim}}^{1^-})$
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{1^-} - x_{\text{dim}}^{s^+})$
    else
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{s^-} - x_{\text{dim}}^{\text{idx}[1]-})$
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{\text{idx}[1]-} - x_{\text{dim}}^{s^-})$
    end if
    if $s =$ idx$[-1]$ **and** $k = len(l)$ **then**
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{s^+} - x_{\text{dim}}^{N_s^+})$
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{N_s^+} - x_{\text{dim}}^{s^+})$
    else
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{s^+} - x_{\text{dim}}^{\text{idx}[-1]+})$
      $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{\text{idx}[-1]+} - x_{\text{dim}}^{s+})$
    end if
  end for
  if $k > 1$ **then**
    prev_last = get_num$(l[k-1])[-1]$
    $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{\text{idx}[1]} - x_{\text{dim}}^{\text{prev_last}^+})$
    $h \leftarrow$ concatenate$(h, x_{\text{dim}}^{\text{prev_last}^+} - x_{\text{dim}}^{\text{idx}[1]-})$
  end if
  if depth$(l) > 1$ **then**
    $h \leftarrow$ Constr_Func$(l[k], a, h, x_k^{[1:N_s]\pm})$
  end if
end for
return h

Algorithm 2 Main Construction [15]

$h \leftarrow [\,]$
for $s = 1$ **to** N_s **do**
    $h \leftarrow \text{concatenate}(h, x^{1^-} - x^{s^\pm})$
    $h \leftarrow \text{concatenate}(h, x^{s^\pm} - x^{N_s^+})$
end for
for $i = 1$ **to** N **do**
    $h \leftarrow \text{Constr\_Func}(l, a, h, x_i^{[1:N_s]\pm})$
end for

The algorithms 1 and 2 present the pseudocode used to ensure the correct alignment of subregions within the reachable sets [15]. This alignment ensures that the reachable sets $\mathcal{X}_i$ are completely filled by non-overlapping subregions. The function `CONSTR_FUNC` outputs the required set of alignment constraints h, such that all subregions $x_i^{[1:N_s]\pm}$ within a reachable set $\mathcal{X}_i$ are properly positioned relative to each other.

The function is recursively defined and takes as input a nested list l as described in 2.13, a list of indices a, the constraint vector h, and the collection of subregions represented as hyperrectangles ($x_i^{[1:N_s]\pm}$). The list l defines the partitioning structure, a specifies the order in which the dimensions are partitioned, and h accumulates the constraints required by the ROC.

For example, if l is defined as in the nested list shown in Figure 2.13, and $a = [0, 1, 2]$, then the first element of l corresponds to the first dimension, and all subregions in $\mathcal{S}_1^0$ are aligned along dimension one. The function continues to align the m subregions $\mathcal{S}_i^m$ in each dimension i until the final level is reached, where the terminal subregions are aligned. This alignment must be enforced for each reachable set $\mathcal{X}_i$, and thus the procedure is repeated throughout the prediction horizon N.

B.0.1 Alignment Constraints of Subregions

The following section outlines the alignment constraints that ensure the proper positioning of subregions across all dimensions. These constraints are implemented through the pseudocode presented in Algorithms 1 and 2 [15].

$$0 \leq x_i^{s^\pm} - x_i^{s_0^{(i,m)\pm}} \leq 0, \quad \forall s \in \mathcal{S}_i^m \setminus s_0^{(i,m)},\ \forall i = 1, \ldots, n_x - 1,\ \forall m = 0, \ldots, k_i \tag{B.1}$$

This constraint ensures that in each dimension i, all subregions $s \in \mathcal{S}_i^m$ are aligned with the first subregion $s_0^{(i,m)}$ of the same superregion $\mathcal{S}_i^m$. It guarantees internal consistency within each hyperrectangle partition.

$$0 \leq x_i^{s_0^{(i,m)+}} - x_i^{s_0^{(i,m+1)-}} \leq 0, \qquad \text{if } \exists q : \{s_0^{(i,m)}, s_0^{(i,m+1)}\} \subset \mathcal{S}_{i-1}^q \tag{B.2}$$

This constraint ensures that neighboring subregions are properly aligned. In dimension i, the lower left corner of the $(m+1)$-th superregion is aligned with the upper right corner of the m-th one, assuming both belong to the same parent superregion in the previous (i-1) dimension $\mathcal{S}_{i-1}^{q}$.

$$0 \leq x_i^{s^-} - x_i^{1^-} \leq 0, \qquad \forall s \in \mathcal{S}_i^m, \text{ if } \exists q : s = s_0^{(i-1,q)} \tag{B.3}$$

This ensures that all subregions along the left edge of $\mathcal{X}_i$ are bounded by its global lower corner $x_i^{1^-}$, which acts as a common reference for alignment.

