Advances in Decision-making Under Uncertainty with Nonlinear Model Predictive Control

Submitted in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

CHEMICAL ENGINEERING

by

ZHOU (JOYCE) YU

B.S., CHEMICAL ENGINEERING, WASHINGTON UNIVERSITY IN ST. LOUIS
M.ENG., ENERGY, ENVIRONMENTAL AND CHEMICAL ENGINEERING,
WASHINGTON UNIVERSITY IN ST. LOUIS

Carnegie Mellon University

Pittsburgh, PA

December 2019

Acknowledgments

I feel extremely grateful to be given this opportunity to write this thesis. First of all, I would like to thank my advisor Dr. Larry Biegler. It has been a priviledge to work with Larry for the past four and a half years. Larry has been a constant source of ideas and support; his unexhausive energy, unpretentious authenticity, and unwavering dedication have been more than inspiring. He also exemplifies kindness and is *überzeugend* by his own actions; there are unlimited things one can learn from him. Thank you, Larry, for being a great mentor and a role model; I feel very fortunate to be your student.

My gratitude also goes to the members of my commitee, Dr. Ignacio Grossmann, Dr. Erik Ydstie, and Dr. Javier Peña. Not only their insights and guidance shed light on this thesis, they have been one of the key motivations during my PhD. I feel grateful having this opportunity to work with such great minds. My appreciation also goes to all members of faculty and staff for all the help and memorable conversations throughout the years.

I would also take this opportunity to thank many professors/mentors that I've worked with over the years. I thank Dr. Cynthia Lo for introducing me to computational research and also for her timeless mentorship. I thank Dr. Renato Feres for teaching me that math and programming are not only useful but also fun. I would like to thank Dr. Mike Duduković and the CREL group for guiding me through my master's studies and encouraging me to apply for Ph.D. programs; special thanks to Mike who has held a high standard since the early stage of my research career that I could never appreciate enough. I thank Dr. Thomas Badgwell for his industrial perspectives and guidance, and also Exxon-Mobil Research and Engineering for the support during the early phase of my Ph.D. I thank Dr. Chris Atkeson for fruitful discussions and feedbacks, and for constantly inspiring me to think out of the box. I would like to thank Dr. Kevin Zaseck, the control group and the DRAKE group at Toyota Research Institute for their help and support during my summer internship. I thank Kevin for being such a great mentor that allows my endless questions, and I thank both groups for providing an enjoyable and rewarding summer experience.

I thank Department of Chemical Engineering for the financial support of my PhD study.

This work would not be possible without the support of my family. I'm forever indebted to my parents for their unconditional love and support, and for always believing in me even in difficult times. I would also like to thank my grandparents for their encouragement and nurturing throughout all these years. Thanks to all my friends in the US and back in China and everywhere else who have helped me and supported me during this journey. Thank you all for helping me grow into who I am today; this thesis is dedicated to you.

Zhou (Joyce) Yu
Pittsburgh, PA
Dec 2019

Abstract

Model predictive control (MPC) has been a very successful advanced process control technique for many applications especially in process industries because of its ability to handle hard constraints and multiple inputs and outputs. However, the presence of uncertainty deteriorates the performance of nominal MPC, and a robust model predictive control becomes necessary. The existing development of robust (nonlinear) MPC has yet to be widely applied in industry, mainly due to conservatism of the algorithm and also the impracticality of implementation; thereby robust NMPC has been mostly conceptual until a scenario-based robust NMPC recently emerged. A scenario tree is generated to represent the evolution of states with respect to uncertain parameters, and a multistage stochastic programming formulation has been employed to continuously solve for the optimal control action in a moving horizon fashion. This is a good place to start, however, many challenges still remain and need to be addressed. This thesis develops easily implementable robust NMPC strategies which provide performance guarantees and computational efficiency.

One of the major issues of multistage NMPC approaches is computational complexity. Due to the construction of the scenario tree, multistage NMPC models are inevitably larger than their nominal counterparts, and their size grows exponentially with respect to the number of uncertain parameters and the length of robust horizons. To solve this issue, we present an efficient parallelizable advanced-step multistage NMPC (as-msNMPC) approach, which explicitly deals with two types of uncertainty: model parameters and unmeasured noise. The first type is attended to by incorporating multistage scenario trees and the second by applying nonlinear programming (NLP) sensitivity. The framework of as-msNMPC has been demonstrated on two examples with robust performance and significantly faster online computation compared to benchmark methods.

We also conduct a stability analysis for the newly constructed robust NMPC schemes. Based on Lyapunov stability theory, we show that the origin is asymptotically stable under standard NMPC if the model is perfect. And when additive disturbance is present, the system is robustly stable to a neighborhood around the origin. When the system allows both types of uncertainty, we first show that recursive feasibility can be ensured for fully-expanded multistage NMPC under mild assumptions. With both advanced-step and ideal multistage NMPC, we then show that robust stability can be achieved with input-to-state practical stability (ISpS).

Last but not least, a sensitivity-assisted multistage NMPC (samNMPC) algorithm has been proposed to deal with the size of the multistage formulation on a linear algebra level. A block-bordered-diagonal structure of the KKT matrix naturally arises with the multistage NLP problem, and Schur complement decomposition can be performed to decouple scenarios. In this case, we can approximate many scenarios with the solution to the nominal scenario. In addition, we apply a scenario generation technique to determine which scenario is most likely to violate constraints. We then formulate an approximate multistage formulation that has a much smaller problem size. The tracking performance of samNMPC and exact multistage NMPC have been compared and similar performance has been reached with only a fraction of the computational effort of the exact multistage formulation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Decision-making under uncertainty is a growing field that has attracted much attention in recent years. In a dynamic system with model-based approach, one often requires to make an educated and optimized decision promptly within a limited time. In a real-world scenario, perfect information regarding the model is rarely available hence uncertainty occurs naturally. In addition, one should also take into account constraints arising from safety perspectives, physics limitations, and quality requirements. Achieving this goal in an uncertain environment has been a challenging topic in advanced process control.

One of the popular modern control strategies is model predictive control (MPC). MPC has been successful for many applications in the process, automotive and aerospace industries [5], mainly because of its effectiveness in respecting constraints and multiple-input-multiple-output (MIMO) systems. MPC replaces the off-line feedback control *law* with an on-line control *action* by solving an optimization problem. MPC is particularly advantageous when an off-line control law is difficult or impossible to obtain [6]. An excellent review paper on the history of industrial applications of MPC can be found in [5]. In particular, Nonlinear Model Predictive Control (NMPC), the nonlinear counterpart of MPC, is gaining more attention because of its capability to capture detailed nonlinear dynamics of the system throughout the entire state space.

While under suitable conditions NMPC has some inherent robustness [7, 8], its performance is impaired under the influence of uncertainty. More recently, robust MPC has gained much attention, especially for systems that require satisfaction of stability and performance metrics under model variations and noise signals [9]. To meet both robust stability and performance requirements, Bemporad and Morari classify two ways to design a robust MPC controller: formulating an optimal control objective and uncertainty set that leads to robust stability, or explicitly applying robust contraction constraints to guarantee stability. Min-max MPC [10] and tube-based MPC [11] follow these two options, respectively.

As one of the earliest robust MPC schemes, min-max MPC seeks to optimize the cost for the worst-case uncertainty in the plant while satisfying constraints for all uncertainty cases. This approach is conservative due to the lack of feedback mechanism such that future-available new information cannot be exploited; hence the system is likely to reach infeasibility. In order to resolve this issue, feedback min-max MPC [12] is introduced to take into account future disturbances and consequently avoid the infeasibility problem. But the computational cost for feedback min-max can be prohibitively high for long horizons. Alternatively, tube-based MPC incorporates an ancillary controller, which forces the evolution of the uncertain system to stay within a tube centered around the nominal trajectory [11]. However, compared with its linear version, tube-based nonlinear ancillary control law is extremely difficult to compute offline. A modified tube-based robust NMPC [13] is proposed with an online control sequence with guaranteed system stability and constraint satisfaction, but does not ensure optimal performance under uncertainty.

Stochastic programming-based methods provide an alternative approach to solve robust MPC problems. A scenario-based stochastic MPC is developed for linear systems with multiplicative disturbances, where the A and B matrices are functions of disturbances [14]. An air separation problem is studied under plant-model mismatch by a multi-scenario formulation with some robust stability analysis [15]. However, this approach does not take advantage of the degrees of freedom of the control variables; hence it's relatively conservative. A multistage NMPC scheme is proposed to improve on that, where a major assumption is that a discretized scenario tree represents the uncertainty evolution throughout the entire horizon [16]. This framework has low conservatism because future control actions act as recourse variables to counteract the effects of uncertainty. More recently, an inexpensive prediction of worst-case parameter realizations has been developed along with fast, heuristic multi-stage methods for worst-case NMPC [17]. In addition, the two-stage stochastic MPC framework has been applied to a battery management study to illustrate its superior performance over deterministic MPC [18].

For conventional multi-stage NMPC, the scenario tree structure inevitably increases the problem size and raises the challenge of solving the control problem online, especially when the dimensionality of the tree becomes large. The resultant delayed availability of control actions might contribute to system instability or suboptimal control performance [19]. Efficient solution of multistage stochastic programs can be enabled through parallelizable decomposition methods [20], where primal decompositions perform iterations on stages and dual decompositions iterate on scenarios. Recent decomposition algorithms for multistage NMPC include decompositions for each scenario, with QPs solved in an inner layer, and with a non-smooth Newton iteration in an outer loop to satisfy non-anticipativity constraints [21]. However, dual decomposition relaxes the non-anticipativity constraints, which may result in different control inputs for different subproblems. A primal decomposition method is proposed that ensures the feasibility of non-anticipativity constraints [22, 23]. Alternately, a parallelizable primal-dual interior-point method is presented that exploits the structure of the problem, but is limited to two-stage robust MPC with linear dynamics [24]. More recently, the optimal cost-to-go functions of different scenarios are approximated by neural networks and a semi-batch reactor example has been studied [25]. Though online implementation is faster than the original multistage method, the neural network-based approach requires tuning of the reduced length of robust horizon, which is cumbersome and may introduce large constraint violations if not done carefully. Finally, an online scenario generation method is introduced to approximate multistage NMPC with far fewer scenarios, but only by optimizing a worst case cost function [26].

1.2 Research problem statement

The goal of this thesis is to provide systematic and real-time enabled approaches to obtain robust and stable system behaviors under uncertainty using NMPC. We seek to extend the current state-of-the-art robust NMPC approaches with meaningful uncertainty descriptions, and with controllers of verified stability properties. And we propose methods that significantly reduce computational efforts while keeping the control performance on a par with the current best available approaches, which allows real-time implementation. Through the help of control and optimization theory, certain problem structures associated with multistage NMPC have been exploited to accelerate the algorithm remarkably. To make robust NMPC more accessible to researchers and developers, we also make our codes and package open-source so it becomes possible to use robust NMPC formulations without taking a long detour.

1.3 Thesis outline

This thesis is organized as follows:

Chapter 2 introduces the basic terminology in order to understand a control system.

Lyapunov stability theory is presented as the foundation of stability analysis throughout this thesis. Also the history and evolution of MPC has also been covered.

Chapter 3 presents an overview of solution strategies for solving constrained dynamic optimization problems, which are used for optimal control applications. After transcribing a dynamic optimization problem to an nonlinear programming problem, a suitable solver can then be used. This chapter also covers the basic concepts in NLP and NLP sensitivity, that is further extended in the rest of the thesis for perturbed solutions.

Chapter 4 proposes a parallelizable advanced-step multistage NMPC, which provides a non-conservative robust control solution that explicitly addresses two types of uncertainty: model parameters and unmeasured noise. Type 1 uncertainty is attended to by incorporating scenario trees and Type 2 uncertainty is addressed by applying NLP sensitivity. A scenario tree is constructed to represent the evolution of states with respect to different Type 1 uncertainty realizations. Non-anticipativity constraints (NAC) are enforced among branches of the tree that share the same root state. On top of that, the concepts of advanced-step and multistage scenario trees are integrated to achieve a better online performance.

Chapter 5 focuses on the stability properties of NMPCs under different uncertainty exposure. The nominal stability is proved for standard NMPC (ideal NMPC) and advanced-step NMPC when the model is perfect and there is no disturbance. However, when uncertainty is present, one resorts to robust stability such as input-to-state stability (ISS), or input-to-state practical stability (ISpS). This chapter also goes into detail about proving recursive feasibility and robust stability for ideal and advanced-step multistage NMPC.

Chapter 6 develops a computationally efficient approach called sensitivity-assisted multistage NMPC (samNMPC) that emulates the multistage NMPC with scenario generation. The exact multistage NMPC formulation generally needs to take on a large NLP due to its model construction, especially when the number of scenarios grows exponentially with respect to the number of uncertain parameters and the number of robust horizons. SamNMPC considers an approximate formulation to the conventional multistage NMPC such that the number of scenarios in the optimization problem is only proportional to the number of active inequality constraints, which is often much less expensive in comparison. At the same time, it also considers the weighted sum of stage costs for *all* scenarios by performing a NLP sensitivity calculation. SamNMPC has been implemented in CasADi and has demonstrated potential to promote the use of robust NMPC for many applications.

Chapter 7 concludes the thesis with its main contributions and also gives an outlook on directions for future work.

Chapter 2

Nonlinear model predictive control and stability concepts

2.1 Introduction

This chapter introduces the fundamental concepts of stability analysis and basic NMPC concepts. It covers the state-space representation of the controlled system as well as generic system assumptions, Lyapunov stability theory and standard setpoint tracking NMPC. These existing frameworks facilitate future discussions towards the design of robust NMPC controllers in later chapters.

2.2 Notations and definitions

Consider the dynamics of the plant by a discrete-time system:

$$x_{k+1} = f(x_k, u_k, d_k) \tag{2.1}$$

where $x_k \in \mathbb{X} \subset \mathbb{R}^{n_x}, u_k \in \mathbb{U} \subset \mathbb{R}^{n_u}$ are the system states and controls, and $d_k \in \mathbb{D} \subset \mathbb{R}^{n_d}$ is the vector of disturbances defined at time step t_k where $k \geq 0$ is the time index. Note that the set $\mathbb{X}$ is called a region of attraction in this chapter.

We use $\mathbb{R}, \mathbb{R}_+, \mathbb{Z}$ and $\mathbb{Z}_+$ to represent the real and non-negative real, and the integer and non-negative integer numbers, respectively. Let $|\cdot|$ and $\|\cdot\|$ denote the Euclidean vector norm and the corresponding induced matrix norm. The sequence of disturbance is denoted by $\mathbf{d} \triangleq [d_0, d_1, d_2, \dots]$ and the truncated sequence at time $k \in \mathbb{Z}_+$ is denoted by $\mathbf{d}_k = [d_0, d_1, \dots, d_{k-1}, 0, \dots]$. And for a given sequence, $\|\mathbf{d}\| \triangleq sup_{k\in\mathbb{Z}_+} \{|d_k|\}$.

Definition 1. *[27] A function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is continuous in $\mathbb{R}^n$ if for any x_1, x_2 and all $\epsilon > 0$, there exists a $\delta > 0$ such that $\|x_1 - x_2\| < \delta \Rightarrow \|f(x_1) - f(x_2)\| < \epsilon$.*

A continuous function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous in $\mathbb{R}^n$ if for any x_1, x_2 there exists a finite $L > 0$ such that $\|f(x_1) - f(x_2)\| < L\|x_1 - x_2\|$.

Definition 2. *[28] A continuous function $\alpha(\cdot) : \mathbb{R}_+ \to \mathbb{R}_+$ is a $\mathcal{K}$ function if $\alpha(0) = 0, \alpha(s) > 0, \forall s > 0$ and it is strictly increasing.*

A continuous function $\alpha(\cdot) : \mathbb{R}_+ \to \mathbb{R}_+$ is a $\mathcal{K}_\infty$ function if it is a $\mathcal{K}$ function and $\alpha(s) \to \infty$ as $s \to \infty$.

A continuous function $\beta(\cdot, \cdot) : \mathbb{R}_+ \times \mathbb{Z}_+ \to \mathbb{R}_+$ is a $\mathcal{KL}$ function if $\beta(s, k)$ is a $\mathcal{K}$ function in s for any $k \geq 0$ and for each $s \geq 0, \beta(s, \cdot)$ is nonincreasing and $\beta(s, k) \to 0$ as $k \to \infty$.

Definition 3. *(Stable equilibrium point) The point $x = 0$ is called a stable equilibrium point of* (2.1) *if for all $k_0 \in \mathbb{Z}_+$ and $\epsilon_1 > 0$, there exists $\epsilon_2 > 0$ such that $|x_{k_0}| < \epsilon_2 \;\Rightarrow\; |x_k| < \epsilon_1$ for all $k \geq k_0$.*

We then make the following assumptions for the system (2.1):

Assumption 4. *(System assumptions)*

- *The system function $f(\cdot, \cdot, \cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ is continuous, with an equilibrium point at the origin, i.e. $f(0, 0, 0) = 0$.*
- *The set $\mathbb{X}$ is closed and bounded, and contains the origin in its interior. And the set $\mathbb{X}$ is positive invariant and the set $\mathbb{A} \subseteq \mathbb{X}$ is closed and positive invariant for system* (2.1).
- *The set $\mathbb{U}$ is closed and bounded, and contains the origin in its interior.*
- *The set $\mathbb{D}$ is a known compact set containing the origin.*
- *The states x_k of the system can be measured at each sampling time.*

2.3 Lyapunov stability theory

To understand Lyapunov stability theory, a useful analogy is energy in a passive mechanical system (e.g. a pendulum). The total energy for a pendulum system is the sum of the kinetic and potential energies, which monotonically decreases as time proceeds because of friction. Eventually, the system reaches equilibrium when the total energy decays to zero and the pendulum stops at the bottom. Following a similar trajectory, if one can find a real-valued function (i.e. Lyapunov function) that is positive and decreasing except at the origin, then it can be proved that the state converges to the origin. Lyapunov stability theory establishes the foundation of stability properties of NMPC.

For the clarity of the discussion, we refer to the case in the absence of uncertainty as *nominal stability*, and the case that considers the effect of uncertainty as *robust stability*.

2.3.1 Nominal stability

First we have a controlled system (2.1) in absence of uncertainties, i.e. $d_k = 0$ for $k \in \mathbb{Z}_+$. The system now becomes

$$x_{k+1} = f(x_k, u_k, 0) = \tilde{f}(x_k, u_k) \tag{2.2}$$

Definition 5. *(Control invariant set) A set $\mathbb{X} \subseteq \mathbb{R}^{n_x}$ is control invariant for $x^+ = f(x, u), u \in \mathbb{U}$ if there exists a $u \in \mathbb{U}$ such that $f(x, u) \in \mathbb{X}$ for all $x \in \mathbb{X}$.*

Definition 6. *(Asymptotic stability) Suppose Assumption 4 holds for system* (2.2)*, the set $\mathbb{A}$ is asymptotically stable in $\mathbb{X}$ for system* (2.2) *if there exists a $\mathcal{KL}$ function $\beta(\cdot)$ such that $\forall x_0 \in \mathbb{X}$,*

$$|x_k| \leq \beta(|x_0|, k), \ \ \forall k \geq 0 \tag{2.3}$$

Definition 7. *(Lyapunov function) Suppose Assumption 4 holds for system* (2.2)*, a function $V(\cdot)$: $\mathbb{X} \to \mathbb{R}_+$ is called a Lyapunov function in $\mathbb{X}$ for system (2.2) if there exist a feedback control law*

$h(x)$ and $\mathcal{K}_\infty$ functions α_1, α_2 and α_3 such that, $\forall x \in \mathbb{X}$

$$V(x) \geq \alpha_1(|x|) \tag{2.4a}$$

$$V(x) \leq \alpha_2(|x|) \tag{2.4b}$$

$$\Delta V(x) = V(f(x, h(x))) - V(x) \leq -\alpha_3(|x|) \tag{2.4c}$$

Theorem 8. *Under Assumption 4, if system* (2.2) *admits a Lyapunov function, then the system* (2.1) *is asymptotically stable on $\mathbb{X}$.*

Details of Theorem 8 and its proof can be found in Appendix B in [28].

2.3.2 Robust stability

We now extend the concept of Lyapunov functions to system (2.1) with disturbances. The robust stability property concerns with the boundedness of the state in terms of a bounded disturbance sequence. To analyze the robustness of a controlled system, we introduce the input-to-state stability (ISS) property. Inspired by input-to-state stability for continuous-time nonlinear systems [29, 30], ISS has been extended to discrete-time nonlinear systems in [31], and more recently has been used as a unifying framework to analyze stability of NMPC in [32].

Definition 9. *(Robust positive invariant (RPI) set) [28] Consider that Assumption 4 holds for system* (2.1)*, a set $\mathbb{A} \subseteq \mathbb{X}$ is a robust positive invariant (RPI) set for system* (2.1) *if for all $x \in \mathbb{A}, d \in \mathbb{D}$, there exists a $u \in \mathbb{U}$ such that $f(x, u, d) \in \mathbb{A}$.*

Definition 10. *(Input-to-state stability (ISS)). Under Assumption 4, the system* (2.1) *is input-to-state stable (ISS) in $\mathbb{X}$ if there exists a $\mathcal{KL}$ function $\beta(\cdot)$ and $\mathcal{K}$ function $\sigma(\cdot)$ such that, $\forall x \in \mathbb{X}, d \in \mathbb{D}$,*

$$|x_k| \leq \beta(|x_0|, k) + \sigma(\|\mathbf{d}\|) \;\; \forall k \geq 0 \tag{2.5}$$

This definition of ISS shows that the norm of the state is asymptotically bounded by $\sigma(\|\mathbf{d}\|)$, which also implies that if the disturbance sequence is zero, then the origin is asymptotic stable. Similar to asymptotic stability, we seek a form of Lyapunov function that ensures ISS.

Definition 11. *(ISS Lyapunov function) A function $V : \mathbb{X} \to \mathbb{R}_+$ is an ISS-Lyapunov function in $\mathbb{X}$ for system* (2.1) *if there exist $\mathcal{K}_\infty$ functions $\alpha_1(\cdot), \alpha_2(\cdot), \alpha_3(\cdot)$ and $\mathcal{K}$ function $\sigma(\cdot)$ such that, $\forall x \in \mathbb{X}, d \in \mathbb{D}$*

$$\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|) \tag{2.6a}$$

$$V(f(x,u,d)) - V(x) \leq -\alpha_3(|x|) + \sigma(|d|) \tag{2.6b}$$

Theorem 12. *(ISS-Lyapunov function implies ISS). Under Assumption 4, if the system* (2.1) *admits a continuous ISS-Lyapunov function in $\mathbb{A}$, then the system* (2.1) *is ISS in $\mathbb{A}$.*

Details of proof to Theorem 12 can be found in [31].

We then employ a more general definition of input-to-state stability as the input-to-state practical stability (ISpS). ISpS was first introduced in [33], and it then has provided an appropriate framework to analyze the stability of uncertain nonlinear discrete-time system in [34].The term "practical" means a neighborhood of the origin, instead of the origin itself, can be ensured.

Definition 13. *(Input-to-state practical stability (ISpS)). [32] Under Assumption 4, the system* (2.1) *is input-to-state practical stable (ISpS) in $\mathbb{X}$ if there exists a $\mathcal{KL}$ function $\beta(\cdot)$, $\mathcal{K}$ function $\sigma(\cdot)$ and a constant $c \geq 0$ such that, $\forall x \in \mathbb{X}, d \in \mathbb{D}$,*

$$|x_k| \leq \beta(|x_0|, k) + \sigma(||\mathbf{d}||) + c \ \ \forall k \geq 0 \tag{2.7}$$

The constant c is a non-vanishing term that reflects the system (2.1) may not evolve to the origin, but only to a compact neighborhood of the origin. Note that if $c = 0$ ISpS reverts to ISS in $\mathbb{X}$. Again, we seek a sufficient condition to ensure ISpS by forming a Lyapunov-like function.

Definition 14. *(ISpS Lyapunov function) A function $V : \mathbb{X} \to \mathbb{R}_+$ is an ISpS-Lyapunov function in $\mathbb{X}$ for system* (2.1) *if there exist $\mathcal{K}_\infty$ functions $\alpha_1(\cdot), \alpha_2(\cdot), \alpha_3(\cdot)$, $\mathcal{K}$ function $\sigma(\cdot)$ and a couple of constants $c_1, c_2 \geq 0$ such that, $\forall x \in \mathbb{X}, d \in \mathbb{D}$*

$$\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|) + c_1 \tag{2.8a}$$

$$V(f(x,u,d)) - V(x) \leq -\alpha_3(|x|) + \sigma(|d|) + c_2 \tag{2.8b}$$

Theorem 15. *(ISpS-Lyapunov function implies ISpS) Under Assumption 4, if the system* (2.1) *admits a continuous ISpS-Lyapunov function in $\mathbb{A}$ and $f(\cdot)$ is continuous, then the system* (2.1) *is ISpS in $\mathbb{A}$.*

Details of the proof to Theorem 15 can be found in [34].

2.4 Basic NMPC

Rooted in optimal control, Model Predictive Control (MPC) is a form of advanced modern control that optimizes the control performance of a dynamic system, of which a model is used to forecast the future system behavior. An implicit feedback control is then generated by MPC through solving a mathematical programming problem. A typical application to use a feedback controller is to drive the system output trajectory to a predefined setpoint, which is commonly referred to as setpoint tracking and is the main focus of this thesis.