$$0 \leq x_i^{s^+} - x_i^{N_s^+} \leq 0, \qquad \forall s \in \mathcal{S}_i^m, \text{ if } \exists q : s = s_{-1}^{(i-1,q)} \tag{B.4}$$

Similarly, this condition ensures that all subregions on the right edge of $\mathcal{X}_i$ are bounded by its global upper corner $x_i^{N_s^+}$, corresponding to the final subregion.

$$x_s^- - x_s^+ \leq 0, \qquad \forall s \in \mathcal{S} \tag{B.5}$$
$$x_1^- - x_s^- \leq 0, \qquad \forall s \in \mathcal{S} \tag{B.6}$$
$$x_s^+ - x_N^+ \leq 0, \qquad \forall s \in \mathcal{S} \tag{B.7}$$

These final constraints guarantee:

- Each subregion $s \in \mathcal{S}$ is non-degenerate; i.e., the lower bound is less than or equal to the upper bound.
- Each subregion lies fully within the global bounds of the reachable set, bounded below by x_1^-.
- No subregion exceeds the global upper bound x_N^+ of the reachable set.

B.0.2 Full Partitioning and Alternating Constant MPC

$$
\begin{aligned}
\min_{x^{s^\pm}_{[0:N]},\, u^s_{[0:N-1]},\, \forall s \in \mathcal{S}} \quad & J\left(x^{[1:N_s]}_{[1:N]}, u^{[1:N_s]}_{[1:N-1]}, u_0\right) \\
\text{s.t.} \quad & \textbf{1.}\ x_0^{s^\pm} = x_0 \\
& \textbf{2.}\ x_{k+1}^{N_s^+} \geq f(x_k^{s^+}, u_k^s, p^+) \\
& \textbf{3.}\ x_{k+1}^{1^-} \leq f(x_k^{s^-}, u_k^s, p^-) \\
& \textbf{4.}\ [x_k^{1^-},\, x_k^{N_s^+}] \subseteq \mathcal{X} \\
& \textbf{5.}\ u_k^s \in \mathcal{U} \\
& \textbf{6.}\ [x_N^{1^-},\, x_N^{N_s^+}] \subseteq \mathcal{X}_f \\
& \textbf{7.}\ u_0^s = u_0 \\
& \textbf{8.}\ h(x_k^{[1:N_s]^\pm}) \leq 0, \quad \forall k \in \{0, \ldots, N\},\, \forall s \in \mathcal{S}
\end{aligned}
$$

Cost function:

$$
\begin{aligned}
J\left(x^{[1:N_s]}_{[1:N]}, u^{[1:N_s]}_{[1:N-1]}, u_0\right) = & \sum_{i=0}^{N} \sum_{s=1}^{N_s} (x_i^s - x_{\text{ref}})^\top Q (x_i^s - x_{\text{ref}}) \\
& + \sum_{i=1}^{N-1} \sum_{s=1}^{N_s} (u_i^s - u_{i-1}^s)^\top R (u_i^s - u_{i-1}^s) \\
& + \sum_{s=1}^{N_s} (u_0^s - u_0)^\top R (u_0^s - u_0) \\
& + 10 \cdot \sum_{s=1}^{N_s} (x_N^s - x_{\text{ref}})^\top Q (x_N^s - x_{\text{ref}})
\end{aligned}
$$

B.0.3 Alternating Variable Partitioning MPC

$$min := \begin{cases} J_1\left(x^{[0:n_{s_2}]}_{\text{even}[0:N_2]},\, x^{[0:n_{s_1}]}_{\text{odd}[0:N_1]},\, u^{[0:n_{s_2}]}_{\text{even}[0:N_2-1]},\, u^{[0:n_{s_1}]}_{\text{odd}[0:N_1]}\right), & \text{if } N_2 > N_1 (Nmod2 = 1) \\ J_1\left(x^{[0:n_{s_2}]}_{\text{even}[0:N_2]},\, x^{[0:n_{s_1}]}_{\text{odd}[0:N_1]},\, u^{[0:n_{s_2}]}_{\text{even}[0:N_2]},\, u^{[0:n_{s_1}]}_{\text{odd}[0:N_1-1]}\right), & \text{if } N_2 = N_1 (Nmod2 = 0) \end{cases}$$

Subject to:

1. $x_0^{s_2^{\pm}} = x_0$

2. $u_0^{s_2} = u_0$

3.
$$\begin{cases} x^{N_{s_1}^{+}}_{\text{count,odd}+1} \geq f\left(x^{s_2^{+}}_{\text{count,even}},\, u^{s_2}_{\text{count,even}},\, p^{+}\right), & \forall s_2 \in \mathcal{S}_2,\ \text{if count}_{\text{even}} > \text{count}_{\text{odd}}(\, i \bmod 2 = 0) \\ x^{N_{s_2}^{+}}_{\text{count,even}+1} \geq f\left(x^{s_1^{+}}_{\text{count,odd}},\, u^{s_1}_{\text{count,odd}},\, p^{+}\right), & \forall s_1 \in \mathcal{S}_1,\ \text{if count}_{\text{even}} = \text{count}_{\text{odd}}(\, i \bmod 2 = 1) \end{cases}$$