MPC obtains the control *action* by solving a finite horizon open-loop optimal control problem *online* at each time step. It differs from the classical control in which an *off-line* feedback control *law* is precomputed (which provides the optimal control for *all* states). The solution of MPC consists of an optimal control sequence, of which the first step is implemented to the plant. MPC is particularly useful when an off-line control law is difficult or impossible to obtain [6]. A great advantage of MPC is that the open-loop optimization problem can be solved fast enough to allow the implementation of the feedback control, which is especially true with the development of modern computing infrastructure.

We study a plant with a nonlinear behavior in the form of continuous-time nonlinear differential equations,

$$\frac{dx}{dt} = F(x, u) \tag{2.9}$$

In order to achieve the best closed-loop control performance, we need to solve the following infinite horizon, constrained optimal control problem. The objective is to minimize a cost-to-go function which has the form

$$V_\infty(x, u(\cdot)) = \int_0^\infty \Phi(x(t), u(t))dt \tag{2.10}$$

where $x(t)$ and $u(t)$ satisfy (2.9). We now have the optimal control problem $\mathcal{P}_\infty(x)$:

$$\begin{aligned} V_\infty^0(x) = \min_{u(\cdot)} \quad & \int_0^\infty \Phi(x(t), u(t)) && \text{(2.11a)} \\ \text{s.t.} \quad & \dot{x} = F(x, u) && \text{(2.11b)} \\ & x(0) = x_0 && \text{(2.11c)} \\ & x(t) \in \mathbb{X},\ u(t) \in \mathbb{U} \quad \forall t \in \mathbb{R}_+ && \text{(2.11d)} \end{aligned}$$

where the optimal cost-to-go function is denoted as $V_\infty^0(x)$. Assume the stage cost function $\varphi(\cdot)$ is positive definite, the goal of $\mathcal{P}_\infty(x)$ is to converge the system states to the origin.

The optimal control sequence $u^0_\infty(\cdot; x)$ for problem $\mathcal{P}_\infty(x)$ with the given initial state measurement x is very difficult to compute. First, solving a infinite-dimensional function in a mathematical programming setting is challenging; second, the semi-infinite time interval $[0, \infty]$ may impose some numerical complexities.

In order to have a tractable $\mathcal{P}_\infty(x)$, we use a discrete-time difference equation instead of its continuous-time counterpart for a practical computer implementation. The discrete-time infinite horizon MPC problem becomes $\mathcal{P}^k_\infty(x)$:

$$V^k_\infty(x) = \min_{z_l, v_l} \quad \sum_{l=0}^{\infty} \varphi(z_l, v_l) \tag{2.12a}$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l) \;\; \forall\, l = 0 \ldots \infty \tag{2.12b}$$

$$z_0 = x_k \tag{2.12c}$$

$$z_l \in \mathbb{X}, v_l \in \mathbb{U} \;\; \forall\, l = 0 \ldots \infty \tag{2.12d}$$

where $f(z_l, v_l) = \int_{t_l}^{t_{l+1}} F(z(t), v(t))dt$ and v_l is constant over $t \in [t_l, t_{l+1})$. Details of derivation will be discussed in Chapter 3.

At time k, the beginning state of the plant is x_k that acts as the initial condition for (2.12), which computes an optimal trajectory over the entire infinite horizon. At time $k+1$, since no new horizon enters the system, the same trajectory can be used (excluding the first step) and still remains optimal. This is often referred to as Bellman's *principle of optimality*. See the bottom diagram of Fig. 2.1 for illustration. As a result, recursive feasibility holds for $\mathcal{P}^k_\infty(x)$ because $\mathcal{P}^k_\infty(x), k \in \mathbb{Z}_+$ has feasible solutions as long as $\mathcal{P}^0_\infty(x)$ can be solved. Under the assumption of perfect model and no uncertainties, the cost-to-go function $V^k_\infty(x)$ decreases as k increases, which satisfies Definition 7 as a Lyapunov function candidate to guarantee asymptotic stability [35, 36].

Even though more recent work [37] shows that it is possible to solve infinite horizon optimal control problem online for linear systems with certain limitations, it still remains

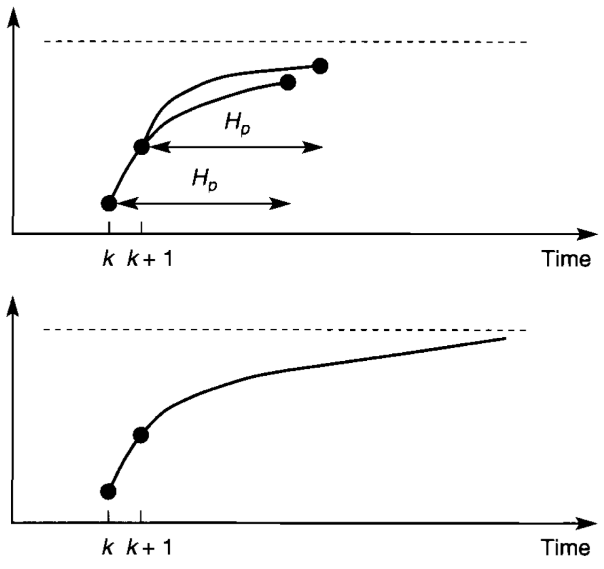


Figure 2.1: Adapted from [4]. Finite horizon and infinite horizon MPC optimal trajectory for a constant setpoint (perfect model without uncertainties).

an open issue for nonlinear systems. Actually it turns out that the infinite horizon problem can be approximated by a finite horizon problem [38]. For an *infinite horizon* problem with no uncertainties or model errors, the open-loop and closed-loop trajectories are the same, hence it only requires solving once. For *finite horizon* MPC, on the other hand, it needs to execute in every time instant, yet finite horizon is still much more viable [39].

At time k, one obtains the optimal state trajectory for a finite horizon H_p. For a perfect model without disturbances, the predicted state at $k+1$ will be realized in the plant precisely. However, the optimal trajectory from $k+1$ to $k+1+H_p$ may not follow the previously optimal trajectory from k to $k+H_p$ because of the new interval between step $k+H_p$ and $k+1+H_p$ that is missing at time k for the finite horizon MPC problem. See the top diagram in Fig. 2.1 for a finite horizon MPC.

For a finite horizon control problem, in order to guarantee the closed-loop stability, one can add to the objective function a terminal cost $\phi(\cdot)$ which is a global control Lyapunov function [40]. In the case where a global control Lyapunov function is unavailable, one can find a local control Lyapunov function defined in a neighborhood of the origin, which is also referred to as the terminal region $\mathbb{X}_f$ [41]. In this case, one can write the finite horizon problem $\mathcal{P}_N(x)$ with a finite horizon N with terminal cost and region as,

$$\min_{z_l, v_l} \quad \phi(z_N) + \sum_{l=0}^{N-1} \varphi(z_l, v_l) \tag{2.13a}$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l) \;\; \forall\, l = 0 \dots N-1 \tag{2.13b}$$

$$z_0 = x_k \tag{2.13c}$$

$$z_l \in \mathbb{X}, v_l \in \mathbb{U} \;\; \forall\, l = 0 \dots N-1 \tag{2.13d}$$

$$z_N \in \mathbb{X}_f \tag{2.13e}$$

In the discrete-time formulation, we denote $z_l \in \mathbb{R}^{n_x}$ and $v_l \in \mathbb{R}^{n_u}$ as the predicted state and control variables, respectively. $l = 0, 1, ..., N-1$ is the horizon index within the finite prediction horizon N. $f(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the discretized dynamic function. The tracking stage cost is represented by $\varphi(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_+$, which characterizes the deviation from the setpoint. The terminal cost is denoted by $\phi(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}_+$. For each time step k, (2.13) is solved with the optimal control sequence as $\{v_0^*, v_1^*, ..., v_{N-1}^*\}$, in which the first step is injected to the plant, i.e. $u_k = v_0^*$. Then the plant evolves with $k = k+1$ and the procedure repeats. This is also commonly referred to as *moving horizon* or *receding horizon*.

There are alternative terminal conditions for stabilizing finite horizon MPC [41]. One case would be to use *dual-mode* predictive control, in which there is only terminal region but no terminal cost. The receding horizon controller is applied until the system reaches the terminal region, inside of which a linear stablizing controller is employed [42]. In addition, one can assume that the finite horizon N is long enough by analyzing the turnpike property [43], which is difficult for nonlinear systems. Alternatively, one could argue that imposing terminal constraints and costs are unnecessary [44]. More recently, one could adaptively update the finite horizon length to approximate the infinite horizon NMPC problem [45]. In practice, one generally assumes N is long enough and drop the terminal conditions in for implementation purposes.

Chapter 3

Computational strategies

As mentioned in Chapter 2, MPC obtains the control action by solving a mathematical programming problem online. This chapter starts by describing the generic dynamic optimization problem class, which includes the optimal control problem of interest. For nonlinear optimal control problems, one relies on numerical solutions since closed-form solutions are almost impossible to obtain. We introduce solution strategies and numerical methods as toolboxes to facilitate that. In particular, the direct method transcribes the dynamic optimization problem to a nonlinear programming (NLP) problem, which we can then resort to a NLP solver such as IPOPT. A brief introduction of interior point algorithm and NLP sensitivity are also reviewed.

3.1 Dynamic optimization

Many applications in science and engineering deal with nonlinear differential-algebraic (DAE) models, which includes aerospace systems, chemical processes, economics and finance, robotics systems and many others. Consider a generic form of DAE-constrained dynamic optimization problem:

$$\min \quad \int_{t_0}^{t_f} \Phi(x, y, u, p) dt \tag{3.1a}$$

$$\text{s.t.} \quad \frac{dx}{dt} = F(x, y, u, p), \quad x(t_0) = x_0 \tag{3.1b}$$

$$c(x, y, u, p) = 0 \tag{3.1c}$$

$$g(x, y, u, p) \leq 0 \tag{3.1d}$$

where x, y, u represent differential variables, algebraic variables and control inputs, respectively, and p is the parameter vector; $F(\cdot)$ and $c(\cdot)$ are differential equations and algebraic equations, respectively. $g(\cdot)$ denotes the extra constraints which can be variable bounds. The goal of dynamic optimization is to minimize some performance metrics denoted by $\Phi(\cdot)$ for a specified interval $[t_0, t_f]$.

A model in dynamic optimization generally comes from first-principles, though more recently machine learning based (i.e. data-driven) models also come into play. For a chemical process, the differential equations are for the system dynamic behavior (e.g. mass and energy balance), and the algebraic equations ensure constitutive relations such as physics and thermodynamics. A common application of dynamic optimization is optimal control [46].

While formulating a dynamic optimization problem as (3.1) is fairly straightforward, solving one may be difficult. A special case is linear optimal control, where one has a linear differential equation and quadratic cost function, commonly known as Linear-Quadratic-Regulator (LQR). LQR computes an analytical solution to the optimal control gain by solving Riccati differential equation, and stability of LQR can be proved in [47].

As for nonlinear optimal control problems, one generally does not have analytic solutions as in linear quadratic control, hence numerical solutions are sought after. There are two realms of numerical methods that solve dynamic optimization problems: *indirect* method and *direct* method.

The *indirect* method, or *variational approach* derives from Pontryagin's maximum principle [48], where the first order necessary condition for optimality is solved. For problem with equality constraints only, the optimality condition can be formulated as a set of differential-algebraic equations. This DAE system has boundary conditions at both t_0 and t_f. The resulting *two-point boundary value problem* can be solved with different approaches, such as single shooting, invariant embedding, multiple shooting, or discretization method (e.g. collocation on finite elements). On the other hand, handling path inequalities is tricky in indirect methods as reasonable initial guesses for state and adjoint variables are often difficult to find. For a detailed presentation of this material, readers are referred to [27, 49, 50].

Motivated by the inconvenience of *indirect* method, the *direct* method converts a dynamic optimization problem to a nonlinear programming (NLP) problem. Since integrals or differential equations cannot be directly handled by NLP solvers, the continuous dynamics need to be discretized and reformulated as algebraic equations. Direct methods are generally separated into two categories: sequential and simultaneous methods.

3.1.1 Single shooting and multiple shooting

The *single shooting* method, or *sequential* method, only discretizes the control variables and the corresponding solution techniques are also called control variable parameterization. The control inputs after discretization are represented by piecewise constants or piecewise polynomials. By selecting a set of controls, the process model can be integrated with a DAE solver with a given initial condition (i.e. "shoot" as an initial value problem). The DAE models in the optimization problem are essentially replaced by its gradient information with respect to the controls, which are provided either with direct or adjoint sensitivity equations. Then the error in the boundary condition is evaluated, and a NLP solver is used to obtain a new guess of controls. This method keeps iterating between solving a NLP for a control trajectory and solving a DAE with that new guess, until the boundary condition is met.

Sequential methods are straightforward to construct. However, it requires repeated numerical integration with the DAE solver, which is time-consuming for large-scale systems. In addition, it has also been reported to have difficulties handling stiff or unstable systems [51, 52].

In order to reduce the numerical unstability of single shooting, *multiple shooting* approach is introduced where the entire domain is partitioned into smaller intervals such as $[t_0, t_1], [t_1, t_2], ..., [t_{N-1}, t_f]$ (i.e. ”shoot” not that far), and DAE model is integrated in each interval [53]. Unlike the single shooting where only control variables are discretized, multiple shooting discretizes both control variables and the initial condition for state variables in each segment, and gradient information are obtained for both types of variables. In addition, a continuity constraint is needed between two consecutive segments to ensure states are continuous across intervals.

Because of the additional variables and constraints for each shooting interval, multiple shooting approach inevitably faces an increase in the problem size. Fortunately, as a direct outcome of applying this formulation, the Jacobian matrix for Newton iteration is actuallly sparse. It also allows the inequality constraints on state and control variables to be imposed directly at each segment. Moreover, multiple shooting approach reportedly provides significant improvement on stability compared to single shooting [54].

3.1.2 Direct transcription

Unlike the shooting methods in the preceding subsection where a set of differential equations are ”propagated”, *direct transcription* method, also known as *simultaneous approach*, removes the iteration between a DAE integrator and a NLP solver but rather directly couples the dynamic optimization problem with the NLP solution (hence ”simultaneous”). It fully discretizes both state and control variables of the DAE system, where a large-scale NLP problem is formed and its solution is represented by piecewise polynomials.

The simultaneous approach has many advantages. It can solve problems with path constraints and instabilities, and it also allows constraints on state and control variables explicitly. Additionally, it avoids integrating DAE system as intermediate steps, which could be computationally intensive. On the other hand, it transcribes the dynamic optimization problem to a large-scale NLP problem with many variables and degrees of freedom, where a special NLP solver is called for (which we will explain in detail in the following section). In this thesis we focus on direct transcription method, where *transcription* and *collocation* are used interchangeably.

The direct transcription method follows a full discretization methodology, where a continuous time horizon is represented by orthogonal collocation on finite elements. The control variable is usually parametrized as piecewise constant or piecewise linear within finite elements. For the differential state, suppose that the state variable is represented by a polynomial of order $K+1$ (i.e. highest possible degree is K), the polynomial can be written as a power series as below, or B-splines [51, 55]:

$$x(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \ldots + \alpha_K t^K \tag{3.2}$$

We can then choose $K+1$ interpolation points in element i and approximate the differential variables $x(t)$ using Lagrange interpolation polynomials $l_j(\cdot)$ for the reason of exactness at collocation points. For $t \in [t_{i-1}, t_i],\ \tau \in [0,1]$,

$$t = t_{i-1} + h_i \tau, \tag{3.3a}$$

$$x(t) = \sum_{j=0}^{K} l_j(\tau) x_{ij} \tag{3.3b}$$

where $l_j(\tau) = \Pi_{k=0,\neq j}^{K} \frac{\tau - \tau_k}{\tau_j - \tau_k}$ with $\tau_0 = 0, \tau_j < \tau_{j+1}$. In addition, $i = 1, ..., N_{FE}$ is the index of finite elements and $j = 0, ..., K$ is the index of the collocation points. By definition, τ_j is the "intermediate" point that satisfies $x(t_{ij}) \equiv x_{ij}$, where $t_{ij} = t_{i-1} + \tau_j h_i$.

Now the ODE equation (3.1b) can be substituted with (3.3b) where t is a collocation point $t_{ik} = t_{i-1} + \tau_k h_i$. Since $\frac{dx(t_{ik})}{dt} = f(x_{ik}, t_{ik})$, for $k = 1, ..., K$,

$$\sum_{j=0}^{K} x_{ij} \frac{dl_j(\tau_k)}{d\tau} = h_i f(x_{ik}, t_{ik}) \tag{3.4}$$

Similarly, the algebraic variables can be treated as Lagrange polynomials of order $K+1$,

$$y(t) = \sum_{j=0}^{K} l_j(\tau) y_{ij} \tag{3.5}$$

We would also need to add continuity constraints across element boundaries. As a result, the original dynamic optimization model (3.1) is now translated to the following NLP formulation and can be solved using off-the-shelf solvers.

$$\begin{aligned}
\min \quad & \varphi(x_{ik}, y_{ik}, u_{ik}, p_{ik}) && (3.6a)\\
\text{s.t.} \quad & \sum_{j=0}^{K} x_{ij} \frac{dl_j(\tau_k)}{d\tau} = h_i f(x_{ik}, y_{ik}, u_{ik}), \quad x_{0,0} = x_0, && (3.6b)\\
& c(x_{ik}, y_{ik}, u_{ik}, p_{ik}) = 0 && (3.6c)\\
& g(x_{ik}, y_{ik}, u_{ik}, p_{ik}) \le 0 && (3.6d)\\
& x_{i+1,0} = x_{iK} && (3.6e)\\
& i = 0, ..., N_{FE} - 1, \;\; k = 1, ..., K && (3.6f)
\end{aligned}$$

3.2 Nonlinear optimization

After transcribing a dynamic optimization problem using simultaneous methods, we now seek an efficient NLP solver. Among nonlinear optimization algorithms, the interior point algorithm is especially preferrable for large-scale optimal control problems [56]. For the purpose of this thesis, an interior point algorithm implementation IPOPT [57] is applied to solve nonlinear optimization problems. This section will first introduce the theoretical foundation of NLP, then briefly cover the interior point algorithm, and finally discuss NLP sensitivity, which will be used extensively throughout this thesis.

3.2.1 Nonlinear programming theory

Consider a generic NLP formulation as follows,

$$\min_{\mathbf{x}} \; F(\mathbf{x}; p) \tag{3.7a}$$

$$\text{s.t.} \; c(\mathbf{x}, p) = 0, \tag{3.7b}$$

$$g(\mathbf{x}, p) \leq 0 \tag{3.7c}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the decision variable and $p \in \mathbb{R}^p$ is the parameter. The objective function is $F(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}$, the equality constraint is $c(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{m_e}$, and the inequality constraint is $g(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{m_i}$. $\mathcal{I} = \{1, ..., m_e\}$ and $\mathcal{J} = \{1, ..., m_i\}$ are the index sets for equality constraints and inequality constraints, respectively.

In order to find the optimal solution of (3.7), let's first define the basic concepts in nonlinear programming. For convenience of notation, we denote the feasible region $\mathcal{F} = \{\mathbf{x} | c(\mathbf{x}, p) = 0, \; g(\mathbf{x}, p) \leq 0\}$.

Definition 16. *(Local solution, [27, 58])*

- $\mathbf{x}^*$ *is a local solution of* (3.7) *if* $\mathbf{x}^* \in \mathcal{F}$ *and there exists a neighborhood* $\mathcal{N}$ *of* $\mathbf{x}^*$ *such that* $F(\mathbf{x}) \geq F(\mathbf{x}^*)$ *for all* $\mathbf{x} \in \mathcal{N}(\mathbf{x}^*) \cap \mathcal{F}$
- $\mathbf{x}^*$ *is a strict local solution of* (3.7) *if* $\mathbf{x}^* \in \mathcal{F}$ *and there exists a neighborhood* $\mathcal{N}$ *of* $\mathbf{x}^*$ *such that* $F(\mathbf{x}) > F(\mathbf{x}^*)$ *for all* $\mathbf{x} \in \mathcal{N}(\mathbf{x}^*) \cap \mathcal{F}$ *and* $\mathbf{x} \neq \mathbf{x}^*$

Definition 17. *(Active set, [58]) For inequality constraint, the active set $\mathcal{J}(\mathbf{x})$ for any feasible $\mathbf{x}$ is $\mathcal{J}(\mathbf{x}) = \{j \mid g_j(\mathbf{x}) = 0\}$. In contrast, the inequality constraint $j \in \mathcal{J}$ is inactive if strict inequality is satisfied $g_j(\mathbf{x}) < 0$.*

Definition 18. *(LICQ, [58]) Given a point $\mathbf{x}$ and its active set $\mathcal{J}(\mathbf{x})$, the linear independence constraint qualification (LICQ) holds when the constraint gradients*

$$\begin{aligned} &\nabla c_i(\mathbf{x}, p), \quad i \in \mathcal{I}, \\ &\nabla g_j(\mathbf{x}, p), \quad j \in \mathcal{J}(\mathbf{x}). \end{aligned} \tag{3.8}$$

are linearly independent.

Definition 19. *(Lagrange function) The Lagrange function of* (3.7) *can be written as:*

$$\mathcal{L}(\mathbf{x}, \nu, \eta, p) = F(\mathbf{x}, p) + c(\mathbf{x}, p)^T \nu + g(\mathbf{x}, p)^T \eta \tag{3.9}$$

where $\nu \in \mathbb{R}^{m_e}$ and $\eta \in \mathbb{R}^{m_i}$ are multipliers of equality and inequality constraints, respectively.

Definition 20. *(KKT conditions, [27, 58]) Suppose $\mathbf{x}^*$ is a local minimum of* (3.7)*, and some constraint qualification holds at $\mathbf{x}^*$, then there exist Lagrange multipliers ν and η such that,*

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \nu, \eta, p) = 0 \tag{3.10a}$$

$$c(\mathbf{x}^*, p) = 0 \tag{3.10b}$$

$$g(\mathbf{x}^*, p) \leq 0 \tag{3.10c}$$

$$g(\mathbf{x}^*, p)^T \eta = 0 \tag{3.10d}$$

$$\eta \geq 0 \tag{3.10e}$$

where (3.10d) *are complementarity conditions. The Karush-Kuhh-Tucker (KKT) conditions are also referred to as the first-order necessary conditions.*

An important observation is that it may have many sets of multipliers (ν, η) or none that satisfy (3.10) for a given KKT point $\mathbf{x}^*$ of (3.7). However, if LICQ is satisfied, it implies that the multipliers (ν, η) are unique.

Definition 21. *(SC, [58]) Given a local solution x^* of* (3.7) *and multipliers (ν, η) that satisfy* (3.10)*, the strict complementarity condition (SC) holds if $\eta_j - g_j(\mathbf{x}^*, p) > 0$ for each $j \in \mathcal{J}(\mathbf{x}^*)$. Equivalently, η_j is strictly positive when the corresponding inequality constraint becomes active.*

After introducing the first-order conditions for optimality, we now take a look at second-order conditions.

Definition 22. *(SOSC, [58]) The second-order sufficient condition (SOSC) holds if KKT conditions* (3.10) *are satisfied for $\mathbf{x}^*$ and multipliers (ν, η), and also*

$$q^T \nabla_{\mathbf{xx}} \mathcal{L}(\mathbf{x}^*, \nu, \eta, p) q > 0 \quad \textit{for all } q \neq 0 \tag{3.11}$$

is satisfied under the following condition,

$$\begin{aligned} &\nabla c_i(\mathbf{x}^*, p)^T q = 0, \quad i \in \mathcal{I}, \\ &\nabla g_j(\mathbf{x}^*, p)^T q = 0, \quad j \in \mathcal{J}(\mathbf{x}^*) \quad \textit{if } \eta_j > 0, \\ &\nabla g_j(\mathbf{x}^*, p)^T q \leq 0, \quad j \in \mathcal{J}(\mathbf{x}^*) \quad \textit{if } \eta_j = 0. \end{aligned} \tag{3.12}$$

then $\mathbf{x}^*$ is a strict local solution for (3.7).

Definition 23. *(SSOSC, [59]) The strong second order sufficient condition (SSOSC) holds if KKT conditions* (3.10) *are satisfied for $\mathbf{x}^*$ and multipliers (ν, η), and also*

$$q^T \nabla_{\mathbf{xx}} \mathcal{L}(\mathbf{x}^*, \nu, \eta, p) q > 0 \quad \textit{for all } q \neq 0 \tag{3.13}$$

is satisfied under the following condition,

$$\begin{aligned} &\nabla c_i(\mathbf{x}^*, p)^T q = 0, \quad i \in \mathcal{I}, \\ &\nabla g_j(\mathbf{x}^*, p)^T q = 0, \quad j \in \mathcal{J}(\mathbf{x}^*) \quad \textit{if } \eta_j > 0. \end{aligned} \tag{3.14}$$

then $\mathbf{x}^*$ is a strict local solution for (3.7).

We also introduce a generalization of LICQ as follows:

Definition 24. *(MFCQ,[58]) The Mangasarian-Fromovitz Constraint Qualification (MFCQ) holds at the KKT point* $\mathbf{x}^*$ *if,*

$$\begin{aligned}
&\nabla_{\mathbf{x}} c_i(\mathbf{x}^*, p)^T q = 0, \quad i \in \mathcal{I}, \\
&\nabla_{\mathbf{x}} g_j(\mathbf{x}^*, p)^T q < 0, \quad j \in \mathcal{J}(\mathbf{x}^*).
\end{aligned} \tag{3.15}$$

and the gradients of equality constraints $\{\nabla c_i(\mathbf{x}^*, p), \ \ i \in \mathcal{I}\}$ *are linearly independent.*

MFCQ is a weaker version of LICQ, which implies the boundedness of Lagrange multipliers (ν, η) that satisfy KKT conditions (3.10).