4.
$$\begin{cases} x^{1^{-}}_{\text{count,odd}+1} \leq f\left(x^{s_2^{-}}_{\text{count,even}},\, u^{s_2}_{\text{count,even}},\, p^{-}\right), & \forall s_2 \in \mathcal{S}_2,\ \text{if count}_{\text{even}} > \text{count}_{\text{odd}}(\, i \bmod 2 = 0) \\ x^{1^{-}}_{\text{count,even}+1} \leq f\left(x^{s_1^{-}}_{\text{count,odd}},\, u^{s_1}_{\text{count,odd}},\, p^{-}\right), & \forall s_1 \in \mathcal{S}_1,\ \text{if count}_{\text{even}} = \text{count}_{\text{odd}}(\, i \bmod 2 = 1) \end{cases}$$

5. $u^{[0:n_{s_2}]}_{\text{even}} \in \mathcal{U}, \quad u^{[0:n_{s_1}]}_{\text{odd}} \in \mathcal{U}$

6. $x^{[0:n_{s_2}]}_{\text{even}} \in \mathcal{X}, \quad x^{[0:n_{s_1}]}_{\text{odd}} \in \mathcal{X}$

7.
$$\begin{cases} [x^{1^{-}}_{N_2},\, x^{N_{s_2}^{+}}_{N_2}] \in \mathcal{X}_f, & \text{if } N_2 > N_1, (Nmod2 = 0) \\ [x^{1^{-}}_{N_1},\, x^{N_{s_1}^{+}}_{N_1}] \in \mathcal{X}_f, & \text{if } N_2 = N_1, (Nmod2 = 1) \end{cases}$$

8.
$$\begin{cases} h\left(x^{s_2^{\pm}}_{\text{count,even}}\right) \leq 0, & \forall count, even \in \{0, \ldots, N_2\},\ \forall s_2 \in \mathcal{S}_2 \\ h\left(x^{s_1^{\pm}}_{\text{count,odd}}\right) \leq 0, & \forall count, odd \in \{0, \ldots, N_1\},\ \forall s_1 \in \mathcal{S}_1 \end{cases}$$

$$min := \begin{cases} J_2\left(x^{[0:n_{s_1}]}_{\text{even}[0:N_2]},\, x^{[0:n_{s_2}]}_{\text{odd}[0:N_1]},\, u^{[0:n_{s_1}]}_{\text{even}[0:N_2-1]},\, u^{[0:n_{s_2}]}_{\text{odd}[0:N_1]}\right), & \text{if } N_2 > N_1 (Nmod2 = 1) \\ J_2\left(x^{[0:n_{s_1}]}_{\text{even}[0:N_2]},\, x^{[0:n_{s_2}]}_{\text{odd}[0:N_1]},\, u^{[0:n_{s_1}]}_{\text{even}[0:N_2]},\, u^{[0:n_{s_2}]}_{\text{odd}[0:N_1-1]}\right), & \text{if } N_2 = N_1 (Nmod2 = 0) \end{cases}$$

Subject to:

1. $x_0^{s_1^{\pm}} = x_0$

2. $u_0^{s_1} = u_0$

3.
$$\begin{cases} x^{N_{s_2}^+}_{\text{count,odd}+1} \geq f\left(x^{s_1^+}_{\text{count,even}},\, u^{s_1}_{\text{count,even}},\, p^+\right), & \forall s_1 \in \mathcal{S}_1,\ \text{if count}_{\text{even}} > \text{count}_{\text{odd}}(\, i \bmod 2 = 0) \\ x^{N_{s_1}^+}_{\text{count,even}+1} \geq f\left(x^{s_2^+}_{\text{count,odd}},\, u^{s_2}_{\text{count,odd}},\, p^+\right), & \forall s_2 \in \mathcal{S}_2,\ \text{if count}_{\text{even}} = \text{count}_{\text{odd}}(\, i \bmod 2 = 1) \end{cases}$$

4.
$$\begin{cases} x^{1^-}_{\text{count,odd}+1} \leq f\left(x^{s_1^-}_{\text{count,even}},\, u^{s_1}_{\text{count,even}},\, p^-\right), & \forall s_1 \in \mathcal{S}_1,\ \text{if count}_{\text{even}} > \text{count}_{\text{odd}}(\, i \bmod 2 = 0) \\ x^{2^-}_{\text{count,even}+1} \leq f\left(x^{s_2^-}_{\text{count,odd}},\, u^{s_2}_{\text{count,odd}},\, p^-\right), & \forall s_2 \in \mathcal{S}_2,\ \text{if count}_{\text{even}} = \text{count}_{\text{odd}}(\, i \bmod 2 = 1) \end{cases}$$