3.2.2 Interior point algorithm

The interior-point method, also referred to as barrier method, has been seen as a powerful tool to solve large-scale nonlinear programming problems. One successful barrier method implementation is IPOPT [57]. The problem under consideration (3.7) is reformulated as:

$$\min_{\mathbf{x}} \ F(\mathbf{x}; p) \tag{3.16a}$$

$$\text{s.t.} \ \ c(\mathbf{x}, p) = 0, \tag{3.16b}$$

$$\mathbf{x} \geq 0 \tag{3.16c}$$

where (3.16) can be obtained by adding nonnegative slack variables. For an optimal control problem, $\mathbf{x}$ includes all the state and control variables, and p is the parameter vector.

IPOPT handles the inequality constraints implictly by a barrier function in the objective and solves the following problem:

$$\min_{\mathbf{x}} \ F(\mathbf{x}; p) - \mu \sum_{i=1}^{n_{\mathbf{x}}} ln(\mathbf{x}_i) \tag{3.17a}$$

$$\text{s.t.} \ \ c(\mathbf{x}, p) = 0. \tag{3.17b}$$

Given a barrier parameter $\mu > 0$ but negligibly small, the primal-dual system of optimality conditions of (3.17) are solved directly at p_0,

$$\begin{aligned} \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*, \lambda^*, \upsilon^*; p_0) &= \nabla_{\mathbf{x}} F(\mathbf{x}^*; p_0) + \nabla_{\mathbf{x}} c(\mathbf{x}^*; p_0)\lambda^* - \upsilon^* = 0 \\ c(\mathbf{x}^*; p_0) &= 0 \\ \mathbf{X}^*\mathbf{V}^* e &= \mu e \end{aligned} \tag{3.18}$$

In order to solve (3.18), Newton's method is applied to obtain the search direction. Given a current iterate $[\mathbf{x}_k, \lambda_k, \upsilon_k]^T$, a linearization of (3.18) leads the search direction $[d_k^{\mathbf{x}}, d_k^{\lambda}, d_k^{\upsilon}]^T$,

$$\begin{bmatrix} \nabla_{\mathbf{xx}}\mathcal{L}(s_k(\mu; p)) & \nabla_{\mathbf{x}} c(s_k(\mu; p)) & -I \\ \nabla_{\mathbf{x}} c(s_k(\mu; p))^T & 0 & 0 \\ \mathbf{V}(\mu; p) & 0 & \mathbf{X}(\mu; p) \end{bmatrix} \begin{bmatrix} d_k^{\mathbf{x}} \\ d_k^{\lambda} \\ d_k^{\upsilon} \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{x}}\mathcal{L}(s_k(\mu; p)) \\ c(\mathbf{x}_k(\mu; p)) \\ \mathbf{X}_k\mathbf{V}_k e - \mu e \end{bmatrix} \tag{3.19}$$

with $\mathbf{V} = diag(\upsilon), \mathbf{X} = diag(\mathbf{x})$, and $e^T = [1, .., 1]$.

After solving a sequence of problems (3.19) with $\mu \to 0$, the solution of (3.17) approaches the solution of the original NLP (3.16). The primal and dual optimal solution vector is $s(\mu; p)^T = [\mathbf{x}(\mu; p)^T, \lambda(\mu; p)^T, \upsilon(\mu; p)^T]$.

3.2.3 NLP sensitivity

Theorem 25. *(NLP Sensitivity) [59, 60]. If $F(\cdot)$ and $c(\cdot)$ of the parametric NLP problem (3.16) are twice continuously differentiable in a neighborhood of the nominal primal and dual solution $s^*(p_0)$ and this solution satisfies the strong complementarity (SC), linear independence constraint qualifications (LICQ) and strong second order sufficient conditions (SSOSC) then,*

- *$s^*(p_0)$ is an isolated local minimizer of $\mathcal{P}(p_0)$ and the associated Lagrange multipliers are unique.*
- *For p in a neighborhood of p_0 there exists a unique, continuous and differentiable vector function $s^*(p)$ which is a local minimizer satisfying SSOSC and LICQ for $\mathcal{P}(p)$.*

- *There exists a positive Lipschitz constant L_S such that $|s^*(p) - s^*(p_0)| \leq L_S|p - p_0|$ where $|\cdot|$ is the Euclidean norm.*
- *There exists a positive Lipschitz constant L_J such that the optimal cost values $J_N(p)$ and $J_N(p_0)$ satisfy $|J_N(p) - J_N(p_0)| \leq L_J|p - p_0|$.*

Theorem 25 can also be extended to Mangasarian-Fromovitz constraint qualification (MFCQ), which is less restrictive, but requires a more complex sensitivity step. Details of this extension are described in [61] and are beyond the scope of this dissertation.

When SC, LICQ and SSOSC are satisfied at $s(\mu; p_0)$, we apply Theorem 25 to (3.19) and solve the following linear system for the sensitivity,

$$M(s(\mu; p_0))\Delta s = -N(s(\mu; p_0); p) \tag{3.20}$$

where $M(s(\mu; p_0)) = \begin{bmatrix} \nabla_{\mathbf{xx}}\mathcal{L}(s(\mu; p_0)) & \nabla_{\mathbf{x}}c(s(\mu; p_0)) & -I \\ \nabla_{\mathbf{x}}c(s(\mu; p_0))^T & 0 & 0 \\ V(\mu; p_0) & 0 & X(\mu; p_0) \end{bmatrix}$ is called the KKT matrix, and $N(s(\mu; p_0); p) = \begin{bmatrix} \nabla_{\mathbf{x}}\mathcal{L}(s(\mu; p_0); p) \\ c(\mathbf{x}(\mu; p_0); p) \\ 0 \end{bmatrix}$.

We write the optimal solution with perturbed parameter as,

$$s(0; p) = s^*(\mu; p_0) + \Delta s + O(||p - p_0||^2) + O(\mu) \tag{3.21}$$

A first-order approximation solution is computed as,

$$\tilde{s}(p) = s^*(\mu; p_0) + \Delta s \tag{3.22}$$

Then the difference between the optimal solution and approximate solution of the original problem (3.16) is,

$$|s(p) - \tilde{s}(p)| = O(||p - p_0||^2) \tag{3.23}$$

Assume the differentiability and Lipschitz continuity of the optimal solution, a positive Lipschitz constant L_s exists such that

$$|s(p) - \tilde{s}(p)| \leq L_s|p - p_0|^2 \tag{3.24}$$

This approximate solution Δs can be computed either by sIPOPT [62] (an extension of IPOPT) or with assistance of CasADi [63] for an update of p using the previous solution at p_0.

The NLP sensitivity provides a great opportunity for optimal control applications where a control input is imminently needed and an accurate solution is not required. Three significant applications of NLP sensitivities are advanced-step NMPC [64], advanced-step multistage NMPC [65] and samNMPC [66], all of which will be explored in this dissertation.

Chapter 4

Advanced-step multistage nonlinear model predictive control

Nonlinear model predictive control (NMPC) has been popular in many applications, especially when constraint satisfaction is critical. However, due to plant-model mismatch and disturbances, robust NMPC generally faces three challenges: robust performance, real-time implementation, and stability. This chapter mainly addresses the first two issues.

In this chapter, we propose an efficient parallelizable advanced-step multistage NMPC (as-msNMPC) controller that exploits the structure of scenario trees and is robust to two types of uncertainties. In the background, as-msNMPC precomputes possible control actions one step ahead as parallelizable subproblems. Depending on whether unmeasured noise is present, the online computation step selects the corresponding control with possibly additional NLP sensitivity correction. The performance of as-msNMPC is demonstrated on two case studies and compared with the performance of several other robust NMPC schemes.

The work presented in this chapter has been largely published in [65].

4.1 Current robust NMPC techniques

We represent the dynamics of the plant under uncertainty by the following discrete-time model:

$$x_{k+1} = f(x_k, u_k, d_k) + w_k \tag{4.1}$$

where $x_k \in \mathbb{X} \subset \mathbb{R}^{n_x}, u_k \in \mathbb{U} \subset \mathbb{R}^{n_u}$ are the plant states and controls, and $d_k \in \mathbb{D} \subset \mathbb{R}^{n_d}$, $w_k \in \mathbb{W} \subset \mathbb{R}^{n_x}$ are the parametric disturbances and bounded additive noise, defined at time step t_k where $k \geq 0$ is the time index. Without loss of generality, the mapping $f : \mathbb{R}^{n_x+n_u+n_d} \to \mathbb{R}^{n_x}$ with $f(0,0,0) = 0$ represents the nominal model.

Sources of uncertainty are divided into two distinctive categories. *Type 1 uncertainty* is denoted as d_k: model parameters that may be uncertain and will be realized before the next step. *Type 2 uncertainty*, denoted as w_k, is unmeasured noise, e.g. process and measurement noise, which will not be realized within one sampling time, if ever. w_k often contributes to the discrepancies between predicted states and actual states. In the following section, we deploy different techniques to handle both error types explicitly. Let's first start with the existing techniques.

4.1.1 Standard NMPC

The basic concepts of standard NMPC can be found in [67] and Section 2.4. After obtaining the current state x_k at t_k from the measurements, one solves for a sequence of states and controls for the next N steps. We use z_l and v_l to represent the state and control variables in the controller at t_{k+l}, respectively.

Standard NMPC neglects both uncertainties. Given the plant dynamics as (4.1), we write the standard NMPC as the following nonlinear programming (NLP) problem:

$$\min_{z_l, v_l} \quad \phi(z_N, \bar{d}_{N-1}) + \sum_{l=0}^{N-1} \varphi(z_l, v_l, \bar{d}_l) \tag{4.2a}$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l, \bar{d}_l) \qquad l = 0, ..., N-1 \tag{4.2b}$$

$$z_0 = x_k \tag{4.2c}$$

$$z_l \in \mathbb{X}, v_l \in \mathbb{U}, z_N \in \mathbb{X}_f \tag{4.2d}$$

The stage cost is given by $\varphi(\cdot,\cdot,\cdot) : \mathbb{R}^{n_x+n_u+n_d} \to \mathbb{R}$ and terminal cost is $\phi(\cdot,\cdot) : \mathbb{R}^{n_x+n_d} \to \mathbb{R}$. For a control problem with a setpoint tracking objective, the stage cost and terminal cost terms are often quadratic. The constraints (4.2b) represent the dynamic model used in the standard NMPC controller, which keeps the model parameter at its nominal value $\bar{d}_l$ for all stages. (4.2c) sets the initial state of the controller at the current plant state x_k at t_k. The last constraint (4.2d) denotes the bound constraints, where $\mathbb{X}$ and $\mathbb{U}$ are domains for state and control variables, respectively. $\mathbb{X}_f \in \mathbb{X}$ where $\mathbb{X}_f$ is the terminal region.

After solving the optimization problem (4.2) for standard NMPC, the control action injected into the plant at time step k is the first stage of control profile $u_k = v_0^*$. Later this feedback control law is also referred to as $u_k = h(x_k)$ with $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$. After one time step, the plant evolves as in (4.1) with one step forward at t_{k+1}, the new state x_{k+1} is obtained, and the procedure repeats.

Depending on the problem size, it often requires a non-trivial amount of time to solve (4.2) in order to obtain the control action v_0^*. The time difference between measuring the state information and injecting the control action for the same step is referred to as *computational delay*. Avoiding this delay is the key advantage to advanced-step multistage NMPC, which will be discussed in Section 4.2.

Finally, the rolling horizon nature of standard NMPC provides some inherent robustness [7, 8], but its performance deteriorates under uncertainty. We therefore summarize several robust NMPC schemes that handle uncertainty explicitly.

4.1.2 Ideal multistage NMPC

The main idea behind multistage NMPC is to explicitly account for the model uncertain parameters and their evolution and propagation throughout the time horizon. Similar problem settings have been established in stochastic programming literature, where a scenario tree is utilized to capture the uncertainty evolution. The ideal multistage NMPC is built upon the basic scenario tree structure. To distinguish from Section 4.2, with ideal multistage NMPC (ideal-msNMPC) we assume computational delay is neglected so that the control inputs can be injected to the plant immediately.

Conventionally, two major assumptions are made:

1. Each uncertain parameter follows a discretized probability distribution with a finite set of values.

2. The resolution of uncertain parameters occurs before the next time step, which qualifies the nature of uncertain parameters to be exogeneous [68].

The prediction horizon N for NMPC problems is generally long enough to lead to application of *multistage* stochastic programming [20]. Consistent with the terminology in stochastic programming, the zeroth control variable v_0 is commonly referred to as the *here-and-now* decision, and the remaining control variables ($v_l, \forall l \geq 1$) are *wait-and-see* decisions, which are often referred to as *recourse* actions (i.e. future decision variables that compensate for the realized uncertainty of the current step). In fact, different realizations result in different recourse variables in order to achieve the corresponding objective. Further discussion of this topic is beyond the scope of our paper and can be found in [20, 69].

Consider the state of the plant at the current step t_k at the root node of the tree, and that the system has one uncertain parameter that can take $|\mathbb{Q}|$ representative values from its probability distribution. Following the canonical examples in stochastic programming

[16], we also choose $|\mathbb{Q}| = 3$ where $\mathbb{Q} = \{\text{max, nominal, min}\}$. As a result, three distinctive states will be reached at t_{k+1}, which represents all possible states with respect to three possible uncertain parameter realizations (without considering additive noise). As the process moves forward in time, the states evolve with 9 states at t_{k+2}, and then 27 states at t_{k+3}, as seen in Fig. 4.1 for a prediction horizon of three. Note that a *scenario* is defined as a sequence of states from the root node to a leaf node.

If the total length of prediction horizon is N, the number of states at the end of the prediction horizon is $|\mathbb{Q}|^{n_d \cdot N}$, where n_d is the dimensionality of the uncertain parameter vector. By construction, the scenario tree captures every possible uncertainty evolution profile from t_k to t_{k+N}. However, since N needs to be long enough to match plant time constraints in MPC applications, the number of states at N grows exponentially. Hence, the resulting optimization problem size explodes.

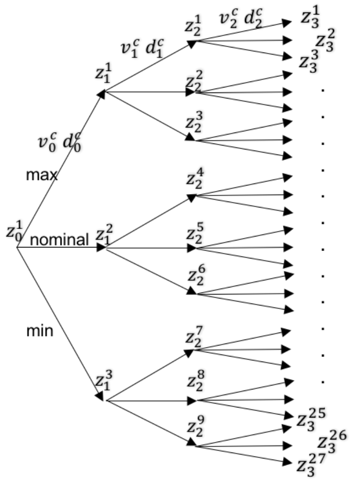


Figure 4.1: Prediction horizon N = 3, the fully-branched 27-scenario tree.

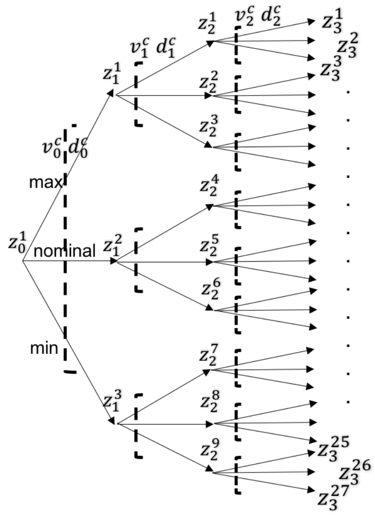


Figure 4.2: Prediction horizon N = 3, the fully-branched 27-scenario tree with non-anticipativity constraints (NACs).

When implementing the control action, regardless of the number of scenarios, there is only one control action v_0 that can be injected into the plant at time t_k. Owing to the construction of the scenario tree, the controls needn't be the same for every step and every scenario; they only need to be the same if they act on the same state (e.g. in Fig. 4.2, $v_0^1 = v_0^2$ but v_1^1 and v_1^6 are not restricted to be the same). Before the uncertain parameters are realized, the system requires the control action to be determined. Hence, constraints that enforce the same control action for scenarios that stem from the same state node are called *Non-Anticipativity Constraints (NACs)*. Since the number of NACs also grows exponentially with respect to N, the NLP problem rapidly becomes intractable. See (5.17) for multistage NMPC formulation for a fully expanded tree.

In order to obtain an online-tractable problem formulation, [16] apply a robust horizon to limit scenario tree branching up to a shorter horizon (e.g. Fig. 4.3 shows a truncated scenario tree with robust horizon of two). A robust horizon N_r means that the scenario tree develops in the same way as the fully-expanded scenario tree *up to* t_{N_r}; after t_{N_r}, the uncertain parameters remain at their last values in robust horizon throughout the rest of the prediction horizion. In fact, for all $c \in \mathbb{C}$, $d^c_{N_r-1} = d^c_{N_r} = d^c_{N_r+1} = ... = d^c_{N_p-1}$. The benefit of applying robust horizons is that the number of scenarios is significantly reduced to $|\mathbb{C}| = |\mathbb{Q}|^{n_d \cdot N_r}$ instead of $|\mathbb{Q}|^{n_d \cdot N}$ with $N_r \ll N$. While the truncated form of scenario tree will not capture every evolution of uncertainty within N as the fully-expanded scenario tree, a justification could be that all controls for all stage index $l > 0$ will be recalculated in the moving horizon setting, considering that only the immediate control is implemented to the plant. Frequently, the branches that are neglected by applying a robust horizon have less impact on MPC applications. As developed in [16], the optimization formulation for ideal multistage NMPC (assuming no computational delay) is shown as (4.3) (also denoted as $\mathcal{P}^{id}_N$).

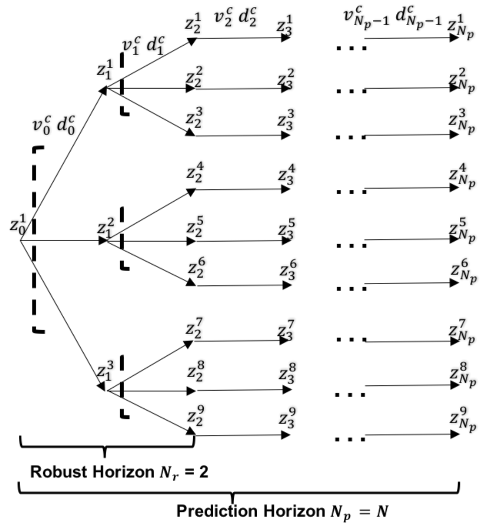


Figure 4.3: Robust horizon N_r = 2, the 9-scenario tree with prediction horizon $N_p = N$.

$$J_N^{id}(x_k) = \min_{z_l^c, v_l^c} \quad \sum_{c \in \mathbb{C}} p^c \Big(\phi(z_N^c, d_{N-1}^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l^c, d_l^c) \Big) \tag{$\mathcal{P}_N^{id}(x_k)$}$$

$$\text{s.t.} \quad z_{l+1}^c = f(z_l^c, v_l^c, d_l^c) \quad l = 0, ..., N-1 \tag{4.3a}$$

$$z_0^c = x_k \tag{4.3b}$$

$$v_l^c = v_l^{\bar{c}} \quad \text{if} \quad z_l^c = z_l^{\bar{c}} \tag{4.3c}$$

$$d_{l-1}^c = d_l^c \text{ for } l = N_r, \dots N-1 \tag{4.3d}$$

$$z_l^c \in \mathbb{X}, v_l^c \in \mathbb{U}, z_N^c \in \mathbb{X}_f^c, d_l^c \in \mathbb{D} \tag{4.3e}$$

$$\forall c, \bar{c} \in \mathbb{C} \tag{4.3f}$$

The optimal objective for (4.3) is denoted as $J_N^{id}(x_k)$, where the parameter x_k is the current state of the plant at t_k and N is the length of prediction horizon. z_l^c, v_l^c, d_l^c represent the state, control and parametric disturbance at stage l and scenario c. (4.3c) is the NAC constraint, where scenario c and $\bar{c}$ share the same control if the corresponding states of c

and $\bar{c}$ are the same at stage l.

4.1.3 Min-max NMPC

Min-max MPC [10, 12] has been seen as an effective robust MPC scheme. The name *min-max* comes from the two-layer operator, where the inner layer (max operator) rules on the worst case and the outer layer (min operator) optimizes the cost function for the selected (worst) scenario. One unified control profile applies to all scenarios, and same constraints are enforced for every scenario including the worst scenario.

The same scenario tree structure of ideal multistage NMPC is used for min-max NMPC. One major difference between ideal multistage NMPC to min-max NMPC is the objective, where ideal multistage NMPC optimizes a weighted average performance for all scenarios, yet min-max NMPC only optimizes the performance for the worst-case scenario.

The NLP problem is defined as:

$$\min_{z_l^c, v_l} \max_{c} \quad \phi(z_N^c, d_{N-1}^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l, d_l^c) \tag{4.4a}$$

$$\text{s.t.} \quad z_{l+1}^c = f(z_l^c, v_l, d_l^c) \quad l = 0, ..., N-1 \tag{4.4b}$$

$$z_0^c = x_k \tag{4.4c}$$

$$z_l^c \in \mathbb{X}, v_l \in \mathbb{U}, z_N^c \in \mathbb{X}_f^c, c \in \mathbb{C}, d_l^c \in \mathbb{D} \tag{4.4d}$$

where $\mathbb{C}$ is the set of scenarios.

In practice, problem (4.4) is reformulated and solved as a standard single level optimization problem. A scalar variable is introduced to serve as the upper bound in the form of constraints to replace the max operator. The reformulation can be written as:

$$\min_{z_l^c, v_l, \mu} \quad \mu \tag{4.5a}$$

$$\text{s.t.} \quad \phi(z_N^c, d_{N-1}^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l, d_l^c) \leq \mu, \quad l = 0, ..., N-1 \quad \forall c \tag{4.5b}$$

$$z_{l+1}^c = f(z_l^c, v_l, d_l^c) \tag{4.5c}$$

$$z_0^c = x_k \tag{4.5d}$$

$$z_l^c \in \mathbb{X}, v_l \in \mathbb{U}, z_N^c \in \mathbb{X}_f^c, c \in \mathbb{C}, d_l^c \in \mathbb{D} \tag{4.5e}$$

4.1.4 Robust NMPC with explicit back-off constraints

Another strategy of optimizing under uncertainty is robust NMPC with backoff constraints. One can directly modify the original inequality constraints in the standard NMPC formulation (4.2) with backoff terms so that the same problem structure is maintained.

To explain this in mathematical form, we first re-write the bound constraints in standard NMPC formulation (4.2d) as the following equivalent expression:

$$g(z_l, v_l) \leq \mathbf{0} \quad l = 0, ..., N \tag{4.6}$$

where $g(\cdot,\cdot) : \mathbb{R}^{n_x+n_u} \rightarrow \mathbb{R}^B$, and B is defined as the number of bound constraints (i.e. inequality constraints) in the formulation (4.2d). By adding backoff terms, (4.6) would become:

$$g(z_l, v_l) + b \leq \mathbf{0} \quad l = 0, ..., N \tag{4.7}$$

where term $b \in \mathbb{R}^B$ is the *backoff* term. In this way, the problem of handling model uncertainty is converted to calculation of the backoff term *offline* such that the original inequality constraints (4.6) are satisfied *online* with pre-determined probability threshold. If the probability is set to be 1, then it can be interpreted that the constraints are met even for the worst scenario. Generally, the backoff term is nonnegative, implying that the feasible region of the problem needs to be tightened to guarantee robustness. The case of negative backoff terms, where the inequality constraints can be relaxed, is not in the scope of this work. For discussions regarding negative backoff terms, one can refer to [70].

The offline procedure to obtain the backoff term is by performing open-loop Monte Carlo simulation, i.e. there is no feedback from the real-time plant data to the controller except for the initial state value x_0. The updated optimization problem becomes:

$$\min_{z_l, v_l} \quad \phi(z_N, \bar{d}_{N-1}) + \sum_{l=0}^{N-1} \varphi(z_l, v_l, \bar{d}_l) \tag{4.8a}$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l, \bar{d}_l) \tag{4.8b}$$

$$z_0 = x_k \tag{4.8c}$$

$$g(z_l, v_l) + b \leq \mathbf{0} \qquad l = 0, ..., N \tag{4.8d}$$

After obtaining the back-off term from the offline Monte Carlo simulation, on-line NMPC (4.9) is solved at each sampling time with updated initial values from plant. In practice, only the first control action matters because the plant only implements the first step. In order to monitor feasibility strictly on plant behavior, a soft penalty is applied to all steps. The online NMPC formulation is as below:

$$\min_{z_l, v_l} \quad \phi(z_N, \bar{d}_{N-1}) + \sum_{l=0}^{N-1} \varphi(z_l, v_l, \bar{d}_l) + M \sum_{l=0}^{N} \delta_l \tag{4.9a}$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l, \bar{d}_l) \tag{4.9b}$$

$$z_0 = x_k \tag{4.9c}$$

$$g(z_l, v_l) + b \leq \delta_l \qquad l = 0, ..., N \tag{4.9d}$$

$$\delta_l \geq 0 \qquad l = 0, ..., N \tag{4.9e}$$

The following procedure (Algorithm 1) demonstrates the steps to obtain the back-off term within NMPC framework. More detailed methodologies to compute the back-off term can be found in [71].