5. $u^{[0:n_{s_1}]}_{\text{even}} \in \mathcal{U}, \quad u^{[0:n_{s_2}]}_{\text{odd}} \in \mathcal{U}$

6. $x^{[0:n_{s_1}]}_{\text{even}} \in \mathcal{X}, \quad x^{[0:n_{s_2}]}_{\text{odd}} \in \mathcal{X}$

7.
$$\begin{cases} [x^{1^-}_{N_2},\, x^{N_{s_1}^+}_{N_2}] \in \mathcal{X}_f, & \text{if } N_2 > N_1, (Nmod2 = 0) \\ [x^{1^-}_{N_1},\, x^{N_{s_2}^+}_{N_1}] \in \mathcal{X}_f, & \text{if } N_2 = N_1, (Nmod2 = 1) \end{cases}$$

8.
$$\begin{cases} h\left(x^{s_1^{\pm}}_{\text{count,even}}\right) \leq 0, & \forall count, even \in \{0, \ldots, N_2\},\ \forall s_1 \in \mathcal{S}_1 \\ h\left(x^{s_2^{\pm}}_{\text{count,odd}}\right) \leq 0, & \forall count, odd \in \{0, \ldots, N_1\},\ \forall s_2 \in \mathcal{S}_1 \end{cases}$$

Section B.0.2 defines the ROC $P^{\text{full},N}\text{CL}(x_0)$ for full partitioning MPC applied to a discrete-time system. The optimization variables are $x[0:N]^{s^{\pm}}$ and $u^s_{[0:N-1]}$, where $x^{s^{\pm}}_{[0:N]}$ represents the lower ($-$) and upper ($+$) bounds of all subregions s across the N reachable sets, and $u^s_{[0:N-1]}$ are the control inputs assigned to each subregion at each prediction step. The uncertainty set $\mathcal{P}$, as defined in 2.7, is assumed to be compact. In the case of monotone systems, it suffices to evaluate only the extreme points p^- and p^+ to conservatively overapproximate the reachable sets as hyperrectangles. Individual constraints are described as follows:

Constraint 1: Initializes $\mathcal{X}_0$ as the known initial state x_0, by setting all N_s subregions in $\mathcal{X}_0$ equal to x_0.

Constraints 2 and 3: As shown in figure 2.15, the bounding principle using monotonicity. All propagated subregions in $\mathcal{X}k$ are enclosed by the lower left corner $xk + 1^{1^-}$ and the upper right corner $x^{N_s^+}_{k+1}$ of $\mathcal{X}_{k+1}$.

Constraint 4: Ensures that each reachable set $\mathcal{X}_k \in \mathcal{X}$.

Constraint 5: Guarantees that $u_k^s \in \mathcal{U}$, the admissible input set.

Constraint 6: Ensures recursive feasibility by enforcing that the final reachable set $\mathcal{X}_N$ lies within $\mathcal{X}_f$, bounded by its upper right corner $x_N^{N_s^+}$ and lower left corner $x_N^{1^-}$. Here, $\mathcal{X}_f$ represents a one-step RCIS in the case of full partitioning, and a two-step RCIS in the case of alternating partitioning.

Constraint 7: Ensures that only one control input u_0^s is assigned to the initial state x_0.

Constraint 8: Encodes the partitioning constraints for each reachable set $\mathcal{X}_k$. These include all constraints that govern the alignment of the subregion and the partitioning structure. The constraints in 8 are expressed as inequality functions h, defining whether the same partitioning is used as in the full partitioning MPC or whether alternating-partitioning MPC is applied. In the alternating case, partitioned dimensions and the number of subregions per dimension can vary periodically.

These alignment constraints can be constructed using the `CONSTR_FUNC` routine by providing the switching pattern with a switching dimension ordering a in each prediction step, enabling dimension-wise alternating partitioning. With these adjustments, constraints 1 to 8 formulate the problem $P_{\text{CL}}^{N,\text{alt\_const}}(x_0)$. The difference compared to full partitioning lies in the constraints given in Constraint 8.

To ensure correct alignment in alternating constant partitioning, two different ROC problems are solved in alternating steps. These differ in Constraint 8 so that the subregions of two consecutive closed-loop steps can be aligned, allowing the solution from one step to serve as a feasible initial guess in the next as shown in Figure 2.19.

This is achieved by partitioning $\tilde{\mathcal{X}}_1$ in the same dimensions, in the same order and frequency as $\mathcal{X}_2$ from the previous simulation step. This shifting of $\mathcal{X}i$ to $\tilde{\mathcal{X}}i+1$ can be continued up to $\mathcal{X}_N$, where the two-step RCIS comes into play.