Algorithm 1: Robust NMPC with backoff terms from Monte Carlo

```
1 Initialize $b = 0$ and obtain the offline optimal control trajectory $u(t)$ by solving (4.8)
2 Apply control profile $u(t)$ to perform Monte Carlo simulation with known uncertainty
    probability distribution, and obtain state profiles.
3 Calculate the backoff term $b^*$ for $g(z_l, v_l)$ from the largest boundary violation
    magnitude in the Monte Carlo simulation result.
4 if $||b^* - b||_2 \leq \epsilon$ (tolerance) then
5     go to step 9;
6 else
7     update (4.8) with the new backoff term $b = b^*$, obtain the new offline optimal
        control profile and go back to step 2;
8 end
9 Apply $b$ to solve the online optimization (4.9) to see the optimization performance and
    constraint satisfaction.
```

4.1.5 Perfect information

As the name suggests, perfect information assumes that the controller obtains the uncertainty information *a priori* up to certain steps ahead of the current step. In other words, the perfect information NMPC controller uses a model in accordance with the plant up to certain steps, after which the controller keeps the same uncertain parameter values.

Since uncertainty cannot be predicted, the perfect information NMPC does not carry a practical implementation. However, its performance provides a meaningful benchmark that no robust NMPC strategies can surpass. Hereby the research focus of practical robust NMPC is to get as close as possible to the performance of perfect information.

The optimization formulation that is solved at every time step for each scenario $c \in \mathbb{C}$ is written as:

$$\min_{z_l, v_l} \quad \phi(z_N^c, d_{N-1}^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l^c, d_l^c) \tag{4.10a}$$

$$\text{s.t.} \quad z_{l+1}^c = f(z_l^c, v_l^c, d_l^c), \qquad l = 0, ..., N-1 \tag{4.10b}$$

$$z_0^c = x_k \tag{4.10c}$$

$$z_l^c \in \mathbb{X}, v_l^c \in \mathbb{U}, z_N^c \in \mathbb{X}_f^c, d_l^c \in \mathbb{D}, c \in \mathbb{C} \tag{4.10d}$$

Notice that the perfect information formulation does not need to have NACs because the uncertainty is assumed known, thus (4.10) can be solved in a parallel fashion as each scenario is independent.

4.2 Advanced-step multistage NMPC scheme

4.2.1 Advanced-step and NLP sensitivity

To directly tackle nontrivial delays from the solution of NMPC problem (4.2), *advanced-step* NMPC was designed to separate the computational workload into two parts: *background* and *online*, and ultimately have negligible computation delays *online* [64]. Advanced-step NMPC (*asNMPC*) also carries the same nominal and robust stability properties of standard NMPC, which will be discussed and proved in Chapter 5.

In order to apply either advanced-step NMPC or advanced-step multistage NMPC, NLP sensitivity is used to have a quick perturbed solution. A brief NLP sensitivity recap is provided here. Assuming the dynamic optimization problem of interest can be reformulated as a canonical NLP formulation as follows:

$$\min_{\mathbf{x}} \ F(\mathbf{x}; p) \tag{4.11a}$$

$$\text{s.t.} \ c(\mathbf{x}, p) = 0, \ \ \mathbf{x} \geq 0 \tag{4.11b}$$

where $\mathbf{x}$ represent all the primal variables, and p is the parameter. Assume the original NLP has the parameter value p_0. If we send (4.11) to a solver using the barrier method such as IPOPT [57], instead of solving (4.11) directly, IPOPT forms barrier subproblems as this:

$$\min_{\mathbf{x}} \; F(\mathbf{x};p) - \mu \sum_{i=1}^{n_{\mathbf{x}}} ln(\mathbf{x}_i) \tag{4.12a}$$

$$\text{s.t. } c(\mathbf{x},p) = 0. \tag{4.12b}$$

where $\mu \geq 0$ is called the barrier parameter. (4.12) is free of inequality constraints. Under certain conditions, the solution of (4.12) converges to (4.11) as $\mu \to 0$. As a result, the optimal solution is denoted as $s(\mu;p_0)$.

Given a barrier parameter $\mu > 0$ but negligibly small, first order approximation of KKT conditions of (4.12) are solved directly at p_0,

$$\begin{aligned} \nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*, \upsilon^*; p_0) &= \nabla_{\mathbf{x}} F(\mathbf{x}^*; p_0) + \nabla_{\mathbf{x}} c(\mathbf{x}^*; p_0)\lambda^* - \upsilon^* = 0 \\ c(\mathbf{x}^*; p_0) &= 0 \\ \mathbf{X}^* \mathbf{V}^* e &= \mu e \end{aligned} \tag{4.13}$$

with $\mathbf{V} = diag(\upsilon), \mathbf{X} = diag(\mathbf{x})$, and $e^T = [1, .., 1]$. The primal and dual solution vector is $s(\mu;p)^T = [\mathbf{x}(\mu;p)^T, \lambda(\mu;p)^T, \upsilon(\mu;p)^T]$. When LICQ, SSOSC and SC are satisfied at $s(\mu;p_0)$, we apply Theorem 25 to (4.13) and solve the following linear system for the sensitivity,

$$M(s(\mu;p_0))\Delta s = -N(s(\mu;p_0);p) \tag{4.14}$$

where $M(s(\mu;p_0)) = \begin{bmatrix} \nabla_{\mathbf{xx}} L(s(\mu;p_0)) & \nabla_{\mathbf{x}} c(s(\mu;p_0)) & -I \\ \nabla_{\mathbf{x}} c(s(\mu;p_0))^T & 0 & 0 \\ V(\mu;p_0) & 0 & X(\mu;p_0) \end{bmatrix}$ is the KKT matrix,

and $N(s(\mu;p_0);p) = \begin{bmatrix} \nabla_{\mathbf{x}} L(s(\mu;p_0);p) \\ c(\mathbf{x}(\mu;p_0);p) \\ 0 \end{bmatrix}$ with $s(0;p) = s(\mu;p_0) + \Delta s + O(||p-p_0||^2) + O(\mu)$.

An approximate solution is computed as $\tilde{s}(p) = s^*(\mu; p_0) + \Delta s$, and sIPOPT computes this approximate solution for an update of p using the previous solution at p_0.

The main benefit of the sensitivity step is to save unnecessary computational effort by avoiding recalculation of the KKT matrix. The formation and factorization of the KKT matrix is the most expensive step of the NLP algorithm. By reusing the KKT matrix from the previous session at p_0, the sensitivity update spares solving an NLP from scratch. The reduced online computational workload is critical in real-time implementation of NMPC. The readers are referred to section 3.2.3 for more details regarding NLP sensitivity.

4.2.2 Advanced-step multistage NMPC (as-msNMPC)

Literature studies [16, 17, 18, 19, 21, 22, 24] on ideal multistage NMPC assume that the nonlinear optimization problem by a multistage scenario tree is solved without computational delay. In reality, there are limited resources to allow online computations, and the multistage scenario problem size is inevitably larger than its counterpart of standard NMPC, even with a short robust horizon. Instead, we exploit the recursive and decomposable nature of the scenario tree structure such that the advanced-step concept can be integrated seamlessly.

To illustrate the decomposition of scenario tree, consider the ideal 9-scenario tree in Fig. 4.3. At time t_k, with the current state x_k and control u_k from a previous calculation, obtaining three predicted states $\{x^1_{k+1|k}, x^2_{k+1|k}, x^3_{k+1|k}\}$ is straightforward, and every predicted state becomes the root node for a subtree starting at t_{k+1} with prediction horizon $N-1$, as shown in Fig. 4.5. Each subtree problem is mutually independent and can be solved separately. In addition, the subtree problem size is also smaller than the original tree problem approximately by $|\mathbb{Q}|^{-n_d}$, and will take less time to solve.

After decomposing the scenario tree, the next question is how to make the control action

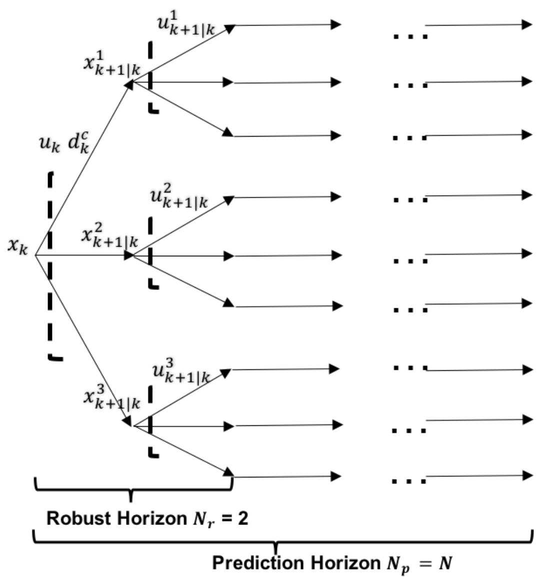


Figure 4.4: Robust horizon N_r = 2, the 9-scenario tree with prediction horizon N. (same as Fig. 4.3)

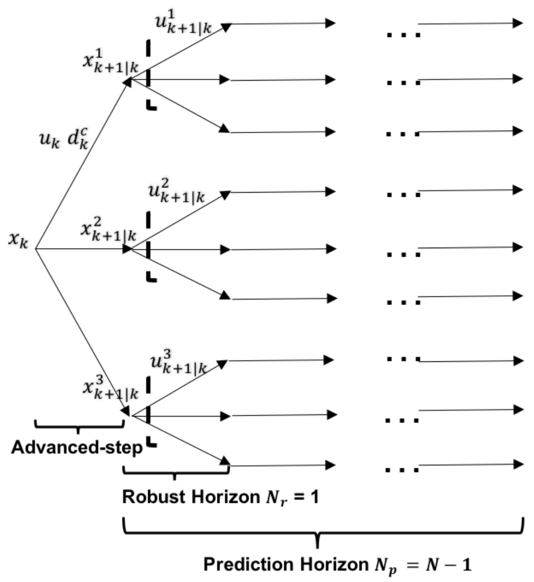


Figure 4.5: Example of advanced-step multistage NMPC transforming the original 9-scenario tree in Fig. 4.3

u_k available at t_k. Here, one has *a priori* knowledge of the finite set of uncertain parameters. At t_k the uncertainty in model parameters d^c_l takes values of d_j where $j \in \mathbb{Q}^{n_d}$ follows from the Cartesian product for possible values for each d_j.

With *a priori* knowledge of uncertainty selections, one can pre-compute the corresponding library of control actions for t_{k+1} in parallel. Fig. 4.5 shows the number of uncertain parameters $n_d = 1$ and d_k takes values $\{d^{max}, d^{nom}, d^{min}\}$ (i.e. $|\mathbb{Q}|^{n_d} = 3$). Consequently, if only Type 1 uncertainty appears, the plant evolves as one of the predicted states. On the other hand, if both Type 1 and 2 uncertainties appear in the system, a sensitivity correction using (4.14) is applied to the background solution to include unmeasured noise. In this way, as-msNMPC applies NLP sensitivity to approximate the corrected solution, as shown in the as-msNMPC algorithm below.

Background:

- Given x_k and u_k, predict the states at t_{k+1} for all d_k^c values as $x_{k+1|k}^c$. Solve $|\mathbb{Q}|^{n_d}$ individual problems $\mathcal{P}_{N-1}^{id}(x_{k+1|k}^c)$.
- Solving problems $\mathcal{P}_{N-1}^{id}(x_{k+1|k}^c)$ is independent and separable, so this step can be parallelized to ensure fast solution of background problem.

Online:

- At t_{k+1}, the actual state x_{k+1} is observed.

  Without Type 2 error, the actual state belongs to the set of background predictions $x_{k+1} \in \{x_{k+1|k}^c, c \in \mathbb{C}\}$. The only online step is to select the corresponding solution from the realized d_j and to repeat the background calculation at t_{k+1}.
- With Type 2 error, the actual state will not correspond to the background predictions. Instead, u_{k+1} is obtained from the sensitivity-based correction step (4.14). Once corrected, advance one step and repeat the background calculation at t_{k+1}.

4.2.3 Discussion on performances of as-msNMPC and ideal-msNMPC

By decomposing the same scenario tree into a set of subtrees, one can solve the subtree problems in the background, as in as-msNMPC. However, by construction, the solutions to ideal-msNMPC and as-msNMPC are *not* identical. As in Fig. 4.4 and Fig. 4.5, the effective robust horizon for as-msNMPC is one step shorter than the one of ideal-msNMPC $N_r^{as} = N_r^{ideal} - 1$. Similarly, for prediction horizon, $N^{as} = N^{ideal} - 1$. Hence, the implemented first control from as-msNMPC and ideal-msNMPC controllers are similar not the same assuming that one structure of the scenario tree is used for both controllers.

On the other hand, one *could* achieve identical performance between as-msNMPC and ideal-msNMPC under Type 1 only uncertainty, with a modified construction. If the subtrees for as-msNMPC have the same robust and prediction horizon lengths as the tree for the ideal case, i.e. $N_r^{as} = N_r^{ideal}$ and $N^{as} = N^{ideal}$, then one of the subproblems solved in as-msNMPC is the same as that for ideal-msNMPC. However, in this case, each subproblem has exactly the same size as that of ideal-msNMPC, which may become less advantageous if the background computational resource is limited.

4.3 Case studies

4.3.1 CSTR example

Consider the nonlinear CSTR benchmark problem [1], where the dynamics are described by the following four differential equations:

$$\begin{aligned}
\tfrac{dc_A}{dt} &= F(c_{A0} - c_A) - k_1 c_A - k_3 c_A^2 \\
\tfrac{dc_B}{dt} &= -F c_B + k_1 c_A - k_2 c_B \\
\tfrac{dT_R}{dt} &= F(T_{in} - T_R) + \tfrac{k_W A}{\rho c_p V_R}(T_K - T_R) - \tfrac{k_1 c_A \Delta H_{AB} + k_2 c_B \Delta H_{BC} + k_3 c_A^2 \Delta H_{AD}}{\rho c_p} \\
\tfrac{dT_K}{dt} &= \tfrac{1}{m_K c_{pK}}(\dot{Q}_K + k_W A(T_R - T_K))
\end{aligned} \tag{4.15}$$

where the reaction rate k_i follows the Arrhenius law, $k_i = k_{0,i} e^{\frac{-E_{A,i}}{R(T_R + 273.15)}}$. State variables are $[c_A, c_B, T_R, T_K]$, which are concentrations of component A and B, reactor temperature and coolant temperature, respectively. The control inputs are the inlet flow rate normalized by the reactor volume $F = \dot{V}_{in}/V_R$, and the heat removed by the coolant $\dot{Q}_K$.

The initial condition and constraints of the state variables are described in Table 4.2. And the upper and lower bounds of the manipulated variables are listed in Table 4.3.

The control task is to track a predefined set-point for the desired product concentration c_B. The setpoint is set to be $c_B^{ref} = 0.5$ for $t \leq 0.3h$ and $c_B^{ref} = 0.7$ for $t > 0.3h$. The

Parameter	Value	Unit
$k_{0,1}$	$1.287 \cdot 10^{12}$	h^{-1}
$k_{0,2}$	$1.287 \cdot 10^{12}$	h^{-1}
$k_{0,3}$	$9.043 \cdot 10^{9}$	$Lmol^{-1}h^{-1}$
$E_{A,1}/R$	9758.3	K
$E_{A,2}/R$	9758.3	K
$E_{A,3}/R$	8560	K
ΔH_{AB}	4.2	$kJmol^{-1}$
ΔH_{BC}	-11.0	$kJmol^{-1}$
ΔH_{AD}	-41.85	$kJmol^{-1}$
ρ	0.9342	kgL^{-1}
c_p	3.01	$kJkg^{-1}K^{-1}$
c_{pK}	2.0	$kJkg^{-1}K^{-1}$
A	0.215	m^2
V_R	10.0	L
m_k	5	kg
T_{in}	130.0	$^{\circ}C$
k_W	4032	$kJh^{-1}m^{-2}K^{-1}$
c_{A0}	5.1	$molL^{-1}$

Table 4.1: Parameter values of the CSTR, adapted from [1, 2]

uncertain system parameters are activation energy for the third reaction $E_{A,3}(t)$ and/or the inlet flow concentration $c_{A0}(t)$. Note that both uncertain parameters are time-variant

State	Init. cond.	Min.	Max.	Unit
c_A	0.8	0.1	5.0	$molL^{-1}$
c_B	0.5	0.1	5.0	$molL^{-1}$
T_R	134.14	50	140	$°C$
T_K	134.0	50	180	$°C$

Table 4.2: Initial conditions and state constraints

Control	Init. cond.	Min.	Max.	Unit
F	18.83	5	100	h^{-1}
$\dot{Q}_k$	-4495.7	-8500	0	kJh^{-1}

Table 4.3: Bounds on the manipulated variables

and of known distribution. The stage cost to be minimized is defined as below:

$$\varphi_l = (c_{Bl} - c_{Bl}^{ref})^2 + r_1 \Delta F_l^2 + r_2 \Delta \dot{Q}_{Kl}^2 \tag{4.16}$$

where c_B^{ref} is predefined setpoint for c_B, and $\Delta F_l = F_l - F_{l-1}$ and $\Delta \dot{Q}_K = \dot{Q}_{Kl} - \dot{Q}_{Kl-1}$ are the difference between consecutive steps after discretization for flow rate and cooling rate, respectively. The second and third terms are generally referred to as the penalty terms for control movements, and their corresponding penalty coefficients are $r_1 = 10^{-7}$ and $r_2 = 10^{-11}$. We choose the prediction horizon for the controller being $N = 40$ steps with a zero terminal constraint $z_N^c = z_{ss}(d_N^c)$ so that $\phi(z_N^c) = 0$. The final results are obtained after 120 runs with the entire control time period being 0.6 hours. All computational experiments are modeled in AMPL, and solved using IPOPT 3.12 with Intel i7-6700 Quad-Core CPU 3.40 GHz.

The robust horizon is set to be $N_r \leq 2$ in all robust NMPC schemes for uncertainties c_{A0} and $E_{A,3}$. For $n_d = 1, N_r = 2$ the 9-scenario structure applies to the ideal-msNMPC, perfect information and the worst case schemes. With as-msNMPC the corresponding three 3-scenario structures are solved and can be trivially parallelized. For the case with $n_d = 2, N_r = 2$, an 81-scenario tree is solved by ideal-msNMPC and nine 9-scenario trees are solved by as-msNMPC.

In the following subsections, three groups of case studies are performed. First, section 4.3.1.1 compares different robust NMPC schemes under uniform and non-uniform parameter distribution. Next, section 4.3.1.2 presents similar performances between as-msNMPC and ideal-msNMPC under Type 1 only and Type 1 + 2 uncertainty. Finally, section 4.3.1.3 tests the constraint satisfaction on different robust NMPC controllers, including both as-msNMPC and ideal-msNMPC.

4.3.1.1 Robust NMPC schemes results

Uniform distribution

We present case studies where five robust NMPC controllers are applied and compared. In this section, the uncertain parameter is activation energy $E_{A,3}$, and the level of uncertainty is set to be $\pm 10\%$ with respect to its nominal value. The uncertainty in the plant is assumed time-varying such that for each horizon d_k can take any value from $\{\mathsf{max}, \mathsf{nominal}, \mathsf{min}\}$. The performances of standard NMPC, min-max NMPC, back-off, perfect information and ideal-multistage NMPC, are presented and compared in Table 4.4.

The performance is gauged based on an averaged accumulated cost of 10 random runs. Each accumulated cost is the implemented stage costs accumulated over 120 runs, i.e. $\sum_{k=1}^{120}(c_{Bl} - c_{Bl}^{ref})^2$. Then we repeat this procedure for 10 random seeds of uncertainty realizations and record the averaged accumulated costs. In the case of back-off, Monte

Carlo simulation is performed with 200 random seeds. Each random seed corresponds to a sequence of uncertainty realizations for the full horizon N, where at each time $E_{A,3}$ takes its -10%, nominal and +10% value with equal probability. Then the back-off term is determined as the maximum constraint violation magnitude.

The averaged accumulated cost and CPU seconds are listed in Table 4.4 for both 3-scenario and 9-scenario cases for five robust schemes. Standard NMPC, where $E_{A,3}$ remains at its nominal value, is indifferent to scenario trees. Due to time-varying uncertainty in the plant, standard NMPC suffers from significant plant-model mismatch, which results in the highest tracking error among all the formulations. At the same time, standard NMPC also violates state constraints, which will be elaborated in Section 4.3.1.3.

	# Scenarios	Standard	Perfect	Min-max	Back-off	Ideal-ms
Costs	3	1.70659	0.21432	1.10159	1.63201	0.37350
	9		0.15349	1.10191		0.24160
CPUs	3	0.1315	0.1208	0.2712	0.1777	0.3057
	9		0.1170	0.7936		0.6393

Table 4.4: Averaged accumulated stage costs and CPU seconds for standard, perfect information, min-max NMPC, back-off and ideal-multistage NMPC under 3-scenario and 9-scenario tree structure for uncertainty is $d = E_{A,3} \pm 10\%$.

The perfect information case gives the best performance among all formulations as shown in Table 4.4. In fact, 9-scenario behaves even better than 3-scenario because 9-scenario assumes two steps of perfect information on uncertainty are obtained, instead of one step in the 3-scenario case. Despite its desirable performance, perfect information is an unrealistic situation which postulates that the controller is able to predict the future uncertainty realization. However, the objective value of the perfect information case provides a lower performance bound for all other robust NMPC schemes.

Both min-max and back-off approaches stay robust despite uncertainties. In the case of min-max NMPC, there is only one control profile minimizing the worst cost scenario. While constraints are satisfied for all possible scenarios, the feasible region is shrunk and hence the controller behaves cautiously. In comparison, the back-off constraint approach appears to be more conservative and sacrifices even more performance. On the computational side, back-off outperforms other methods due to a single-scenario on-line computational load, while min-max increases on-line efforts because of its larger multi-scenario problem.

Lastly, ideal-multistage NMPC formulation is able to get closer to perfect information compared to min-max NMPC and back-off. Multistage NMPC applies non-anticipativity constraints to control variables only within the robust horizon, leaving the future manipulated variables as recourse variables to counteract future uncertainties. These extra degrees of freedom decrease model conservatism and consequently achieve a lower cost. Computationally, ideal-msNMPC inevitably increases the on-line CPU seconds owing to the multi-scenario model and additional non-anticipativity constraints; this is remedied by the advanced-step multistage approach.

Non-uniform distribution

In the previous case study, we assume that each scenario is of equal probability $\frac{1}{|\mathbb{C}|}$. However, that is not always the case in reality. We now explore two non-uniform distributions: symmetric and skewed. The *symmetric* case represents a discretized Gaussian distribution, i.e. a higher probability (0.5) for the nominal scenario and lower probability (0.25) for the other two closer to the tail. The *skewed* case represents the distribution where one of the scenario closer to the tail has a higher probability (0.5) while the other two are of lower probability (0.25). To compare with the results of the uniform distribution, we keep the same 3-scenario structure among all distributions, i.e. every distribution has the same three discrete values ($-10\%, 0\%$ and $+10\%$) but different probability values as shown in Table 4.5. The uniform distribution is the same with the ones in the previous sections. The CPU time in the case of non-uniform distribution is similar to that of the uniform distribution shown in Table 4.4.

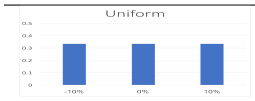
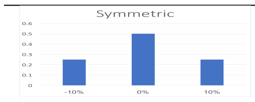
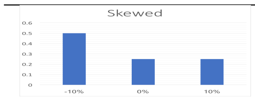
Parameter probability distribution	Standard	Perfect	Multistage	Min-max	Back-off
	1.70659	0.21432	0.37350	1.10159	1.63201
	0.82820	0.15506	0.27143	0.92928	1.77445
	4.55494	0.55056	0.70611	1.63751	0.76611

Table 4.5: Averaged accumulated tracking error (cost) for standard, perfect information, multistage, min-max and back-off under 3-scenario tree structure for uncertainty is $E_{A,3}$ when the parameter probability distribution is uniform, symmetric and skewed. The gray column implies infeasible solver status with the corresponding formulation.

Table 4.5 shows a consistent trend that multistage NMPC demonstrates a clear advantage over other robust NMPC approaches regardless of the uncertain parameter probability distribution. It is not too surprising that standard NMPC fails to stay feasible in all three cases. Min-max and back-off formulations, as two other robust NMPC methods, show varying degrees of conservatism with respect to the uncertain parameter probability distribution. Multistage NMPC formulation still achieves the realistically minimal tracking error - closest to the error in perfect information case.

After establishing the argument that ideal-multistage NMPC is a non-conservative robust NMPC scheme, we now conduct comparison studies of ideal-multistage and advanced-step multistage NMPC.

4.3.1.2 Ideal multistage and advanced-step multistage

Type 1 uncertainty only

We first consider model parameter uncertainty without unmeasured disturbances. Two case studies have been conducted with the model parameter being c_{A0} or $E_{A,3}$, where the range for c_{A0} is $\pm 30\%$ and it is $\pm 5\%$ for $E_{A,3}$. Figs. 4.6 and 4.7 compare the time trajectories for state variables c_A, c_B and control variable F for both multistage NMPCs. From Figs. 4.6 and 4.7 and we see that both as-msNMPC and ideal-msNMPC satisfy state and control variable constraints. Moreover, state and control profiles for as-msNMPC and ideal-msNMPC are reasonably close to each other, and both approaches manage to reach the setpoint for c_B. Note that the oscillation does not die out due to the fact that both uncertainties vary at each time step.

A comparison of the averaged accumulated cost of four robust schemes with respect to two different uncertain parameters is presented in Table 4.6. Note that all four robust NMPC schemes ensure constraint satisfaction under uncertainty. The accumulated cost in the table is calculated by averaging the current stage cost (i.e. $\sum_{k=1}^{120} \varphi(x_k, u_k.d_k)$) in 10 random runs. As expected, the perfect information scheme achieves the lowest tracking error for both uncertainties, although it cannot be implemented in practice. By using multistage formulations, both ideal-ms and as-ms obtain similar performance and both of them have significantly smaller error compared to the worst case. As expected, the worst case ap-

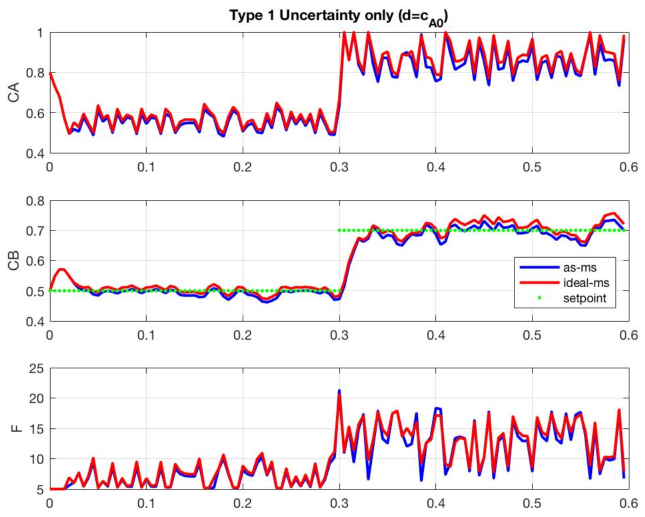


Figure 4.6: Comparison of as-msNMPC with ideal-msNMPC with Type 1 uncertainty for $d = c_{A0} \pm 30\%$.

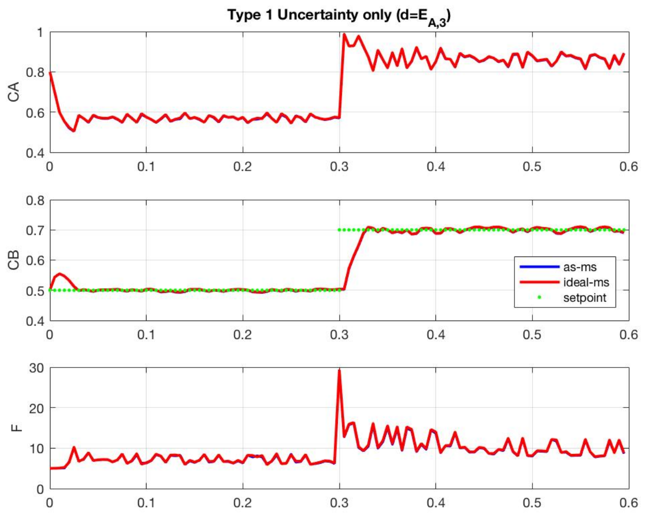


Figure 4.7: Comparison of as-msNMPC with ideal-msNMPC with Type 1 uncertainty for $d = E_{A,3} \pm 5\%$.

proach achieves robustness with the highest tracking errors among all methods. Table 4.6 demonstrates that using ideal-ms or as-ms approach reduces the controller conservatism while maintaining solution robustness.

Tracking costs for uncertain parameter(s)	$\mathbf{c_{A0}}$	$\mathbf{E_{A,3}}$
ideal-ms	0.1387	0.0749
as-ms	0.1402	0.0759
perfect info	0.0669	0.0599
worst case	0.2560	0.2009

Table 4.6: Accumulated stage costs of robust NMPC schemes with robust horizon $N_r = 2$, $d = [c_{A0} \pm 30\%, E_{A,3} \pm 5\%]$.

In addition, both uncertain parameters c_{A0} and $E_{A,3}$ are considered simultaneously in the model to demonstrate a real-life example. With each d_k containing three possible realizations, the number of branches per node is 9. As a result, an 81-scenario tree is generated for $N_r = 2$. Fig. 4.8 shows consistent trajectories for state and control variables for ideal-msNMPC and as-msNMPC, even with the substantial increase in tree size.

	Costs	**CPUs (parallel (est.))**
ideal-ms	0.1490	$17.8s$ $(17.8s)$
as-ms	0.1667	$9.8s$ $(1.1s)$

Table 4.7: Tracking and computational performance of ideal-msNMPC and as-msNMPC for $d = (c_{A0}, E_{A,3})$

A quantitative comparison between ideal-msNMPC and as-msNMPC for two uncertain parameters can be found in Table 4.7. The averaged tracking error column shows that the

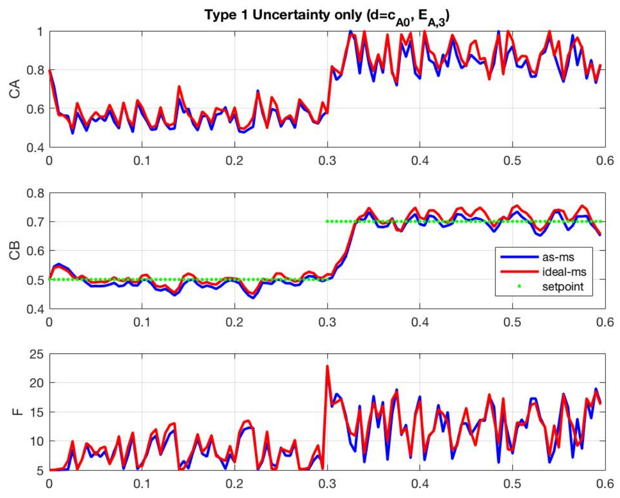


Figure 4.8: Comparison of as-msNMPC with ideal-msNMPC with Type 1 uncertainty for $d = (c_{A0}, E_{A,3})$.

ideal-ms and as-ms have similar performance. However, as-msNMPC requires 9.8 CPUs in total off-line computation, and only 1.1 CPUs when solved in parallel. On the other hand, ideal msNMPC requires even longer off-line computation time (17.8 s). Therefore, parallel as-msNMPC is able to provide background solutions within a much shorter sampling time, and the online step merely selects the corresponding solution. In comparison, ideal-msNMPC solves the entire problem online which requires significant online CPU time and delays the availablity of control actions.

Type 1 and 2 uncertainty

We now introduce additional noise for x_k and compare the as-msNMPC and ideal-msNMPC approaches. Since both uncertain model parameters and unmeasured noise are present, background solutions must be adjusted using sensitivity updates.

To simulate the noise w_k, white noise is introduced to the system with standard deviation σ as a fraction of all states x_k. In Fig. 4.9, we can see as-msNMPC performs very closely to ideal-msNMPC at low noise levels (2.5%). With increasing levels of noise, behavior of as-msNMPC is similar to ideal-msNMPC, but with degrading resemblance. In Fig. 4.10, we observe similar trends for $d = E_{A,3}$, where as-msNMPC and ideal-msNMPC behaviors are almost identical.

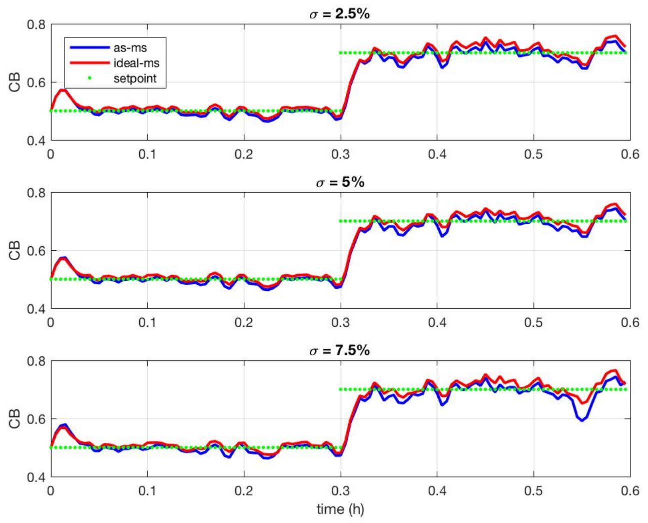


Figure 4.9: Comparison of as-msNMPC with ideal-msNMPC with respect to different levels of noise for $d = c_{A0}$.

Table 4.8 compares computational effort between as-msNMPC and ideal-msNMPC for Type 1 and 2 uncertainty by averaging 10 random runs. Here, as-msNMPC reduces online computational effort by over two orders of magnitude compared to ideal-msNMPC, where the entire computation takes place online. With further parallelization of the background solutions in as-msNMPC, total (offline + online) computational time also decreases significantly. Moreover, the as-msNMPC framework has the potential for further reduction on

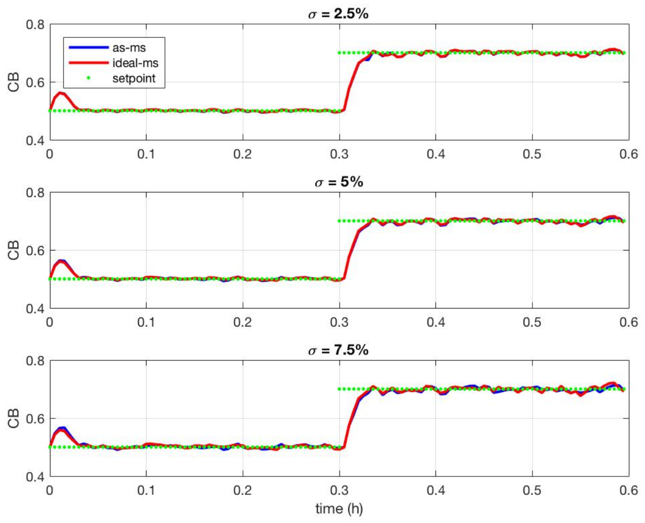


Figure 4.10: Comparison of as-msNMPC with ideal-msNMPC with respect to different levels of noise for $d = E_{A,3}$

large-scale problems with high-dimensional uncertainties in the noisy case as well.

CPU seconds	**as-ms**	**ideal-ms**
Online	$0.006s$	$0.954s$
Background+Online	$0.960s$	$0.954s$
Parallel Background (est.)+Online	$0.320s$	$0.954s$

Table 4.8: Comparison of averaged online computational effort per instance, for $d = c_{A0}$ and $N_r = 2$

4.3.1.3 Feasibility study

Finally, we study the robustness of different NMPC control strategies with both Type 1 and 2 uncertainties, especially in terms of constraint satisfaction. One can often trade off

	$\lvert g_{c_A^{up}}\rvert$	$\lvert g_{c_B^{up}}\rvert$	$\lvert g_{T_R^{up}}\rvert$	**max obj**	**avg obj**	**min obj**
Standard	1.5576	0.3376	0.1192	8.1711	1.7066	0.1297
Min-max	0	0	0	1.2598	1.1019	0.8254
Ideal-ms	0	0	0	0.3019	0.2416	0.1716
As-ms	0	0	0	0.4518	0.3660	0.2622

Table 4.9: Normalized Bound constraint violation (represented by $g_{constraint\_name}$) and cost function values by different NMPC schemes with Type 1 uncertainty for $d = E_{A,3} \pm 10\%$.

	$\lvert g_{c_A^{up}}\rvert$	$\lvert g_{c_B^{up}}\rvert$	$\lvert g_{T_R^{up}}\rvert$	**max obj**	**avg obj**	**min obj**
Standard	1.5518	0.3591	0.1177	3.7450	1.5723	0.1274
Min-max	0	0	0	1.3236	1.0929	0.8075
Ideal-ms	0.0026	0	0	0.3248	0.2456	0.1855
As-ms	0.0027	0	0	0.4625	0.3627	0.2696

Table 4.10: Normalized Bound constraints violation and cost function values by different NMPC schemes with Type 1+2 uncertainty for $d = E_{A,3} \pm 10\%$ and $\sigma = 5\%$.

between robustness and performance of controllers because conversatism usually comes along with robust controllers, and it will inherently hurt control performance. Simultaneously achieving both is often difficult, if not impossible. Hence, the performances (both robustness and tracking objectives) of ideal-msNMPC and as-msNMPC are compared against two baseline NMPCs: standard NMPC and min-max NMPC.

Table 4.9 illustrates the constraint violations as well as corresponding objective values for each NMPC controller when only Type 1 uncertainty is present. The first three columns represent the normalized violation value for its corresponding constraint. For instance, the normalized infeasibility for the upper bound of c_A is determined by $|g_{c_A^{up}}| = \max|\frac{c_A^{actual} - c_A^{up}}{c_A^{up}}|$ where c_A^{actual} is the actual state value in the computed trajectory and c_A^{up} is the upper bound for that state. Note that infeasibility only considers the state and control variables which are implemented in the plant, i.e. only the first step. In addition, only the constraints that are active are listed in the table. The next three columns represent the maximum, average and minimum objective values among all the sample runs. Tables 4.9 and 4.10 are obtained by running 10 random simulations.

As expected, with Type 1 uncertainty alone, standard NMPC has the most infeasibility both in number of constraints and in the magnitude of violation. Furthermore, standard NMPC also has a much higher cost compared with other NMPCs. Among three robust NMPC schemes, ideal-ms and as-msNMPC manage to stay lower on objective values than min-max strategy, and all three robust NMPC schemes are able to satisfy all constraints.

Similar trends are observed in Table 4.10, where both Type 1 and 2 uncertainty are present. Standard NMPC is still the most costly NMPC controller in terms of constraint violation and objective function. Min-max NMPC is able to stay feasible, but with a loss in the tracking performance. Because of the added noise, ideal-ms and as-ms now observe slight infeasibility on c_A upper bound, and the tracking objectives of both are only slightly affected by the additive noise.

4.3.2 Quadtank example

Consider the quadtank problem [3], with dynamics described as follows:

$$\frac{dx_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gx_1} + \frac{a_3}{A_1}\sqrt{2gx_3} + \frac{\gamma_1}{A_1}u_1 \tag{4.17a}$$

$$\frac{dx_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gx_2} + \frac{a_4}{A_2}\sqrt{2gx_4} + \frac{\gamma_2}{A_2}u_2 \tag{4.17b}$$

$$\frac{dx_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gx_3} + \frac{(1-\gamma_2)}{A_3}u_2 \tag{4.17c}$$

$$\frac{dx_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gx_4} + \frac{(1-\gamma_1)}{A_4}u_1 \tag{4.17d}$$

where x_i denotes the water level of tank i, u_i represents the flow rate of pump i, A_i is the cross-sectional area of tank i, a_i is the cross-sectional area of outlet port of tank i, and $\gamma_i \in (0,1)$ are the valve parameters. The two uncertain parameters in the model are γ_1, γ_2.

Table 4.11: Model parameter values, adapted from [3]

	$A_i(cm^2)$	$a_i(cm^2)$
$i=1$	50.27	0.233
$i=2$	50.27	0.242
$i=3$	28.27	0.127
$i=4$	28.27	0.127

The goal is to track the water levels in both lower tanks (tank 1 and 2 in Fig. 4.11) to be $x_{1s} = x_{2s} = 14cm$. Assuming the valve parameters are nominal, $\gamma_1 = \gamma_2 = 0.4$, then the corresponding setpoints are $x_s = [14cm, 14cm, 14.2cm, 21.3cm]^T$ and $u_s = [43.4mL/s, 35.4mL/s]^T$.

The stage cost is computed by:

$$\varphi_l = (x_{1l} - x_{1s})^2 + (x_{2l} - x_{2s})^2 + 0.01(\Delta u_{1l}^2 + \Delta u_{2l}^2) \tag{4.18}$$

where $\Delta u_{1l} = u_{1l} - u_{1l-1}$, and $\Delta u_{2l} = u_{2l} - u_{2l-1}$.

Due to the characteristics of the pump and the range of water levels of each tank, the following input and state constraints are applied:

$$u_{min} = [0mL/s, 0mL/s]^T.$$

$$u_{max} = [60mL/s, 60mL/s]^T.$$

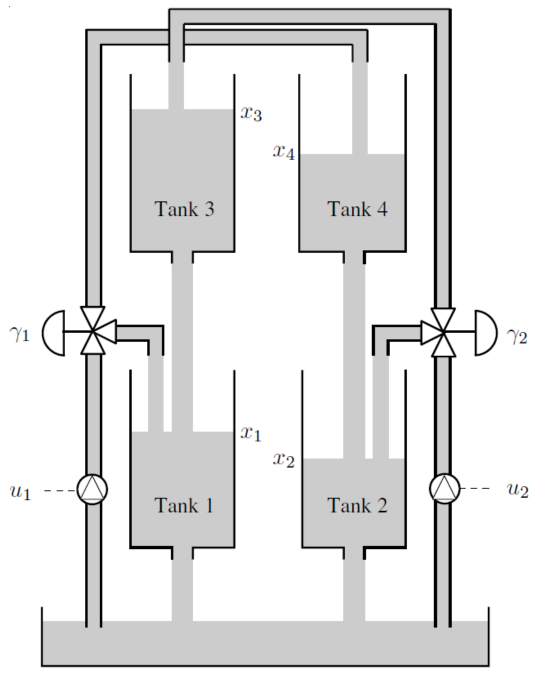


Figure 4.11: Quad Tank [3]

$$x_{min} = [7.5mL/s, 7.5mL/s, 3.5mL/s, 4.5mL/s]^T.$$

$$x_{max} = [28mL/s, 28mL/s, 28mL/s, 28mL/s]^T.$$

We also introduce pulses of predefined state values at time interval k to reinitialize controller tracking. See Table 4.12 for the predefined values.

Table 4.12: Predefined state values

k	x_1	x_2	x_3	x_4
0	x_{1max}	x_{2max}	x_{3s}	x_{4s}
50	x_{1max}	x_{2s}	x_{3max}	x_{4s}
100	x_{1max}	x_{2s}	x_{3s}	x_{4max}

4.3.2.1 Type 1 uncertainty only

First, we compare the performance of ideal-ms and as-ms NMPC without unmeasured noise. The uncertain parameters are γ_1 and γ_2, for both of which, the max, nominal and min values are [0.5, 0.4, 0.3].

From Figs. 4.12 - 4.14, we observe that with only Type 1 uncertainty, the performances of as-msNMPC and ideal-msNMPC are very close. For a quantitative comparison, Table 4.13 presents accumulated stage costs averaging for 10 random runs and $N_r = 2$. The values are mainly dominated by the scale of state variables at transition, but the difference between ideal-ms and as-ms is insignificant. It is also observed that as-msNMPC may sometimes outperform ideal-msNMPC, which is plausible.

Tracking costs for uncertain parameter(s)	γ_1	γ_2	(γ_1, γ_2)
Ideal-ms	267.65	274.69	309.67
As-ms	269.72	239.92	298.00
Ideal-ms ($\sigma = 5\%$)	267.68	274.67	309.54
As-ms ($\sigma = 5\%$)	269.67	239.82	298.49

Table 4.13: Accumulated stage costs of ideal-msNMPC and as-msNMPC with only Type 1 uncertainty (top two rows) and with both Type 1 and 2 uncertainty (bottom two rows), with $N_r = 2$

4.3.2.2 Type 1 and 2 uncertainty

We now introduce additional noise in the system, and NLP sensitivity is used to update the actual controls. As in the CSTR example, σ for the measurement noise is defined as a fraction of the state variables. The bottom rows of Table 4.13 compare the tracking per-

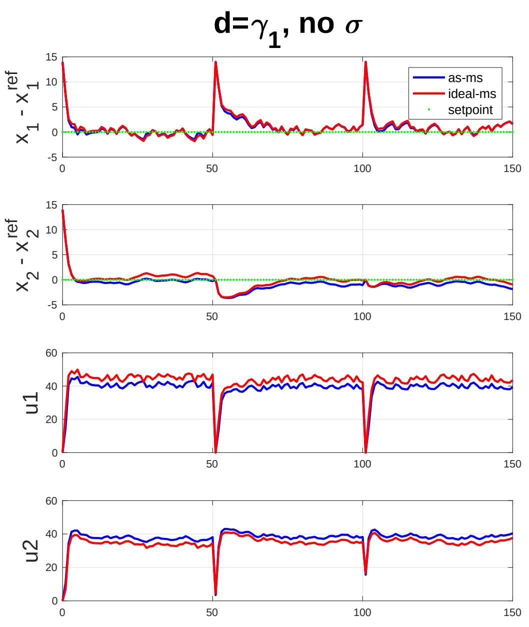


Figure 4.12: States and controls trajectories of as-msNMPC and ideal-msNMPC for Type 1 uncertainty $d = \gamma_1$

formance of as-msNMPC and ideal-msNMPC when the level of noise is 5%, and they preserve the trend for the Type 1 only case.

To compare the online and total computational resources needed for as-ms and ideal-ms NMPC, Table 4.14 shows that as-msNMPC partitions the majority of the workload in the background, and the online computation is two orders of magnitude less than the total computation, which is consistent with our findings from the first case study.

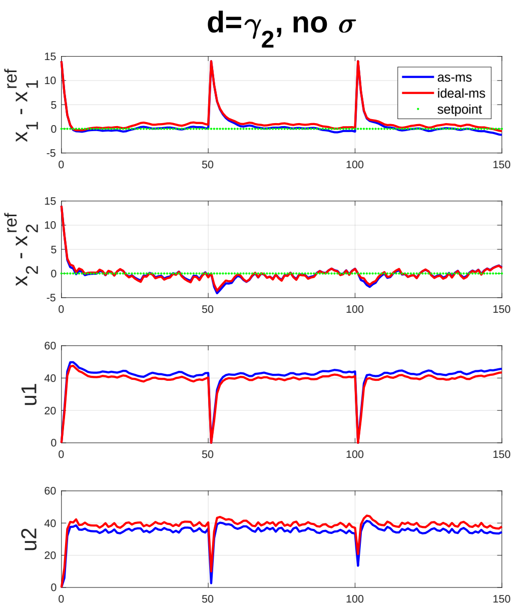


Figure 4.13: States and controls trajectories of as-msNMPC and ideal-msNMPC for Type 1 uncertainty $d = \gamma_2$

CPU seconds	**as-ms**	**ideal-ms**
Online	$0.002s$	$0.205s$
Background+Online	$0.181s$	$0.205s$
Parallel Background (est.)+Online	$0.062s$	$0.205s$

Table 4.14: Comparison of Averaged Online Computational Effort for $d = \gamma_1, N_r = 2$

4.4 Conclusions and remarks

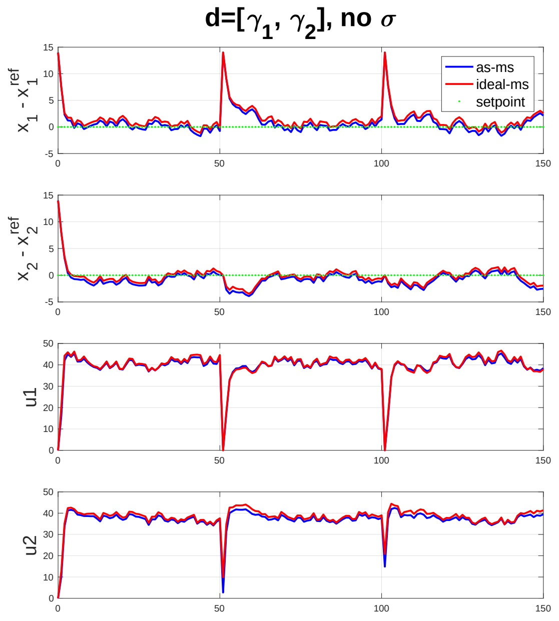


Figure 4.14: States and controls trajectories of as-msNMPC and ideal-msNMPC for Type 1 uncertainty $d = [\gamma_1, \gamma_2]$

In this chapter, we have proposed an advanced-step multistage NMPC framework that is robust to two types of uncertainties and virtually free of computational delay. This framework separates NMPC computations into two parts: the background solution carries most of the computational endeavors by solving one step ahead, and the sensitivity-based correction updates the control injected into the plant. In addition, in the presence of unmeasured noise, as-msNMPC obtains an approximate solution to ideal-msNMPC by updating the background predicted solution with NLP sensitivity information.

To demonstrate the advantages of advanced-step multistage NMPC over standard NMPC and conventional robust NMPC methods, CSTR and quadtank examples have been investigated. In the case of CSTR, ideal-multistage NMPC outperforms standard NMPC, min-max NMPC and back-off approaches in terms of low conservatism and constraint satisfaction. Additionally, advanced-step multistage NMPC emulates ideal-multistage NMPC and extends its robustness with respect to two types of uncertainties. Similar observations can be drawn with the quadtank case study.

We have shown that advanced-step multistage NMPC directly addresses the first two challenges of robust NMPC: control performance and real-time implementation. Stability property is covered by the next chapter, in which a detailed analysis of ideal multistage NMPC and advanced-step multistage NMPC is performed.

Chapter 5

Stability analysis for NMPC and its variants

Earlier in Chapter 2, Lyapunov stability theory has been introduced as basic tools for stability analysis. In this chapter, we are designing robust NMPC algorithms such that the closed-loop stability can be guaranteed by solving open-loop optimal control problem. We will start by analyzing the stability of standard NMPC, where the model is perfect and there is no plant-model mismatch. Then we move on to the cases with two separate types of uncertainty and their corresponding robust NMPC strategies. To be consistent with Chapter 4, the same definition of Type 1 and 2 uncertainty is used throughout this chapter. When only Type 2 uncertainty is present, robust stability of ideal and advanced-step standard NMPC is established based on ISS. When Type 1 uncertainty also enters the system, one needs to rely on ideal and advanced-step multistage NMPC scheme. Under suitable assumptions, both ideal and advanced-step multistage NMPC have demonstrated input-to-state practical stability properties with the presence of two types of uncertainty.