The formulation $P_{\text{CL}}^{N,\text{alt\_var}}(x_0)$ in section B.0.2 and B.0.3 follows the same principle. The periodic switching between two alternating-constant formulations, given in Section B.0.3 ensures that subregions in consecutive steps are aligned and feasible trajectories can be constructed across the prediction steps. The argumentation is identical to that for alternating constant partitioning.

Since the number of subregions N_s in $\mathcal{X}_i$ is no longer constant across steps, the horizon N is split into N_2 even and N_1 odd steps. The optimization variables become $x^{\text{even}}, x^{\text{odd}}, u^{\text{even}}, u^{\text{odd}}$, respectively.

B.0.4 Computation of max-RCIS

$$
\begin{aligned}
\max \quad & V\left(x_{[1:N]}^{[1:N_s]^{\pm}}\right) \\
\text{with} \quad & V\left(x_{[1:N]}^{[1:N_s]^{\pm}}\right) = \prod_{k=1}^{N_{\text{RCIS}}} \prod_{i=1}^{n_x} \left(x_{i,k}^{N_s^+} - x_{i,k}^{1^-}\right) \\
\text{w.r.t.} \quad & x_i^{s^+},\, x_i^{s^-},\, u_i^s \quad \forall s \in \mathbb{N}_s, \forall i \in \mathbb{N} \text{s.t.} \\
& [x_i^{1^-},\, x_i^{N_s^+}] \in \mathcal{X}, \quad \forall i \in \mathbb{N} \\
& u_i^s \in \mathcal{U}, \quad \forall s \mathbb{N}_s,\, \forall i \in \mathbb{N} \\
& h\left(x_i^{[1:N_s]^{\pm}}\right) \leq 0, \quad \forall i \in \mathbb{N} \\
& x_{i+1}^{N_s^+} \geq f\left(x_i^{s^{\pm}}, u_i^s, p^{\pm}\right) \geq x_{i+1}^{1^-}, \quad \forall s \in \mathbb{N}_s,\, \forall i \in \mathbb{N}_N - 1 \\
& x_1^{N_s^+} \geq f\left(x_N^{s^{\pm}}, u_N^s, p^{\pm}\right) \geq x_1^{1^-}, \quad \forall s \in \mathbb{N}
\end{aligned}
$$

Appendix C

Datasheets

Table C.1: Model parameters with their uncertainties for the CSTR [8]

Parameter Name	**Symbol**	**Value**
Inlet concentration of A	c_{A0}	(5.1 ± 0.6) mol A/l
Reaction constant (reaction 1)	$k_{1,0}$	$(1.287 \pm 0.04) \cdot 10^{12}$ h^{-1}
Reaction constant (reaction 2)	$k_{2,0}$	$(1.287 \pm 0.04) \cdot 10^{12}$ h^{-1}
Reaction constant (reaction 3)	$k_{3,0}$	$(1.287 \pm 0.04) \cdot 10^{9}$ l/(mol A · h)
Activation energy (reaction 1)	E_{A12}/R	9758.3 K
Activation energy (reaction 2)	E_{A12}/R	9758.3 K
Activation energy (reaction 3)	E_{A3}/R	8560.0 K
Reaction enthalpy (reaction 1)	ΔH_{AB}	(4.2 ± 2.36) kJ/mol A
Reaction enthalpy (reaction 2)	ΔH_{BC}	(-11.0 ± 1.92) kJ/mol B
Reaction enthalpy (reaction 3)	ΔH_{AD}	(-41.85 ± 1.41) kJ/mol C
Density	ρ	$(0.9342 \pm 4 \cdot 10^{-4})$ kg/l
Heat transfer coefficient (cooling jacket)	k_w	(4032 ± 120) kJ/(h · m^2 · K)
Surface area (cooling jacket)	A_r	0.215 m^2
Reactor volume	V_r	10.0 l
Coolant mass	m_k	5.0 kg
Heat capacity (coolant)	$C_{p,k}$	(2.0 ± 0.05) kJ/(kg · K)
Heat capacity (reactant mixture)	C_p	(3.01 ± 0.04) kJ/(kg · K)

Table C.2: Setpoints for the CSTR [35]

Setpoint Name	**Symbol**	**Value**
Inlet concentration of A in the feed	$c_{A0,s}$	5.1 mol A/l
Inlet feed temperature	ϑ_0	104.9 °C
Steady-state feed	F_s	14.19 h^{-1}
Steady-state cooling rate	Q_s	-1113.5 kJ/h
Setpoint of A	$c_{A,s}$	2.14 mol/l
Setpoint of B	$c_{B,s}$	1.09 mol/l
Setpoint of reactor temperature	ϑ_s	114.2 °C
Setpoint of cooling jacket temperature	$\vartheta_{k,s}$	112.9 °C

Table C.3: Overview of some monotone NN architectures and their training, validation, and test MSE for Singlepoint-predictions. N_1 denotes the layers before the input of u enters the NN, and N_2 the layers from and including the entry layer of u.