To recap, we consider the discretized system dynamics following this form in this chapter:

$$x_{k+1} = f(x_k, u_k, d_k) + w_k \tag{5.1}$$

where $x_k \in \mathbb{X} \subset \mathbb{R}^{n_x}, u_k \in \mathbb{U} \subset \mathbb{R}^{n_u}$ represent the state and control variable at time k, and $d_k \in \mathbb{D} \subset \mathbb{R}^{n_d}$, $w_k \in \mathbb{W} \subset \mathbb{R}^{n_x}$ denote the parametric disturbances and bounded additive noise (i.e. Type 1 and 2 uncertainty) at time k.

5.1 Ideal standard NMPC

A nominal NMPC, also called standard NMPC, solves the following finite horizon NLP problem of prediction horizon N at each step k:

$$J_N(x_k) = \min_{z_l, v_l} \quad \phi(z_N) + \sum_{l=0}^{N-1} \varphi(z_l, v_l) \tag{$\mathcal{P}_N^{id}(x_k)$}$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l) \qquad l = 0, ..., N-1 \tag{5.2a}$$

$$z_0 = x_k \tag{5.2b}$$

$$z_l \in \mathbb{X}, v_l \in \mathbb{U}, z_N \in \mathbb{X}_f \tag{5.2c}$$

where $\varphi(\cdot), \phi(\cdot)$ represent the stage cost and terminal cost, respectively. $\mathbb{X}$ and $\mathbb{U}$ are domains for state and control variables, and $\mathbb{X}_f$ is the terminal region.

To recall the implementation of rolling horizons in MPC, at each step k, (5.2) is solved to obtain a sequence of optimal control actions, of which the first step is implemented to the plant as $u_k = h(x_k)$. In particular, $h(\cdot) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ is also referred to as the implicit feedback control law generated by MPC. Then the system advances one step and the process repeats.

The term *"ideal"* here specifically refers to the case where the feedback control law can be computed and implemented instantaneously without any delays. This may not be true for many applications with nonlinear dynamics. Nevertheless, stability analysis of ideal NMPC paves the road for the design and analysis of other NMPCs.

For standard NMPC, uncertainties are not considered in the model as shown in (5.2). In the rest of this section, we first show the nominal stability for standard NMPC when the model is perfect and there is no plant-model mismatch, and then prove the robust stability of ISS with the presence of uncertainty.

5.1.1 Nominal stability

To prove the asymptotic stability, one needs to find a Lyapunov function that satisfies Definition 7. For MPC applications, a natural choice of Lyapunov function is the value function acting as the objective function in (5.2). Suppose Assumption 4 (system assumption) holds, addition assumptions are made to the model:

Assumption 26. *(Nominal stability assumptions of ideal NMPC)*

- *There exists a local control law $u = h_f(x)$ defined on $\mathbb{X}_f$ such that $f(x, h_f(x)) \in \mathbb{X}_f, \forall x \in \mathbb{X}_f$ and $\phi(f(x, h_f(x))) - \phi(x) \leq -\varphi(x, h_f(x))$.*
- *$\varphi(x, u)$ satisfies $\alpha_p(|x|) \leq \varphi(x, u) \leq \alpha_q(|x|)$ where $\alpha_p(\cdot)$ and $\alpha_q(\cdot)$ are $\mathcal{K}$ functions.*
- *$\phi(x)$ satisfies $\alpha'_p(|x|) \leq \phi(x) \leq \alpha'_q(|x|)$ where $\alpha'_p(\cdot)$ and $\alpha'_q(\cdot)$ are $\mathcal{K}$ functions.*

Theorem 27. *(Nominal stability of ideal NMPC) [72] Suppose Assumption 26 holds for* (5.2) *and $u = h(x)$ is the feedback control law computed by* (5.2)*, then $J_N(x)$ is a Lyapunov function and the closed-loop system is asymptotically stable.*

Proof. 1. The lower bound (2.4a) can be shown, which follows from Assumption 26.

$$J_N(x_k) \geq \varphi(x_k, h^{id}(x_k)) \geq \alpha_1(|x_k|) \tag{5.3}$$

2. Since having a terminal constraint $\mathbb{X}_f$ guarantees the recursive feasibility of the solutions [28, 41], if the optimal control sequence of $\mathcal{P}^{id}_N(x_k)$ is $\{v_0, v_1, ..., v_{N-1}\}$, then $\{v_0, v_1, ..., v_{N-1}, u_f\}$ is a feasible solution for $\mathcal{P}^{id}_{N+1}(x_k)$. Using this admissible control sequence, we extend the prediction horizon by one and show that,

$$\begin{aligned} \bar{J}_{N+1}(x_k) &= J_N(x_k) - \phi(z_N) + \phi\big(f(z_N, h_f(z_N))\big) + \varphi(z_N, h_f(z_N)) \\ &\leq J_N(x_k) \end{aligned} \tag{5.4}$$

It is easy to see that if using the optimal control sequence (instead of an admissible one), $J_{N+1}(x_k) \leq \bar{J}_{N+1}(x_k) \leq J_N(x_k)$. By deduction,

$$J_{N+1}(x_k) \leq J_N(x_k) \leq \cdots \leq \phi(x_k) \leq \alpha_2(|x_k|) \tag{5.5}$$

3. Since the model is perfect, the actual state appears exactly as the prediction, i.e. $x_{k+1} = f(x_k, u_k)$. The descent property of Lyapunov function is shown,

$$\begin{aligned} J_N(x_{k+1}) - J_N(x_k) &= J_N(x_{k+1}) - J_{N+1}(x_k) + J_{N+1}(x_k) - J_N(x_k) \\ &\leq J_N(x_{k+1}) - J_{N+1}(x_k) \\ &\leq -\varphi(x_k, h^{id}(x_k)) \\ &\leq -\alpha_3(|x_k|) \end{aligned} \tag{5.6}$$

where the first inequality holds because of (5.4), and the second inequality stands because of recursive feasibility (solution of $\mathcal{P}_{N+1}(x_k)$ is also feasible for $\mathcal{P}_N(x_{k+1})$). This proves that ideal NMPC is asymptotically stable. □

5.1.2 Robust stability

Now we study the robust stability property of ideal NMPC under the presence of uncertainties and disturbances. The inherent robustness associated with nominal NMPC exists under the assumption that the presence of uncertainties do not result in the loss of feasibility in the problem formulation [73]. This can be achieved by reformulating the inequality or bound constraints as soft constraints and penalizing the constraint violations in the objective. For additional details, the readers are referred to [74, 75].

In this section, we only consider *Type 2 uncertainty* (i.e. w) in the system,

$$x_{k+1} = f(x_k, u_k) + w_k \tag{5.7}$$

where ideal NMPC predicts the future states as $\hat{x}_{k+1} = f(x_k, u_k)$ and $u_k = h^{id}(x_k)$ is the feedback control law generated by ideal NMPC.

We define the mismatch term for the value function as,

$$\epsilon(x_{k+1}) = J_N(x_{k+1}) - J_N(\hat{x}_{k+1}) \tag{5.8}$$

Assumption 28. *(Robust stability assumptions of ideal NMPC)*

- *The value function $J_N^{id}(x)$ is Lipschitz continuous with a Lipschitz constant $L_J > 0$ (see Definition 1)*

Theorem 29. *(Robust stability of ideal NMPC) [72] Suppose Assumption 26 holds for* (5.2) *and $u = h^{id}(x)$ is the feedback control law computed by* (5.2)*, then $J_N(x)$ is an ISS-Lyapunov function and the closed-loop system is ISS-stable.*

Proof. Since the first two parts of the proof to Theorem 27 are still valid, we focus on proving the descent property of neighboring cost functions. Notice that with Type 2 uncertainty, we denote the actual future state as x_{k+1} and next step state predicted by ideal NMPC as $\hat{x}_{k+1}$, we have

$$\begin{aligned}
J_N(x_{k+1}) - J_N(x_k) &= J_N(x_{k+1}) - J_N(\hat{x}_{k+1}) + J_N(\hat{x}_{k+1}) - J_N(x_k) \\
&= \epsilon(x_{k+1}) + J_N(\hat{x}_{k+1}) - J_N(x_k) \\
&\leq \epsilon(x_{k+1}) - \varphi(x_k, h^{id}(x_k)) \\
&\leq L_J|x_{k+1} - \hat{x}_{k+1}| - \varphi(x_k, h^{id}(x_k)) \\
&\leq L_J|w_k| - \varphi(x_k, h^{id}(x_k)) \\
&\leq -\alpha_4(|x_k|) + \sigma_1(|w_k|)
\end{aligned} \tag{5.9}$$

The first inequality holds because of Assumption 26 and recursive feasibility (similar as (5.6)), and the rest follows from Assumption 28.

Unlike the nominal stability case where asymptotic stability of the origin is guaranteed, ISS implies that the system evolves to a compact set around the origin, in which the size of the compact set depends on $|w|$. Note that in the nominal case, robust stability defaults to nominal stability with $M = 0$. The readers are referred to [64, 73, 76] for details. $\square$

5.2 Advanced-step standard NMPC

Although ideal NMPC poses nice stability properties, online implementation still remains a challenge: it requires a solution to the optimal control problem almost immediately after new states are measured. This is especially computationally intensive for large-scale nonlinear applications. This delayed control action could deteriorate performance and may also cause instability of the closed-loop [19, 77, 78].

In order to reduce the online computational footprint of solving nonlinear MPC, advanced-step NMPC (*asNMPC*) was introduced [64]. The essence of asNMPC is to separate the computational workload into two parts: *background* and *online*, and ultimately have negligible computation delays *online*. The background problem solves the future optimal control problem in advance using the presumptive initial states from model prediction, whereas the online problem leaves with corrections with respect to the actual plant states. This distinction between background and online computations becomes beneficial because the online correction is much faster than solving the background problem [64]. This controller design allows one sampling time to be used as the computational resource to solve the background problem, which is assumed to be sufficient within the scope of this thesis. A similar assumption is made such that the online update can be obtained within a negligible amount of time.

AsNMPC assumes that at t_k, we compute the predicted states and controls one step ahead for t_{k+1} using the known dynamics. We denote the predicted initial states as $x_{k+1|k}$. When the actual states x_{k+1} are observed at t_{k+1}, control variables are correspondingly updated using NLP sensitivity. The following problem is solved at t_k for asNMPC:

$$J_N^{ad}(x_k) = \min_{z_l, v_l} \quad \phi(z_N) + \varphi(x_k, h^{ad}(x_k)) + \sum_{l=0}^{N-1} \varphi(z_l, v_l) \qquad (\mathcal{P}_N^{ad}(x_k))$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l) \qquad l = 0, ..., N-1 \tag{5.10a}$$

$$z_0 = f(x_k, h^{ad}(x_k)) \tag{5.10b}$$

$$z_l \in \mathbb{X}, v_l \in \mathbb{U}, z_N \in \mathbb{X}_f \tag{5.10c}$$

The predicted states and controls are obtained as $\{z_0^*, ..., z_N^*, v_0^*, ..., v_{N-1}^*\}$, where v_0^* is the predicted control. At t_{k+1}, the actual state is obtained using (5.1), u_{k+1} is updated using an NLP sensitivity correction based on v_0^* and the difference between the prediction and realization (i.e. $x_{k+1} - x_{k+1|k}$); the approximate feedback law is denoted as $h^{ad}(\cdot)$. The concept of NLP sensitivity has been discussed in Chapter 3.2.3. As discussed in [61], the inequality (bound) constraints for states in (5.10c) are also softened as l_1 penalty terms in order to guarantee robustness.

Note that $\mathcal{P}_N^{ad}(x_k)$ is equivalent to the extended horizon problem for the ideal case with precomputed advanced-step control law $\mathcal{P}_{N+1}^{id}(x_k, h^{ad}(x_k))$ [64].

In this section, we again only consider *Type 2 uncertainty* (i.e. w) in the system,

$$x_{k+1} = f(x_k, u_k) + w_k \tag{5.11}$$

At t_k, the predicted states for the future step are $x_{k+1|k} = f(x_k, h^{ad}(x_k))$ that can be computed by solving $\mathcal{P}_N^{ad}(x_k)$, where $u_k = h^{ad}(x_k)$ is the *approximate* feedback control computed at the previous step (i.e. the optimal solution of $\mathcal{P}_N^{ad}(x_{k-1})$) with sensitivity updates when x_k is realized at t_k.

Advanced-step NMPC also carries the same nominal and robust stability properties of standard NMPC. In the nominal case where $w = 0$, then $x_{k+1} = x_{k+1|k}$. Thus the feedback controls computed by standard NMPC and advanced-step NMPC are identical as long as the initial feedback control u_0 is consistent between ideal and advanced-step NMPC. In this way the nominal stability property also follows for advanced-step NMPC.

To facilitate the discussion, we define two mismatch terms, which represent the error introduced by Type 2 uncertainty and by NLP sensitivity, respectively.

$$\epsilon_{ad}(x_{k+1}) \quad := \quad J_N^{ad}(x_{k+1}) - J_{N+1}(x_{k+1}) \tag{5.12a}$$

$$\epsilon_s(x_{k+1}) \quad := \quad J_{N+1}(x_{k+1}) - J_{N+1}(x_{k+1|k}) \tag{5.12b}$$

Despite the different subscript, the value functions $J_N^{ad}(\cdot)$ and $J_{N+1}^{id}(\cdot)$ actually refers to the same horizon length because of the equivalence between $\mathcal{P}_N^{ad}(x_k)$ and $\mathcal{P}_{N+1}^{id}(x_k, h^{ad}(x_k))$. In particular, $J_N^{ad}(x_{k+1}) := J_{N+1}(x_{k+1}, h^{ad}(x_{k+1}))$. Then we can compare these two terms directly: $J_N^{ad}(x_{k+1})$ is a suboptimal cost function with the advanced-step control, whereas the optimal cost function is $J_{N+1}^{id}(x_{k+1}, h^{id}(x_{k+1}))$. In other words, (5.12a) can be written as:

$$\begin{aligned} \epsilon_{ad}(x_{k+1}) &= J_N^{ad}(x_{k+1}) - J_{N+1}(x_{k+1}) \\ &= J_{N+1}(x_{k+1}, h^{ad}(x_{k+1})) - J_{N+1}(x_{k+1}) \\ &= J(x_{k+1}, h^{ad}(x_{k+1})) - J(x_{k+1}, h^{id}(x_{k+1})) \end{aligned} \tag{5.13}$$

(5.12b) can also be written as:

$$\begin{aligned} \epsilon_s(x_{k+1}) &= J(x_{k+1}) - J(x_{k+1|k}) \\ &= J(x_{k+1}, h^{id}(x_{k+1})) - J(x_{k+1|k}, h^{id}(x_{k+1|k})) \end{aligned} \tag{5.14}$$

For the convenience of reading, the subscript is now omitted for the rest of *this* subsection. Additionally, $J(\cdot)$ refers to $J^{id}(\cdot)$ unless otherwise specified.

Theorem 30. *From Theorem 25 and* (3.24)*, with* $p_0 = x_{k+1|k}$ *and* $p = x_{k+1} = x_{k+1|k} + w_k$*, there exists a local positive Lipschitz constant* L_s^{ad} *such that the approximation error of control laws between ideal NMPC and advanced-step NMPC satisfies* $|h^{id}(x_{k+1}) - h^{ad}(x_{k+1})| \leq L_s^{ad}|w_k|^2$.

Proof of Theorem 30 can be found at [64].

Assumption 31. *(Robust stability assumption of advanced-step NMPC)*

- *The value function* $J_N^{ad}(x)$ *is Lipschitz continuous with a Lipschitz constant* $L_J > 0$

Theorem 32. *(Robust stability of advanced-step NMPC) Under Assumptions, the cost function* $J_N^{ad}(x)$ *obtained from the solution of* $\mathcal{P}_N^{ad}(x, u)$ *with* $u = h^{ad}(x)$ *is an ISS-Lyapunov function under Type 2 uncertainty, thus the resulting closed-loop system is ISS stable.*

Proof. Under Theorems 25 and 46 and Assumption 31, there exist positive Lipschitz constants L_J, L_h, L_s^{ad} such that

$$\begin{aligned}
\epsilon_{ad}(x_{k+1}) &\leq L_J(|x_{k+1} - x_{k+1}| + |h^{id}(x_{k+1}) - h^{ad}(x_{k+1})|) \\
&\leq L_J L_s^{ad}|w_k|^2 \\
\epsilon_s(x_{k+1}) &\leq L_J(|x_{k+1} - x_{k+1|k}| + |h^{id}(x_{k+1}) - h^{id}(x_{k+1|k})|) \\
&\leq L_J(1 + L_h)|w_k|
\end{aligned}$$

In particular, the last inequality stands from $|h^{id}(x_{k+1}) - h^{id}(x_{k+1|k})| \leq L_h|w_k|$.

By comparing the cost functions of neighboring problems $J_N^{ad}(x_{k+1})$ and $J_N^{ad}(x_k)$ using defined error terms,

$$\begin{aligned}
J_N^{ad}(x_{k+1}) - J_N^{ad}(x_k) = \;& J(x_{k+1}, h^{ad}(x_{k+1})) - J(x_k, h^{ad}(x_k)) \\
= \;& J(x_{k+1}, h^{ad}(x_{k+1})) - J(x_{k+1}) + J(x_{k+1}) - J(x_{k+1|k}) \\
& + J(x_{k+1|k}) - J(x_k, h^{ad}(x_k)) \\
= \;& \epsilon_{ad}(x_{k+1}) + \epsilon_s(x_{k+1}) + J(x_{k+1|k}) - J(x_k, h^{ad}(x_k)) \qquad (5.16) \\
\leq \;& \epsilon_{ad}(x_{k+1}) + \epsilon_s(x_{k+1}) - \varphi(x_k, h^{ad}(x_k)) \\
\leq \;& L_J L_s^{ad} |w_k|^2 + L_J(1 + L_h)|w_k| - \varphi(x_k, h^{ad}(x_k)) \\
\leq \;& -\alpha_5(|x_k|) + \sigma_2(|w_k|)
\end{aligned}$$

The first inequality holds because of recursive feasibility in which $\mathcal{P}_{N+1}(x_k, h^{ad}(x_k))$ provides a feasible solution of $\mathcal{P}_{N+1}(x_{k+1|k})$, and the last two inequalities follow from $M \geq L_J(1 + L_h + L_s^{ad}|w_k|) > 0$. Then the value function $J_N^{ad}(x_k)$ is an ISS-Lyapunov function, and the resulting system is ISS-stable. □

5.3 Ideal multistage NMPC

To develop stability properties for both ideal and advanced-step multistage NMPC, similar studies [2, 79] have looked into the recursive feasibility and stability properties of ideal multistage NMPC. In this section, these concepts are extended to Type 1, 2 input-to-state practical stability (ISpS) for both ideal-msNMPC and as-msNMPC. Similar to [79], the stability results are only applied to *fully-expanded* multistage formulation where $N_r = N$. These limitations are stated below in Assumption 33.

We introduce a new set of concepts and assumptions of stability properties with Type 1 and 2 stability, then discuss ISpS attributes of ideal-multistage NMPC, and we also develop ISpS properties of advanced-step multistage NMPC.

5.3.1 ISpS with Type 1 uncertainty

Recursive feasibility of multistage NMPC is equivalent to terminal states of each scenario remaining in a control invariant terminal set at the end of the prediction horizon [79].

Assumption 33. *(Recursive feasibility assumptions of fully-expanded ideal-msNMPC ($N_r = N$), adapted from [79])*

- *(Lipschitz continuity) The model dynamics and both stage cost and terminal cost are twice differentiable in x and u and Lipschitz continuous in all arguments, and they satisfy* $\forall j \in \mathbb{Q}^{n_d},\ f(0,0,d^j) = 0,\ \varphi(0,0,d^j) = 0,$ *and* $\phi(0,d^j) = 0$ *where* $\mathbb{Q} = \{$max, nominal, min$\}$
- *(Constraint set) The state bounds $\mathbb{X}$, and terminal region $\mathbb{X}_f \in \mathbb{X}$ are closed, and control set $\mathbb{U}$ is compact. All sets contain the origin.*
- *(Common terminal region) For all $j \in \mathbb{Q}^{n_d}$, $\exists$ a common $\mathbb{X}_f$ that is control invariant for $x_{k+1} = f(x_k, u_k, d_k^j)$ and $u_k \in \mathbb{U},\ \forall x_k \in \mathbb{X}_f$*

Remark 34. *The set of common terminal region is not empty because at least $\{0\} \subseteq \mathbb{X}_f$. Additionally, note that the choice of the common terminal region is not necessarily unique. In this study, $\mathbb{X}_f = \bigcap_{j \in \mathbb{Q}^{n_d}} \mathbb{X}_f^j$, in which $\mathbb{X}_f^j$ is defined as the control invariant set for system $x^+ = f(x,u,d^j)$ with some $j \in \mathbb{Q}^{n_d}$ and $\forall x \in \mathbb{X}_f^j$.*

Under the restrictions of Assumption 33, Problem $P_N^{id}(x_k)$ (4.3) can be rewritten as Problem $P_N^{fi}(x_k)$ for the fully-expanded ideal-msNMPC problem:

$$J_N^{fi}(x_k) = \min_{z_l^c, v_l^c} \sum_{c \in \mathbb{C}} p^c \Big(\phi(z_N^c, d_N^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l^c, d_l^c) \Big) \qquad (\mathcal{P}_N^{fi}(x_k))$$

$$\text{s.t.} \qquad z_{l+1}^c = f(z_l^c, v_l^c, d_l^c) \quad l = 0, ..., N-1 \qquad (5.17a)$$

$$z_0^c = x_k \qquad (5.17b)$$

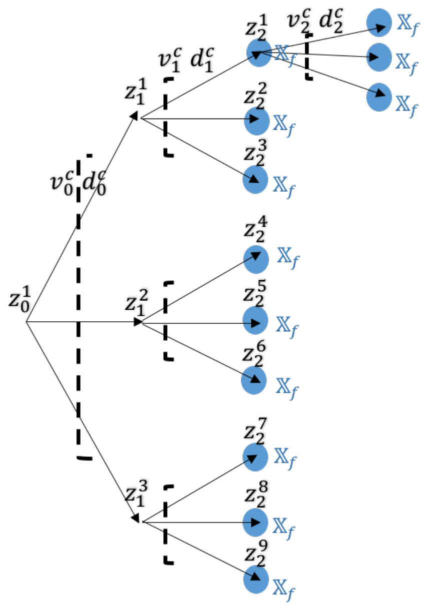


Figure 5.1: The 9-scenario fully-expanded tree with each scenario's terminal region.

$$v_l^c = v_l^{\bar{c}} \quad \text{if} \quad z_l^c = z_l^{\bar{c}} \tag{5.17c}$$

$$z_l^c \in \mathbb{X}, v_l^c \in \mathbb{U}, z_N^c \in \mathbb{X}_f, \forall c, \bar{c} \in \mathbb{C}, d_l^c \in \mathbb{D} \tag{5.17d}$$

In this section, unless otherwise specified, $J_N^{fi}(x_k)$ is often written as $J_N(x_k)$.

To facilitate the discussion, we define p_l^j as the probability (weight) from z_l to $z_{l+1} = f(z_l, v_l, d_l^j)$, where $j \in \mathbb{Q}^{n_d}, l = 0, 1, ..., N-1$.

It is known that,

$$\sum_{j \in \mathbb{Q}^{n_d}} p_l^j = 1, \quad \forall\, l \tag{5.18}$$

We can then calculate the probability (weight) p^c for all scenarios $c \in \mathbb{C}$ in this way:

$$p^c = \prod_{l=0}^{N-1} p_l^{j_l} \tag{5.19}$$

where scenario c can be represented as $c = \{j_0, j_1, ..., j_{N-1}\}$. For example, in Fig. 5.1, we calculate $p^1 = p_0^{max} * p_1^{max} = \frac{1}{3} * \frac{1}{3} = \frac{1}{9}$ for the scenario 1 under uniform distribution.

In this chapter, we assume $p_l^j = p_{l'}^j$ for $l, l' = \{0, 1, ..., N-1\}$. As a result, p^c remains the same for all horizons k. Note that p^c and $p^{c'}$ for $c, c' \in \mathbb{C}$ are not necessarily the same.

In this section, superscript fi related to (5.17) is suppressed for brevity (i.e. J_N refers to J_N^{fi}).

Theorem 35. *(Recursive feasibility of ideal multistage NMPC) Suppose Assumption 33 holds, then problem $\mathcal{P}_N^{fi}(x)$ is recursively feasible: $x \in \mathbb{X}_N$ implies $x^+ = f(x, u, d^j) \in \mathbb{X}_N,\ \forall j \in \mathbb{Q}^{n_d}$*

Proof. (modified from Proposition 4 [79], Theorem 3.9 and Proposition 3.14 [28] and Definition 3 [34]).

Let $\mathbb{X}_N$ be the set of initial conditions where $\mathcal{P}_N^{fi}(x)$ is feasible. That is, $\mathbb{X}_N$ is the admissible state set that can be steered to the terminal region $\mathbb{X}_f$ in N steps.

Select $\mathbb{X}_f = \bigcap\limits_{j \in \mathbb{Q}^{n_d}} \mathbb{X}_f^j$, then $\forall j \in \mathbb{Q}^{n_d}$ let $\mathbb{X}_0^j := \mathbb{X}_f^j$ and define

$$\mathbb{X}_1^j = \{x \in \mathbb{X} \,|\, \exists\, u \in \mathbb{U} : f(x, u, d^j) \in \mathbb{X}_0^j\ \}$$

By Assumption 33 and Remark 34, each terminal set $\mathbb{X}_f^j$ is control invariant. Since $\mathbb{X}_0^j := \mathbb{X}_f^j$, each $\mathbb{X}_0^j$ is also control invariant. By construction, for each parametric disturbance $j \in \mathbb{Q}^{n_d}$, $\mathbb{X}_1^j$ is the set of all states that can be routed into $\mathbb{X}_0^j$. Then $\mathbb{X}_1^j \supseteq \mathbb{X}_0^j$ for each $j \in \mathbb{Q}^{n_d}$.

Define

$$\mathbb{X}_1 = \bigcap_{j \in \mathbb{Q}^{n_d}} \mathbb{X}_1^j$$

Since $\bigcap\limits_{j \in \mathbb{Q}^{n_d}} \mathbb{X}_1^j \supseteq \bigcap\limits_{j \in \mathbb{Q}^{n_d}} \mathbb{X}_0^j = \mathbb{X}_0$, thus $\mathbb{X}_1 \supseteq \mathbb{X}_0$.

By induction, $\mathbb{X}_N \supseteq \mathbb{X}_{N-1} \supseteq ... \supseteq \mathbb{X}_1 \supseteq \mathbb{X}_0 = \mathbb{X}_f$. Hence, for $i \geq 0$, $\mathbb{X}_i$ is a control invariant set for all $j \in \mathbb{Q}^{n_d}$ and corresponding $f_j(\cdot)$. If $x \in \mathbb{X}_N$, then $x^+ \in \mathbb{X}_{N-1} \subseteq \mathbb{X}_N$.

Now suppose $x \in \mathbb{X}_N$, and we have the optimal control for scenario c is $u^{*c}(x) = \{v_0^c, v_1^c, ..., v_{N-1}^c\}$ for problem $\mathcal{P}_N^{fi}(x)$. Each control policy will drive x into $\mathbb{X}_0 = \mathbb{X}_f$ in

N steps. Since $\mathbb{X}_f$ is control invariant, the control sequence $\{v_0^c, v_1^c, ..., v_{N-1}^c, u_f\}$ is feasible for problem $\mathcal{P}_{N+1}^{fi}(x)$. A graphic illustration is represented in Fig. 5.1. This proves the recursive feasibility property. ■

Definition 36. *(Type 1 Robustly Positive Invariant) A set $\mathcal{X} \subseteq \mathbb{X}$ is a Type 1 robustly positive invariant (RPI) set for system $x^+ = f(x,u,d)$ if $x^+ \in \mathcal{X}$ holds for $\forall x \in \mathcal{X}$, and $\forall d \in \mathbb{D}$.*

Definition 37. *(Type 1 Input-to-State Practical Stability) The system $x^+ = f(x,u,d)$ is ISpS in $\mathcal{X}$ if there exists a $\mathcal{KL}$ function β, a $\mathcal{K}$ function γ and $c_0 \geq 0$ such that for all $d \in \mathbb{D}$,*

$$|x_k| \leq \beta(|x_0|, k) + \gamma(|d|) + c_0, \ \forall\, k \geq 0, \ \forall x_0 \in \mathcal{X} \tag{5.20}$$

Definition 38. *(Type 1 ISpS-Lyapunov function) [80] A function $V(\cdot)$ is called an ISpS-Lyapunov function for system $x^+ = f(x,u,d)$ if there exist an RPI set $\mathcal{X}$, $\mathcal{K}$ functions $\alpha_1, \alpha_2, \alpha_3$ and σ, and $c_1, c_2 \geq 0$ such that, $\forall x \in \mathcal{X}$, $\forall w \in \mathbb{W}$ and $\forall d \in \mathbb{D}$,*

$$\begin{aligned} V(x) &\geq \alpha_1(|x|) && \text{(5.21a)} \\ V(x) &\leq \alpha_2(|x|) + c_1 && \text{(5.21b)} \\ \Delta V(x,d) &= V(\hat{f}(x, h(x), d)) - V(x) && \\ &\leq -\alpha_3(|x|) + \sigma(|d|) + c_2 && \text{(5.21c)} \end{aligned}$$

where $h(x)$ is the feedback control law.

Assumption 39. *(ISpS assumptions of ideal-msNMPC)*

- *For each parametric disturbance $j \in \mathbb{Q}^{n_d}$, there exists a local control law $u = h_f(x)$ defined on $\mathbb{X}_f^j$ such that $f(x, h_f(x), d^j) \in \mathbb{X}_f^j, \forall x \in \mathbb{X}_f^j$, and $\phi(f(x, h_f(x), d^j), d^j) - \phi(x, d^{\bar{j}}) \leq -\varphi(x, h_f(x), d^j), \forall x \in \mathbb{X}_f^j$.*
- *For each parametric disturbance $j \in \mathbb{Q}^{n_d}$, the stage cost $\varphi(x,u,d^j)$ satisfies $\alpha_p(|x|) \leq \varphi(x,u,d^j) \leq \alpha_q(|x|) + \sigma_q(|d|)$ where $\alpha_p(\cdot)$, $\alpha_q(\cdot)$ and $\sigma_q(\cdot)$ are $\mathcal{K}$ functions.*

- *The solution of ideal-msNMPC satisfies LICQ and SSOSC such that Theorem 25 applies.*

Theorem 40. *(adapted from [31, 73]) Let $\mathcal{X}$ be a robustly invariant set for system $x^+ = f(x, u, d)$ that contains the origin and let $V(\cdot)$ be a Type 1 ISpS-Lyapunov function for this system, with Assumptions 33 and 39 satisfied, then the resulting system is Type 1 ISpS in $\mathcal{X}$.*

Proof. Let $x_{k+1} = f(x_k, u_k, \hat{d}_k)$, where $\hat{d}_k \in \mathbb{D}$ is the realized value of d_j where $j \in \mathbb{Q}^{n_d}$ for t_k with associated weight p_j.

Based on Theorem 35, recursive feasibility is guaranteed and the control sequence for each scenario $\{v_0^c, v_1^c, ..., v_{N-1}^c, h_f(z_N)\}$ from the optimal solution of $\mathcal{P}_N^{fi}(x_k)$ is feasible for problem $\mathcal{P}_{N+1}^{fi}(x_k)$. By applying this suboptimal solution and writing the corresponding value function as $\widetilde{J}_{N+1}(x_k)$, the following is valid from Assumption 39:

$$\begin{aligned} & \widetilde{J}_{N+1}(x_k) - J_N(x_k) \\ = & \sum_{c \in \mathbb{C}_{N+1}} p^c(\phi(z_{N+1}^c, d_N^c) + \varphi(z_N^c, v_N^c, d_N^c) - \phi(z_N^c, v_N^c)) \\ \leq & \ 0 \end{aligned} \tag{5.22}$$

Define $\mathbb{E}(J_N(x_k, u_k)) = \sum_{j \in \mathbb{Q}^{n_d}} p_j J_N(f(x_k, u_k, d_k^j))$ as the weighted optimal value functions to $|\mathbb{Q}|^{n_d}$ individual subtree problems where u_k is implemented at x_k. We can then write

$$\begin{aligned} \mathbb{E}(J_N(x_k, u_k)) + \sum_{j \in \mathbb{Q}^{n_d}} p_j \varphi(x_k, u_k, d_k^j) &= \sum_{j \in \mathbb{Q}^{n_d}} p_j (J_N(f(x_k, u_k, d_k^j)) + \varphi(x_k, u_k, d_k^j)) \\ &\leq \widetilde{J}_{N+1}(x_k) \\ &\leq J_N(x_k) \end{aligned} \tag{5.23}$$

where the first inequality follows because $\mathbb{E}(J_N(x_k, u_k))$ is composed of optimal solutions to the same subtree problems appearing in $\widetilde{J}_{N+1}(x_k, u_k)$ only differing in the current stage at x_k; and the second inequality follows from (5.22).

Next, we show the difference between the value function for a realized x_{k+1} and a weighted one is bounded:

$$\begin{aligned} J_N(x_{k+1}) - \mathbb{E}(J_N(x_k, u_k)) &= \sum_{j \in \mathbb{Q}^{n_d}} p_j \big(J_N(x_{k+1}) - J_N(f(x_k, u_k, d_k^j))\big) \\ &\leq \sum_{j \in \mathbb{Q}^{n_d}} p_j L_J |f(x_k, u_k, \hat{d}_k) - f(x_k, u_k, d_k^j)| \\ &\leq \sum_{j \in \mathbb{Q}^{n_d}} p_j L_J L_f |\hat{d}_k - d_k^j| \\ &\leq \sigma_0(|\hat{d}_k|) + c_d \end{aligned} \tag{5.24}$$

where $c_d = \sum_j p_j L_J L_f |d_k^j|$. Theorem 25 and Assumption 39 corroborate the first inequality, and Assumption 33 satisfies the second inequality.

Finally, from the above relations we can show the descent property:

$$\begin{aligned} & J_N(x_{k+1}) - J_N(x_k) \\ = \;& J_N(x_{k+1}) - \mathbb{E}(J_N(x_k, u_k)) + \mathbb{E}(J_N(x_k, u_k)) - J_N(x_k) \\ \leq \;& \sigma_0(|\hat{d}_k|) + c_d - \sum_{j \in \mathbb{Q}^{n_d}} p_j \varphi(x_k, u_k, d_k^j) \\ \leq \;& -\alpha_6(|x_k|) + \sigma_0(|\hat{d}_k|) + c_d \end{aligned} \tag{5.25}$$

The result proves that ideal-msNMPC is ISpS-stable when Type 1 uncertainty is present.
■

Note that Type 1 ISpS reverts to nominal (asymptotic) stability when $\hat{d}_k = d_k^j = \bar{d}_k$ (see (6.17)) and thus $|\hat{d}_k - d_k^j| = 0$, meaning that every scenario in ideal-msNMPC becomes the nominal scenario. In this case, the ideal multistage NMPC defaults to nominal NMPC with asymptotic stability.

5.3.2 ISpS with Type 1 and 2 uncertainty

Definition 41. *(Type 1, 2 Robustly Positive Invariant) A set $\mathcal{X} \subseteq \mathbb{X}$ is a Type 1, 2 robustly positive invariant (RPI) set for system $x^+ = f(x, u, d) + w$ if $x^+ \in \mathcal{X}$ holds for $\forall x \in \mathcal{X}$, $\forall d \in \mathbb{D}$ and $\forall w \in \mathbb{W}$.*

Definition 42. *(Type 1, 2 Input-to-State Practical Stability) The system (6.23b) is ISpS in $\mathcal{X}$ if there exists a $\mathcal{KL}$ function β, a $\mathcal{K}$ function γ and $c_0 \geq 0$ such that for all $d \in \mathbb{D}, w \in \mathbb{W}$,*

$$|x_k| \leq \beta(|x_0|, k) + \gamma(|d| + |w|) + c_0, \ \forall\, k \geq 0, \ \forall x_0 \in \mathcal{X} \tag{5.26}$$

Definition 43. *(Type 1, 2 ISpS-Lyapunov function) A function $V(\cdot)$ is called an ISpS-Lyapunov function for system (6.23b) if there exist an RPI set $\mathcal{X}$, $\mathcal{K}$ functions $\alpha_1, \alpha_2, \alpha_3$ and σ, and $c_1, c_2 \geq 0$ such that, $\forall x \in \mathcal{X}$, $\forall d \in \mathbb{D}$, and $\forall w \in \mathbb{W}$,*

$$V(x) \quad \geq \quad \alpha_1(|x|) \tag{5.27a}$$

$$V(x) \quad \leq \quad \alpha_2(|x|) + c_1 \tag{5.27b}$$

$$\begin{aligned} \Delta V(x, d, w) \quad &= \quad V(\hat{f}(x, h(x), d) + w)) - V(x) \\ &\leq \quad -\alpha_3(|x|) + \sigma(|d| + |w|) + c_2 \end{aligned} \tag{5.27c}$$

Lemma 44. *[31, 73] Let $\mathcal{X}$ be a robustly invariant set for system (6.23b) that contains the origin and let $V(\cdot)$ be a Type 1,2 ISpS-Lyapunov function for this system, then the resulting system is ISpS in $\mathcal{X}$.*

To consider both Type 1 and 2 uncertainty, we make the following distinctions that the *realized* states x_{k+1} with d_k^j are

$$x_{k+1} = f(x_k, u_k, d_k^j) + w_k \tag{5.28}$$

with the corresponding *predicted* states $x_{k+1|k}^j = f(x_k, u_k, d_k^j)$ and $u_k = h^{id}(x_k)$.

The future mismatch is defined as follows,

$$w_k = x_{k+1} - x^j_{k+1|k} \tag{5.29}$$

We define the mismatch term for the total costs as the objective of the formulation,

$$\epsilon(x_{k+1}) = J_N(x_{k+1}) - J_N(x^j_{k+1|k}) \tag{5.30}$$

Theorem 45. *(Robust ISpS Stability of ideal-msNMPC) Under Assumptions 33 and 39, the objective function $J_N(x)$ of ideal-msNMPC formulation is a Type 1, 2 ISpS-Lyapunov function and the resulting system is ISpS-stable under Type 1 and 2 uncertainties.*

Proof: From Theorem 25 and Assumption 39, there exists a local positive Lipschitz constant L_J such that $\forall x \in \mathcal{X}$,

$$|\epsilon(x_{k+1})| \quad \leq \quad L_J|w_k|. \tag{5.31}$$

We compare the costs of the neighboring problems $\mathcal{P}^{fi}_N(x_k)$ and $\mathcal{P}^{fi}_N(x_{k+1})$ and introduce the effect of disturbances through $\epsilon(x_{k+1})$, where we consider the optimized values from (5.17),

$$\begin{aligned} J_N(x_{k+1}) - J_N(x_k) &= J_N(x^j_{k+1|k}) - J_N(x_k) + J_N(x_{k+1}) - J_N(x^j_{k+1|k}) \\ &\leq -\alpha_6(|x_k|) + \sigma_3(|\hat{d}_k|) + c_d + L_J|w_k| \\ &\leq -\alpha_6(|x_k|) + \sigma_4(|\hat{d}_k| + |w_k|) + c_d \end{aligned}$$

$\forall \hat{d}_k \in \mathbb{D}, \forall w_k \in \mathbb{W}$. Again, the constant can be computed as $c_d = \sum_j p_j L_J L_f |d^j_k|$. The descent property holds for any qualified x_{k+1} from Theorem 40 and (5.29). The result hereby proves that ideal-msNMPC is Type 1 and 2 ISpS-stable. ■

5.4 Advanced-step multistage NMPC

We now extend the stability discussions to advanced-step multistage NMPC. First, we write the formulation of the optimization problem in as-msNMPC as $\mathcal{P}_N^{as}(x_k)$ which can be seen as (5.17) with an additional constraint (5.32c) for the predicted states at t_{k+1} using the advanced-step control action.

$$J_N^{as}(x_k, h^{as}(x_k)) = \min_{z_l^c, v_l^c} \quad \sum_{c\in\mathbb{C}} p^c\Big(\phi(z_N^c, d_N^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l^c)\Big) \tag{$\mathcal{P}_N^{as}(x_k)$}$$

$$\text{s.t.} \quad z_{l+1}^c = f(z_l^c, v_l^c, d_l^c) \quad l = 1, ..., N-1 \tag{5.32a}$$

$$z_0^c = x_k \tag{5.32b}$$

$$z_1^c = f(x_k, h^{as}(x_k), d_k^c) \tag{5.32c}$$

$$v_l^c = v_l^{\bar{c}} \quad \text{if} \quad z_l^c = z_l^{\bar{c}} \tag{5.32d}$$

$$z_l^c \in \mathbb{X}, v_l^c \in \mathbb{U}, z_N^c \in \mathbb{X}_f, d_l^c \in \mathbb{D}, \forall c, \bar{c} \in \mathbb{C} \tag{5.32e}$$

Similarly, the realized states are represented by $x_{k+1} = f(x_k, h^{as}(x_k), \hat{d}_k) + w_k$ for $\hat{d}_k \in \mathbb{D}$, in which the predicted states are $x_{k+1|k}^{as} = f(x_k, h^{as}(x_k), \hat{d}_k)$ using the advanced-step control law $h^{as}(x_k)$.

We define two mismatch terms, which represent the error introduced by noise and by NLP sensitivity, respectively.

$$\epsilon_{as}(x_{k+1}) \;:=\; J_N^{as}(x_{k+1}) - J_N(x_{k+1}) \tag{5.33a}$$

$$\epsilon_s(x_{k+1}) \;:=\; J_N(x_{k+1}) - J_N(x_{k+1|k}^{as}) \tag{5.33b}$$

Lemma 46. *(Error Bound of sensitivity) [64] From Theorem 25 with $p_0 = x_{k+1|k}$ and $p = x_{k+1} = x_{k+1|k} + w_k$, the approximation error between as-msNMPC and ideal-msNMPC satisfies $|h^{as}(x_{k+1}) - h^{id}(x_{k+1})| \leq L_h|w_k|^2$ with a local positive Lipschitz constant L_h.*

Theorem 47. *(Robust Stability of as-msNMPC) Under Assumptions 33 and 39 the cost function $J_N^{as}(x)$ obtained from the solution of $\mathcal{P}_N^{as}(x,u)$ with $u = h^{as}(x)$ is an ISpS-Lyapunov function under Type 1 and 2 uncertainties. Thus the resulting closed-loop system is ISpS stable.*

Proof: Under Theorems 25 and 46, there exist positive Lipschitz constants L_J, L_h, L_h^{as} such that for all $x \in \mathcal{X}$

$$\begin{aligned}
\epsilon_{as}(x_{k+1}) &\leq L_J|h^{id}(x_{k+1}) - h^{as}(x_{k+1})| \\
&\leq L_J L_h^{as}|w_k|^2 \\
\epsilon_s(x_{k+1}) &\leq L_J(|x_{k+1} - x_{k+1|k}^{as}| + |h^{id}(x_{k+1}) - h^{id}(x_{k+1|k}^{as})|) \\
&\leq L_J(1+L_h)|w_k|
\end{aligned}$$

We now compare the consecutive cost functions $J_N^{as}(x_{k+1})$ and $J_N^{as}(x_k)$ using defined error terms,

$$\begin{aligned}
& J_N^{as}(x_{k+1}) - J_N^{as}(x_k) \\
= \; & J_N^{as}(x_{k+1}) - J_N(x_{k+1}) + J_N(x_{k+1}) - J_N(x_{k+1|k}^{as}) + J_N(x_{k+1|k}^{as}) - J_N^{as}(x_k) \\
= \; & \epsilon_{as}(x_{k+1}) + \epsilon_s(x_{k+1}) + J_N(x_{k+1|k}^{as}) - J_N^{as}(x_k)
\end{aligned}$$

Similar to Theorem 40, using $u_k = h^{as}(x_k)$ we can show that $J_N(x_{k+1|k}^{as}) - J_N^{as}(x_k) \leq -\alpha_7(|x_k|) + \sigma_5(|\hat{d}_k|) + c_d$, hence

$$\begin{aligned}
J_N^{as}(x_{k+1}) - J_N^{as}(x_k) &\leq M|w_k| - \alpha_7(|x_k|) + \sigma_5(|\hat{d}_k|) + c_d \\
&\leq -\alpha_7(|x_k|) + \sigma_6(|\hat{d}_k| + |w_k|) + c_d
\end{aligned}$$

where the last two inequalities follow from $M \geq L_J(1 + L_h + L_h^{as}|w_k|) > 0$. This proves that as-msNMPC is Type 1 and 2 ISpS-stable. ■

Remark 48. *When $w_k = 0$, as-msNMPC satisfies Type 1 ISpS, as with ideal-msNMPC. Also similar to ideal-msNMPC, when $\hat{d}_k = d_k^j = \bar{d}_k$ are valid, Type 1 ISpS reverts to asymptotic stability.*

Chapter 6

Sensitivity-assisted Robust Nonlinear Model Predictive Control with Scenario Generation

We consider an approximation modeling and solution strategy to address multistage stochastic programs for robust NMPC, which we call sensitivity-assisted multistage NMPC (samNMPC). This approach is based on worst-case scenario generation and sensitivity-based approximations for stage costs in the objective function, which leads to an accurate approximate representation of the multi-stage NMPC problem. Moreover, computational costs of this formulation scale independently of the number of disturbance variables. Our novel decomposition method is illustrated on a CSTR case study with two uncertain parameters. Compared to competing approaches, the proposed formulation delivers the robust performance of multi-stage NMPC with significantly less computational cost.

6.1 SamNMPC algorithm

We start with the following discrete-time nonlinear dynamic model:

$$x_{k+1} = f(x_k, u_k, d_k)$$

where $x_k \in \mathbb{X} \subset \mathbb{R}^{n_x}, u_k \in \mathbb{U} \subset \mathbb{R}^{n_u}$ are state and control variables at time step k, and $d_k \in \mathbb{D} \subset \mathbb{R}^{n_d}$ represents the time-varying model parameter. As a typical choice, each element of d_k can take three possible values: {max, nominal, min}.

The sensitivity-assisted multistage NMPC (samNMPC) is built upon the same scenario tree as in regular multistage NMPC, combining with a sensitivity-based approximation algorithm that avoids the growing size of the online optimization problem with respect to number of uncertain parameters. We start by describing conventional multistage NMPC.

6.1.1 Multistage NMPC

Multistage NMPC [16] has been developed at the intersection of stochastic programming and modern control. A scenario tree is formed to represent the state evolutions for all possible uncertain model parameters. In practice, a robust horizon N_r (shorter than prediction horizon N) is also applied to manage a tractable problem size. A scenario tree with robust horizon $N_r = 1$ can be seen as Fig. 6.1, where z_l^c, v_l^c, d_l^c denote the state, control variables, and parameter at stage l and scenario c, respectively. Note that the dashed bracket depicts the non-anticipativity constraint (NAC). NACs are required within the robust horizon to enforce the same control variable for every scenario originating from the same node (state).

To translate Fig. 6.1 to an optimization problem with prediction horizon N, the following formulation is solved for each horizon k

$$J_N(x_k) = \min_{z_l^c, v_l^c} \quad \sum_{c\in\mathbb{C}} p^c \Big(\phi(z_N^c, d_{N-1}^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l^c, d_l^c)\Big) \tag{6.1a}$$

$$\text{s.t.} \quad z_{l+1}^c = f(z_l^c, v_l^c, d_l^c) \quad l = 0, ..., N-1 \tag{6.1b}$$

$$z_0^c = x_k \tag{6.1c}$$

$$v_l^c = v_l^{c'} \quad \{(c, c')| z_l^c = z_l^{c'}\} \tag{6.1d}$$

$$d_{l-1}^c = d_l^c \text{ for } l = N_r, \ldots N-1 \tag{6.1e}$$

$$z_l^c \in \mathbb{X}, v_l^c \in \mathbb{U}, z_N^c \in \mathbb{X}_f^c, d_l^c \in \mathbb{D} \tag{6.1f}$$

$$\forall c, c' \in \mathbb{C} \tag{6.1g}$$

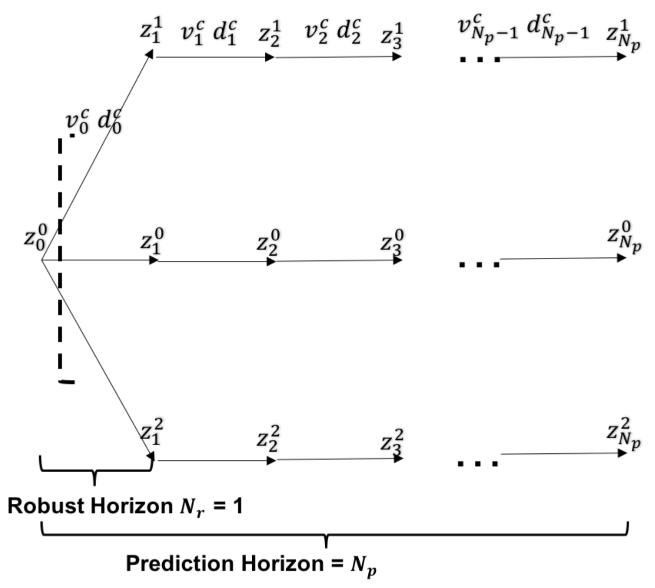

z_1^1 v_1^c d_1^c z_2^1 v_2^c d_2^c z_3^1 $v_{N_p-1}^c$ $d_{N_p-1}^c$ $z_{N_p}^1$
v_0^c d_0^c
z_0^0
z_1^0 z_2^0 z_3^0 $z_{N_p}^0$
z_1^2 z_2^2 z_3^2 $z_{N_p}^2$
Robust Horizon N_r = 1
Prediction Horizon = N_p

Figure 6.1: An example of typical scenario tree in multistage NMPC when $N_r = 1$ and $n_d = 1$

where the objective function (6.1a) contains a weighted terminal costs $\phi(\cdot,\cdot)$ and the integration of stage costs $\varphi(\cdot,\cdot,\cdot)$ over time. (6.1d) denotes the non-anticipativity constraint (NAC) and (6.1e) shows the same uncertain parameter is used for the rest of prediction horizon outside of robust horizon.