Architecture	**Training MSE**	**Validation MSE**	**Test MSE**
$N_1 = 4$, $N_2 = 6$, hidden = 128	0.000896	0.000957	0.000894
$N_1 = 1$, $N_2 = 1$, hidden = 128	0.025178	0.026619	0.024223
$N_1 = 2$, $N_2 = 1$, hidden = 128	0.013202	0.014257	0.012877
$N_1 = 2$, $N_2 = 2$, hidden = 128	0.000738	0.000786	0.000743
$N_1 = 2$, $N_2 = 4$, hidden = 128	0.000283	0.000323	0.000277

Appendix D

Acknowledgements

I would like to thank all those who supported me during the writing of this thesis. In particular, I acknowledge the assistance of DeepL Write for linguistic refinement during the English revision process.

Eidesstattliche Versicherung

(Affidavit)

Kalenkiewicz,Thomas
Name, Vorname
(surname, first name)

202520
Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

☒ Masterarbeit
(Master's thesis)

Titel
(Title)

Investigating the effect of set partitioning strategies on the performance of robust model predictive controllers for monotone systems

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

09.06.2025
Ort, Datum
(place, date)

Kalenkiewicz
Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund,09.06.2025
Ort, Datum
(place, date)

Kalenkiewicz
Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**

Bibliography

[1] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," en, *Computers and Chemical Engineering*, no. 23, pp. 667–682, Sep. 1998.

[2] M. G. Forbes, R. S. Patwardhan, H. Hamadah, and R. B. Gopaluni, "Model Predictive Control in Industry: Challenges and Opportunities," en, *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 531–538, 2015. doi: 10.1016/j.ifacol.2015.09.022. (visited on 05/06/2025).

[3] P. D. Doma□ski, "Performance Assessment of Predictive Control—A Survey," en, *Algorithms*, vol. 13, no. 4, p. 97, Apr. 2020. doi: 10.3390/a13040097. (visited on 05/06/2025).

[4] S. Kouro, M. A. Perez, J. Rodriguez, A. M. Llor, and H. A. Young, "Model Predictive Control: MPC's Role in the Evolution of Power Electronics," en, *IEEE Industrial Electronics Magazine*, vol. 9, no. 4, pp. 8–21, Dec. 2015. doi: 10.1109/MIE.2015.2478920. (visited on 05/06/2025).

[5] Y. Yao and D. K. Shekhar, "State of the art review on model predictive control (MPC) in Heating Ventilation and Air-conditioning (HVAC) field," en, *Building and Environment*, vol. 200, p. 107 952, Aug. 2021. doi: 10.1016/j.buildenv.2021.107952. (visited on 05/06/2025).

[6] S. Lucia, J. A. Andersson, H. Brandt, M. Diehl, and S. Engell, "Handling uncertainty in economic nonlinear model predictive control: A comparative case study," en, *Journal of Process Control*, vol. 24, no. 8, pp. 1247–1259, Aug. 2014. doi: 10.1016/j.jprocont.2014.05.008. (visited on 05/06/2025).

[7] J. Adamek, M. Heinlein, L. Lüken, and S. Lucia, "Deterministic Safety Guarantees for Learning-Based Control of Monotone Nonlinear Systems Under Uncertainty," en, *IEEE Control Systems Letters*, vol. 8, pp. 1030–1035, 2024. doi: 10.1109/LCSYS.2024.3407635. (visited on 05/06/2025).

[8] K.-U. Klatt and S. Engell, "Gain-scheduling trajectory control of a continuous stirred tank reactor," en, *Computers & Chemical Engineering*, vol. 22, no. 4-5, pp. 491–502, Jan. 1998. doi: 10.1016/S0098-1354(97)00261-5. (visited on 05/06/2025).

[9] S. Coogan, "Mixed Monotonicity for Reachability and Safety in Dynamical Systems," en, in *2020 59th IEEE Conference on Decision and Control (CDC)*, Jeju, Korea (South): IEEE, Dec. 2020, pp. 5074–5085, isbn: 978-1-7281-7447-1. doi: 10.1109/CDC42340.2020.9304391. (visited on 05/06/2025).

[10] P. Airikka, "Advanced control methods for industrial process control," en, *Computing and Control Engineering*, vol. 15, no. 3, pp. 18–23, Jun. 2004. doi: 10.1049/cce:20040303. (visited on 05/12/2025).

[11] H. Chen and F. Allgöwer, "Nonlinear Model Predictive Control Schemes with Guaranteed Stability," en, in *Nonlinear Model Based Process Control*, R. Berber and C. Kravaris, Eds., Also available as ISBN 978-94-011-5094-1, Dordrecht: Springer Netherlands, 1998, pp. 465–494, isbn: 978-94-010-6140-7. doi: 10.1007/978-94-011-5094-1_16. [Online]. Available: http://link.springer.com/10.1007/978-94-011-5094-1_16 (visited on 05/06/2025).

[12] J. Baillieul and T. Samad, Eds., *Encyclopedia of Systems and Control*, en. London: Springer London, 2013, isbn: 978-1-4471-5102-9. doi: 10.1007/978-1-4471-5102-9. (visited on 05/14/2025).

[13] M. Abate, M. Dutreix, and S. Coogan, "Tight Decomposition Functions for Continuous-Time Mixed-Monotone Systems With Disturbances," en, *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 139–144, Jan. 2021. doi: 10.1109/LCSYS.2020.3001085. (visited on 05/06/2025).

[14] M. Heinlein, S. Subramanian, M. Molnar, and S. Lucia, "Robust MPC approaches for monotone systems," en, in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Cancun, Mexico: IEEE, Dec. 2022, pp. 2354–2360, isbn: 978-1-6654-6761-2. doi: 10.1109/CDC51059.2022.9992502. (visited on 05/06/2025).

[15] M. Heinlein, S. Subramanian, and S. Lucia, "Robust Model Predictive Control Exploiting Monotonicity Properties," en, *IEEE Transactions on Automatic Control*, pp. 1–8, 2025. doi: 10.1109/TAC.2025.3558137. (visited on 05/06/2025).

[16] D. Limon, J. Bravo, T. Alamo, and E. Camacho, "Robust MPC of constrained nonlinear systems based on interval arithmetic," en, *IEE Proceedings - Control Theory and Applications*, vol. 152, no. 3, pp. 325–332, May 2005. doi: 10.1049/ip-cta:20040480. (visited on 05/06/2025).

[17] B. Kouvaritakis and M. Cannon, *Model Predictive Control*, en. Cham: Springer International Publishing, 2016, isbn: 978-3-319-24851-6. doi: 10.1007/978-3-319-24853-0. (visited on 05/06/2025).

[18] S. Lucia, "Robust Multi-stage Nonlinear Model Predictive Control," Englisch, PhD Thesis, TU Dortmund, Dortmund, 2015. [Online]. Available: https://ifatwww.et.uni-magdeburg.de/syst/about_us/people/lucia/PhD_Dissertation_Lucia_no_CV.pdf.

[19] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*, en, 2nd edition. Madison, Wisconsin: Nob Hill Publishing, 2017, isbn: 978-0-9759377-3-0.

[20] H. Chen and F. Allgower, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," en, in *1997 European Control Conference (ECC)*, Brussels: IEEE, Jul. 1997, pp. 1421–1426, isbn: 978-3-9524269-0-6. doi: 10.23919/ECC.1997.7082300. (visited on 06/02/2025).

[21] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, en. Philadelphia, Pa: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2010, isbn: 978-0-89871-702-0.

[22] S. Lucia, T. Finkler, and S. Engell, "Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty," en, *Journal of Process Control*, vol. 23, no. 9, pp. 1306–1319, Oct. 2013. doi: 10.1016/j.jprocont.2013.08.008. (visited on 05/06/2025).

[23] M. Ciocca, P.-B. Wieber, and T. Fraichard, "Strong recursive feasibility in model predictive control of biped walking," en, in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Birmingham: IEEE, Nov. 2017, pp. 730–735, isbn: 978-1-5386-4678-6. doi: 10.1109/HUMANOIDS.2017.8246953. (visited on 05/06/2025).

[24] Y. Song, M. P. D. Sai, and H. Yu, "Zonotope-based nonlinear model order reduction for fast performance bound analysis of analog circuits with multiple-interval-valued parameter variations," en, in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, Dresden, Germany: IEEE Conference Publications, 2014, pp. 1–6, isbn: 978-3-9815370-2-4. doi: 10.7873/DATE2014.024. (visited on 06/03/2025).

[25] P.-J. Meyer, A. Devonport, and M. Arcak, *Interval Reachability Analysis: Bounding Trajectories of Uncertain Systems with Boxes for Control and Verification*, en. Cham: Springer International Publishing, 2021, isbn: 978-3-030-65109-1. doi: 10.1007/978-3-030-65110-7. (visited on 06/03/2025).

[26] L. Yang and N. Ozay, "Tight decomposition functions for mixed monotonicity," en, in *2019 IEEE 58th Conference on Decision and Control (CDC)*, Nice, France: IEEE, Dec. 2019, pp. 5318–5322, isbn: 978-1-7281-1398-2. doi: 10.1109/CDC40024.2019.9030065. (visited on 05/06/2025).

[27] F. Fiedler, B. Karg, L. Lüken, *et al.*, "Do-mpc: Towards FAIR nonlinear and robust model predictive control," en, *Control Engineering Practice*, vol. 140, p. 105 676, Nov. 2023. doi: 10.1016/j.conengprac.2023.105676. (visited on 06/05/2025).

[28] D. Angeli and E. Sontag, "Monotone control systems," en, *IEEE Transactions on Automatic Control*, vol. 48, no. 10, pp. 1684–1698, Oct. 2003. doi: 10.1109/TAC.2003.817920. (visited on 05/06/2025).

[29] C. Kallies, M. Schliemann-Bullinger, R. Findeisen, S. Lucia, and E. Bullinger, "Monotonicity of Kinetic Proofreading," en, *IFAC-PapersOnLine*, vol. 49, no. 26, pp. 306–311, 2016. doi: 10.1016/j.ifacol.2016.12.144. (visited on 06/03/2025).

[30] Sergio Lucia, M. Schliemann-Bullinger, R. Findeisen, and E. Bullinger, "A Set-Based Optimal Control Approach for Pharmacokinetic/Pharmacodynamic Drug Dosage Design," en, *IFAC-PapersOnLine,Elsevier*, vol. 49, no. 7, pp. 797–802, 2016. doi: 10.1016/j.ifacol.2016.07.286. (visited on 05/06/2025).

[31] S. Lucia, R. Paulen, and S. Engell, "Multi-stage Nonlinear Model Predictive Control with verified robust constraint satisfaction," en, in *53rd IEEE Conference on Decision and Control*, Los Angeles, CA, USA: IEEE, Dec. 2014, pp. 2816–2821, isbn: 978-1-4673-6090-6. doi: 10.1109/CDC.2014.7039821. (visited on 05/06/2025).

[32] S. Subramanian, S. Lucia, R. Paulen, and S. Engell, "Tube-enhanced Multi-stage MPC for Flexible Robust Control of Constrained Linear Systems with Additive and Parametric Uncertainties," en, *International Journal of Robust and Nonlinear Control*, vol. 31, no. 9, pp. 4458–4487, Jun. 2021. doi: 10.1002/rnc.5486. (visited on 06/03/2025).

[33] P.-J. Meyer, A. Girard, and E. Witrant, "Controllability and invariance of monotone systems for robust ventilation automation in buildings," en, in *52nd IEEE Conference on Decision and Control*, Firenze: IEEE, Dec. 2013, pp. 1289–1294, isbn: 978-1-4673-5717-3. doi: 10.1109/CDC.2013.6760060. (visited on 05/06/2025).

[34] P.-J. Meyer, A. Girard, and E. Witrant, "Robust controlled invariance for monotone systems: Application to ventilation regulation in buildings," en, *Automatica*, vol. 70, pp. 14–20, Aug. 2016. doi: 10.1016/j.automatica.2016.03.004. (visited on 05/06/2025).

[35] H. Chen, A. Kremling, and F. Allgower, "Nonlinear Predictive Control of a Benchmark CSTR," en, *European Control Conference 1995*, Jan. 1995.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," en, *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. doi: 10.1038/nature14539. (visited on 05/08/2025).

[37] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville*. MIT Press, 2016.

[38] S. You, D. Ding, K. Canini, J. Pfeifer, and M. Gupta, "Deep Lattice Networks and Partial Monotonic Functions," en, *31st Conference on Neural Information Processing Systems*, 2017.

[39] D. Runje and S. M. Shankaranarayana, *Constrained Monotonic Neural Networks*, en, arXiv:2205.11775 [cs], May 2023. doi: 10.48550/arXiv.2205.11775. [Online]. Available: http://arxiv.org/abs/2205.11775 (visited on 06/06/2025).

[40] B. Amos, L. Xu, and J. Z. Kolter, "Input Convex Neural Networks," en, *Proceedings of the 34 th International Conference on Machine Learning*, 2017.

[41] A. Gupta, N. Shukla, L. Marla, A. Kolbeinsson, and K. Yellepeddi, *How to Incorporate Monotonicity in Deep Networks While Preserving Flexibility?* en, arXiv:1909.10662 [cs], Dec. 2019. doi: 10.48550/arXiv.1909.10662. [Online]. Available: http://arxiv.org/abs/1909.10662 (visited on 06/06/2025).

[42] H. Daniels and M. Velikova, "Monotone and Partially Monotone Neural Networks," en, *IEEE Transactions on Neural Networks*, vol. 21, no. 6, pp. 906–917, Jun. 2010. doi: 10.1109/TNN.2010.2044803. (visited on 05/06/2025).

[43] J. Sill and Y. S. Abu-Mostafa, "Monotonicity Hints," en,

[44] L. Biegler and V. Zavala, “Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization,” en, *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, Mar. 2009. doi: 10.1016/j.compchemeng.2008.08.006. (visited on 06/08/2025).

[45] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi: A software framework for nonlinear optimization and optimal control,” en, *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Mar. 2019. doi: 10.1007/s12532-018-0139-4. (visited on 06/08/2025).

[46] B. Karg, T. Alamo, and S. Lucia, “Probabilistic performance validation of deep learning□based robust NMPC controllers,” en, *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8855–8876, Dec. 2021. doi: 10.1002/rnc.5696. (visited on 05/06/2025).