6.1.2 NLP sensitivity of multistage NMPC

To explore sensitivity properties for multistage NMPC optimization problem (6.1), we apply a barrier NLP solver such as IPOPT [57]. We can rewrite (6.1) as the following generic parametric program $\mathcal{P}(p)$:

$$\min_{\mathbf{x}} \; F(\mathbf{x};p) \;\; \text{s.t.} \;\; c(\mathbf{x},p) = 0, \; \mathbf{x} \geq 0 \tag{6.2}$$

where the variable vector $\mathbf{x}$ includes all primal variables in (6.1), and p_0 and p_1 represent the parameter in the current scenario and in perturbed scenarios, respectively. IPOPT handles the inequality constraints implicitly through a barrier function in the objective, with parameter μ and solves the following problem:

$$\min_{\mathbf{x}} \; F(\mathbf{x};p) - \mu \sum_{i=1}^{n_{\mathbf{x}}} ln(\mathbf{x}_i) \quad \text{s.t. } c(\mathbf{x},p) = 0. \tag{6.3}$$

After solving a sequence of problems (6.3), with $\mu \to 0$ and $p = p_0$ the solutions of (6.3) approach the solution of (6.2) with $\mathbf{x}^* = \mathbf{x}(p_0)$. Now to see how $\mathbf{x}^*$ varies with respect to perturbations of p, we cite the following property:

Theorem 49. *(NLP Sensitivity) [59, 60]. If $f(\cdot,\cdot)$, $\varphi(\cdot,\cdot)$ and $\phi(\cdot)$ of the parametric NLP problem (6.2) are twice continuously differentiable in a neighborhood of the nominal (primal and dual) solution $s^*(p_0)$ and this solution satisfies the linear independence constraint qualifications (LICQ), strong second order sufficient conditions (SSOSC) and strict complementarity (SC), then the solution $s^*(p_0)$ is differentiable in p.*

Moreover, for $\mu > 0$ but negligibly small, the primal-dual optimality conditions (i.e. KKT conditions) of (6.3) are solved directly at p_0,

$$\begin{aligned} \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*,\lambda^*,\upsilon^*;p_0) &= \nabla_{\mathbf{x}}F(\mathbf{x}^*;p_0) + \nabla_{\mathbf{x}}c(\mathbf{x}^*;p_0)\lambda^* - \upsilon^* = 0 \\ c(\mathbf{x}^*;p_0) &= 0 \\ \mathbf{X}^*\mathbf{V}^*e &= \mu e \end{aligned} \tag{6.4}$$

with $\mathbf{V} = diag(\upsilon), \mathbf{X} = diag(\mathbf{x})$, and $e^T = [1,..,1]$. The Lagrange function is

$$\mathcal{L}(\mathbf{x},\lambda,\upsilon) \triangleq F(\mathbf{x}) + c(\mathbf{x})^T\lambda - \mathbf{x}^T\upsilon \tag{6.5}$$

The primal-dual solution vector is $s(\mu;p)^T = [\mathbf{x}(\mu;p)^T, \lambda(\mu;p)^T, \upsilon(\mu;p)^T]$. Applying Theorem 49 and the implicit function theorem to differentiate (6.4) leads to the following linear system for sensitivity of s.

$$M(s(\mu;p_0))\Delta s = -N(s(\mu;p_0);p) \tag{6.6}$$

where $M(s(\mu;p_0)) = \begin{bmatrix} \nabla_{\mathbf{xx}}\mathcal{L}(s(\mu;p_0)) & \nabla_{\mathbf{x}}c(s(\mu;p_0)) & -I \\ \nabla_{\mathbf{x}}c(s(\mu;p_0))^T & 0 & 0 \\ \mathbf{V}(\mu;p_0) & 0 & \mathbf{X}(\mu;p_0) \end{bmatrix}$ is called the KKT matrix, and $N(s(\mu;p_0);p) = \begin{bmatrix} \nabla_{\mathbf{x}}\mathcal{L}(s(\mu;p_0);p) \\ c(\mathbf{x}(\mu;p_0);p) \\ 0 \end{bmatrix}$ with $s(0;p) = s(\mu;p_0) + \Delta s + O(||p-p_0||^2) + O(\mu)$.

When LICQ, SSOSC, and SC are satisfied at $s(\mu;p_0)$, $M(s(\mu;p_0))$ is nonsingular and the sensitivities Δs can be computed as $\Delta s = -M(s(\mu;p_0))^{-1}N(s(\mu;p_0);p)$ by a backsolve if the factorized form of $M(s(\mu;p_0))$ is available.

6.1.2.1 Block-border structure of linear KKT system

In fact, there is an inherent structure with multistage NMPC problems can be exploited to obtain a faster sensitivity update. To facilitate the discussion, the following notations are used in this section:

- n_c is the number of scenarios minus one
- n is the number of primal variables in each scenario
- m is the number of constraints in each scenario
- m_{NAC} is the total number of NAC constraints

In order to detect this structure, one first writes the KKT matrix of the original primal-dual system as

$$M = \begin{bmatrix} H & \mathbf{A} & -I \\ \mathbf{A}^T & 0 & 0 \\ \mathbf{V} & 0 & \mathbf{X} \end{bmatrix} \tag{6.7}$$

with the Hessian of Lagrange function (6.5) $H = \nabla_{\mathbf{xx}}\mathcal{L}(\mathbf{x}, \lambda, v)$, $\mathbf{A} = \nabla c(\mathbf{x})$ is the transpose of Jacobian.

This version of KKT matrix is nonsymmetric, which limits the type of linear solvers that can be used. One can rewrite (6.6) with a symmetric KKT matrix,

$$\begin{bmatrix} H + \Sigma & \mathbf{A} \\ \mathbf{A}^T & 0 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \lambda, v; p) \\ c(\mathbf{x}; p) \end{bmatrix} \tag{6.8}$$

with $\Sigma \triangleq \mathbf{X}^{-1}\mathbf{V}$ is obtained from eliminating the last row of the original primal-dual system (6.7). Note that the KKT matrix on the left hand side is evaluated at p_0.

If one applies NLP sensitivity to multistage NMPC problem (6.1), one can decouple the system between scenarios and write its KKT matrix as

$$\begin{bmatrix} \mathbf{W} & \mathbf{A} \\ \mathbf{A}^T & \end{bmatrix} = \left[\begin{array}{cccc|cccc} \mathbf{W}_0 & & & & \mathbf{A}_0 & & & \widetilde{N}_0 \\ & \mathbf{W}_1 & & & & \mathbf{A}_1 & & \widetilde{N}_1 \\ & & \ddots & & & & \ddots & \vdots \\ & & & \mathbf{W}_{n_c} & & & \mathbf{A}_{n_c} & \widetilde{N}_{n_c} \\ \hline \mathbf{A}_0^T & & & & & & & \\ & \mathbf{A}_1^T & & & & & & \\ & & \ddots & & & & & \\ & & & \mathbf{A}_{n_c}^T & & & & \\ \widetilde{N}_0^T & \widetilde{N}_1^T & \dots & \widetilde{N}_{n_c}^T & & & & \end{array}\right] \tag{6.9}$$

where $n_c = |\mathbb{C}| - 1$ is the last index of scenarios where $|\mathbb{C}|$ is the number of scenarios.

$\mathbf{W} = H + \Sigma$ is the augmented Hessian for the entire problem. After partitioning, $\mathbf{W}_c = \nabla_{\mathbf{x}^c\mathbf{x}^c}\mathcal{L}(\mathbf{x}, \lambda, \upsilon) + X_c^{-1}V_c$ is the augmented Hessian for scenario c. $\mathbf{W}_c$ is symmetric because the partial Hessian is symmetric and the second term $X_c^{-1}V_c$ is a diagonal matrix.

We also decompose the Jacobian with respect to the set of NAC constraints. $\mathcal{M}$ is the index set for all the constraints (i.e. $|\mathcal{M}| = m * |\mathbb{C}|$). Note $\widehat{\mathcal{M}} = \{1, ..., m_{NAC}\}$ is the index set for NACs and $\overline{\mathcal{M}} = \mathcal{M} \setminus \widehat{\mathcal{M}}$. Then the Jacobian of multistage NMPC problem also have a structure based on NACs.

$$\mathbf{A} = \left[\begin{array}{c|c} \nabla c_i(\mathbf{x}) & \nabla c_j(\mathbf{x}) \end{array} \right] \tag{6.10}$$

where $i \in \overline{\mathcal{M}}$ and $j \in \widehat{\mathcal{M}}$. Then we obtain $\mathbf{A}_c = \nabla_{\mathbf{x}^c} c_i(\mathbf{x})$ and $\widetilde{N}_c = \nabla_{\mathbf{x}^c} c_j(\mathbf{x})$

By rearranging,

$$\begin{bmatrix} \mathbf{W}_0 & \mathbf{A}_0 & 0 & 0 & 0 & 0 & 0 & 0 & \widetilde{N}_0 \\ \mathbf{A}_0^T & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{W}_1 & \mathbf{A}_1 & 0 & 0 & 0 & 0 & \widetilde{N}_1 \\ 0 & 0 & \mathbf{A}_1^T & 0 & 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{W}_{n_c} & \mathbf{A}_{n_c} & \widetilde{N}_{n_c} \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{A}_{n_c}^T & 0 & 0 \\ \widetilde{N}_0^T & 0 & \widetilde{N}_1^T & 0 & \cdots & \cdots & \widetilde{N}_{n_c}^T & 0 & 0 \end{bmatrix} \tag{6.11}$$

the linear system (6.8) can be rewritten in the following block-bordered-diagonal (BBD) form (i.e. arrowhead),

$$\begin{bmatrix} \mathbf{K}_0 & & \dots & N_0 \\ & \mathbf{K}_1 & \dots & N_1 \\ \vdots & \vdots & \ddots & \vdots \\ & & \mathbf{K}_{n_c} & N_{n_c} \\ N_0^T & N_1^T & \dots\ N_{n_c}^T & \end{bmatrix} \begin{bmatrix} \Delta\mathbf{s}_0 \\ \Delta\mathbf{s}_1 \\ \vdots \\ \Delta\mathbf{s}_{n_c} \\ \gamma \end{bmatrix} = - \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_{n_c} \\ 0 \end{bmatrix} \tag{6.12}$$

where $\mathbf{K}_c = \begin{bmatrix} \mathbf{W}_c & \mathbf{A}_c \\ \mathbf{A}_c^T & 0 \end{bmatrix}$, $\Delta\mathbf{s}_c = \begin{bmatrix} \Delta\mathbf{x}^c \\ \Delta\lambda^c \end{bmatrix}$, $r_c = \begin{bmatrix} \nabla_{\mathbf{x}^c}\mathcal{L}(\mathbf{x}, d^c) \\ c(\mathbf{x}^c, d^c) \end{bmatrix}$ for each $c \in \mathbb{C}$.
$\mathbf{x}^c = [z_0^c, v_0^c, z_1^c, v_1^c, ..., z_{N-1}^c, v_{N-1}^c, z_N^c]^T$ denotes the primal variables associated with scenario c, and λ^c for the Lagrange multipliers associated with scenario c.

In (6.12), N_c represents the NAC constraint that contains scenario c, where $N_c = [\widetilde{N}_c,\ 0]^T \in \mathbb{R}^{n+m} \times \mathbb{R}^{m_{NAC}*n_u}$ and n_u is the number of control variables in each scenario. Since all scenarios are constructed with the same variable and same horizon length, n and m are the same across all scenarios. Both N_c and $\widetilde{N}_c$ are very sparse, where the only nonzero elements are 1's and -1's of the corresponding control variables for NAC. Additionally, γ in (6.12) is the Lagrange multiplier of NAC (6.1d) in the dimension $\gamma \in \mathbb{R}^{m_{NAC}*n_u}$.

Since each $\mathbf{W}_c$ is symmetric, each individual $\mathbf{K}_c$ is also symmetric by design. This allows the same type of linear solvers for full-space systems to also apply to the individual system after decomposition.

6.1.2.2 Schur complement decomposition

As the complexity of the linear sensitivity system increases and the total number of scenarios n_c gets large, solving (6.12) as a full-space system may become prohibitively expensive. To exploit a bordered block diagonal structure such as the one in (6.12), we make use of Schur complement decomposition strategy.

The first step is to permute all the $\mathbf{K}_c$ blocks into the bottom row which effectively cancels the bottom NAC blocks,

$$\sum_{c\in\mathbb{C}}(N_c^T\mathbf{K}_c^{-1}N_c)\gamma = -\sum_{c\in\mathbb{C}}(N_c^T\mathbf{K}_c^{-1}r_c) \tag{6.13}$$

where the Schur complement S is formed as,

$$S = \sum_{c\in\mathbb{C}}(N_c^T\mathbf{K}_c^{-1}N_c) \tag{6.14}$$

Then the multipliers for NACs can be solved as

$$\gamma = -S^{-1}\sum_{c\in\mathbb{C}}(N_c^T\mathbf{K}_c^{-1}r_c) \tag{6.15}$$

Finally, the remaining unknowns can be solved as

$$\mathbf{K}_c\Delta\mathbf{s}_c = -(r_c + N_c\gamma),\ \forall c\in\mathbb{C} \tag{6.16}$$

$\Delta\mathbf{s}_c$ is the sensitivity with scenario $c\in\mathbb{C}$, which can then be used to calculate a perturbed solution for that scenario as $\tilde{\mathbf{s}}_c(p) = \mathbf{s}_c(p_0) + \Delta\mathbf{s}_c$. This step is trivially parallelizable.

More importantly, in the case of multistage NMPC, each block $\mathbf{K}_c$ is comprised of the variables in its individual scenario, where the only variables that prevent this block to become completely independent are the ones in NACs. These NAC blocks are permuted and hereby decide the size of the Schur-complement. Since (6.15) can only be solved in serial, the computational overhead grows with respect to the increasing number of coupling variables. When the overhead is small, Schur complement decomposition strategy significantly ourperforms its full-space counterpart [81, 82]. For multistage NMPC specifically, the amount of coupling variables between blocks is only determined by the number of control variables in robust horizon, which is generally small. This provides an exciting opportunity to apply Schur complement to solve linear systems formed by multistage NMPC converted NLP problems.

Solving a single bordered block diagonal linear system with the help of Schur-complement can be orders of magnitude faster than solving a NLP problem. It would expedite the algorithm even further if (6.16) is solved in parallel.

One problem remains: what would be the best way to form a such structured KKT system? If one solves a multistage NMPC problem by a NLP solver such as IPOPT, the KKT matrix in (6.12) can be formed and evaluated at the optimal solution of the multistage NLP problem. But a multistage NLP is expensive to solve due to its many scenarios. In order to approximate a solution of the exact multistage NLP, one can rely on the structure of its KKT matrix. If an approximate linear system to (6.12) can be formed without solving the exact multistage problem, we may also have an approximate solution to the original multistage NMPC. To do that, a single-scenario problem is solved with its single-scenario KKT matrix propagating to the rest of the scenarios. The next section provides details for this procedure.

6.1.3 Nominal NMPC

Standard NMPC (or nominal NMPC) considers only the nominal model in the controller, and it solves the following one-scenario problem:

$$\min_{z_l, v_l} \quad \phi(z_N, \bar{d}_{N-1}) + \sum_{l=0}^{N-1} \varphi(z_l, v_l, \bar{d}_l) \tag{6.17a}$$

$$\text{s.t.} \quad z_{l+1} = f(z_l, v_l, \bar{d}_l) \qquad l = 0, ..., N-1 \tag{6.17b}$$

$$z_0 = x_k \tag{6.17c}$$

$$z_l \in \mathbb{X}, v_l \in \mathbb{U}, z_N \in \mathbb{X}_f \tag{6.17d}$$

Problem (6.17) has a smaller problem size than Problem (6.1), so it is more computationally efficient, but it also loses the robustness guarantee of full-multistage NMPC due to plant-model mismatch [65].

If one applies NLP sensitivity to the standard NMPC problem (6.17), one obtains

$$\mathbf{K}_0 \Delta \mathbf{s}_0 = -r_0 \tag{6.18}$$

where $\mathbf{K}_0 = \begin{bmatrix} \mathbf{W}_0 + \Sigma_0 & \mathbf{A}_0 \\ \mathbf{A}_0^T & 0 \end{bmatrix}$, $\Delta \mathbf{s}_0 = \begin{bmatrix} \Delta \mathbf{x}^0 \\ \Delta \lambda^0 \end{bmatrix}$, $r_0 = \begin{bmatrix} \nabla_{\mathbf{x}^0} \mathcal{L}(\mathbf{x}^0, d^0) \\ c(\mathbf{x}^0, d^0) \end{bmatrix}$.

By solving the nominal NMPC problem, the solution to Problem (6.17) can be used to form the sensitivity system (6.12). If we apply the following approximation $\mathbf{K}_c = \mathbf{K}_0$ to (6.12), we can then solve for the approximate sensitivity solution,

$$\Delta \mathbf{s}_c = \begin{bmatrix} \Delta z^c \\ \Delta v^c \\ \Delta \lambda^c \end{bmatrix} = \begin{bmatrix} \bar{z}^c - z^0 \\ \bar{v}^c - v^0 \\ \bar{\lambda}^c - \lambda^0 \end{bmatrix} \tag{6.19}$$

which provides perturbed solutions for all scenarios $\tilde{\mathrm{s}}(p)$. This computation is particularly efficient since the KKT matrix of the nominal NMPC problem (i.e. $\mathbf{K}_0$) has already been factored and easily reusable in (6.13) and (6.16).

6.1.4 Critical scenarios

The remaining component of the algorithm deals with feasible performance under uncertainty. Robustness of multistage NMPC is often enforced by considering scenarios with extreme-value parameters (i.e. $\max$ or $\min$). If one can predict which parameter values are likely to violate constraints, then the scenarios associated with these values should be treated differently in (6.1) than scenarios where the chances of violating constraints are low. We call the former *critical scenarios*, and the latter *non-critical scenarios*. To determine critical scenarios we apply the approach in [26] to the following dynamic model and constraints:

$$x_{k+1} = f(x_k, u_k, d_k), x_k \in \mathbb{X}, u_k \in \mathbb{U}, \tag{6.20}$$

and discretize the uncertainty description (i.e. $\{\max, \text{nominal}, \min\}$) to develop a scenario tree. At the current state of the plant x_k, critical scenarios are then determined by solving the following optimization problem:

$$\begin{aligned} \max_{d_l} \quad & g_j(z_l, v_l, d_l), \\ s.t. \quad & z_{l+1} = f(z_l, v_l, d_l), \;\; l = 0, \ldots, N-1 \\ & z_0 = x_k \end{aligned} \tag{6.21}$$

where the inequality constraints $g_j(\cdot,\cdot,\cdot)$ represent the state variable bounds $x \in \mathbb{X}$ and j denotes the index of inequality constraints. If a fixed trajectory of $(z_l, v_l)_{l=0,\ldots,N-1}$ is used, and strict monotonicity is assumed for the reduced gradients $dg_j/d(d_l)$, then the analytic solution is easily implemented by setting d_l to appropriate upper or lower bounds at t for each stage l and for each inequality constraint g_j. Based on a reference trajectory for $(z_l, v_l)_{l=0,\ldots,N-1}$, one decides the critical value ($\max$ or $\min$) of each uncertain parameter based on the following criteria:

For $l \in N_r$,

$$\begin{aligned} d^{wc}_{l,m} &= \arg\max_{d \in \mathbb{D}} \nabla_d g_j|^T_{l,(z_l,v_l,d_l)|_{ref}} d_l \\ &= \begin{cases} d^{min}_{l,m}, \text{if } \frac{d(g_j)}{d(d_m)}|_{l,(z_l,v_l,d_l)|_{ref}} \leq 0 \\ d^{max}_{l,m}, \text{otherwise} \end{cases} \quad \text{for } \; m = 1, \ldots, n_d \end{aligned} \tag{6.22}$$

and the number of critical scenarios is bounded by the number of active inequality constraints, which in practice is much smaller than the full-size multistage tree, which consists of $3^{n_d N_r}$ scenarios [26].

We then classify the whole set of scenarios $\mathbb{C} = \{0\} \cup \widehat{\mathbb{C}} \cup \overline{\mathbb{C}}$, as

- $\widehat{\mathbb{C}}$ - critical scenarios where active inequalities may be encountered
- $\{0\}$ - nominal scenario

- $\overline{\mathbb{C}}$ - non-critical scenarios with the primal variables $(\bar{z}_l^c, \bar{v}_l^c)$ determined from sensitivity, where $g_j(\bar{z}_l^c, \bar{v}_l^c) \leq 0$ is expected to hold.

Figs. 6.2 and 6.3 illustrate how the nominal and critical scenarios appear as part of the full multistage scenario tree.

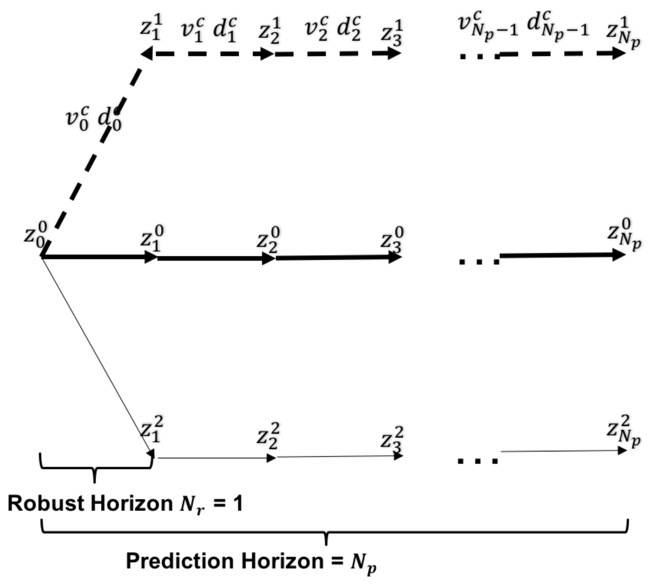
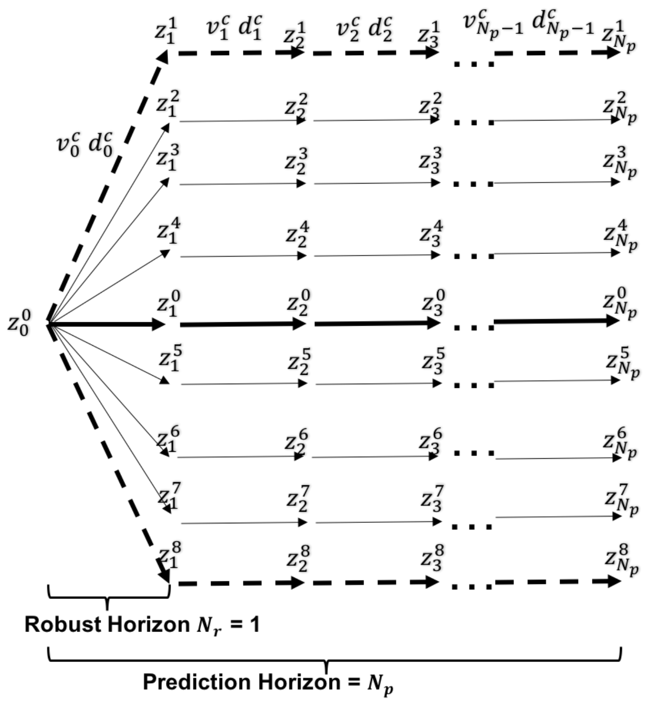

z_1^1 v_1^c d_1^c z_2^1 v_2^c d_2^c z_3^1
$v_{N_p-1}^c$ $d_{N_p-1}^c$ $z_{N_p}^1$
v_0^c d_0^c
z_0^0
z_1^0 z_2^0 z_3^0 $z_{N_p}^0$
z_1^2 z_2^2 z_3^2 $z_{N_p}^2$
Robust Horizon N_r = 1
Prediction Horizon = N_p

Figure 6.2: A example scenario tree where $n_d = 1, N_r = 1$. $\widehat{\mathbb{C}} = \{1\}$ is the critical scenarios set with dashed lines.

One key advantage of this framework is to dynamically update the critical scenarios in each horizon. Practically,

- If one constraint g_j is insensitive to some uncertainty parameter d_m, i.e. $|\frac{d(g_j)}{d(d_m)}| < \epsilon$, it indicates that critical scenarios associated with this constraint, if any, need not be considered for d_m.
- At each time step k in NMPC, active constraints are reevaluated at the updated trajectory (z_l, v_l), together with corresponding worst case parameter values for that con-



z_1^1 v_1^c d_1^c z_2^1 v_2^c d_2^c z_3^1 $v_{N_p-1}^c$ $d_{N_p-1}^c$ $z_{N_p}^1$
v_0^c d_0^c
z_0^0
Robust Horizon N_r = 1
Prediction Horizon = N_p

Figure 6.3: A example scenario tree where $n_d = 2, N_r = 1$. $\widehat{\mathbb{C}} = \{1, 8\}$ is the critical scenarios set with dashed lines.

straint.

- Based on the active constraint g_j, a set of worst case parameter values d^{wc} is selected, and the corresponding critical scenarios $c \in \widehat{\mathbb{C}}$ are included in the final NLP, while the non-critical scenarios $c \in \overline{\mathbb{C}}$ will be approximated by (6.12) through sensitivity.

6.1.5 Approximate multistage problem

The resulting approximate problem is given by

$$\min_{z_l, v_l} \quad \sum_{c \in \widehat{\mathbb{C}} \cup \{0\}} p^c \Big(\phi(z_N^c, d_{N-1}^c) + \sum_{l=0}^{N-1} \varphi(z_l^c, v_l^c, d_l^c) \Big) +$$

$$\sum_{c\in\overline{\mathbb{C}}} p^c\Big(\phi(z_N^0+\Delta z_N^c, d_{N-1}^c)+\sum_{l=0}^{N-1}\varphi(z_l^0+\Delta z_l^c, v_l^0+\Delta v_l^c, d_l^c)\Big) \tag{6.23a}$$

$$\text{s.t.}\quad z_{l+1}^c = f(z_l^c, v_l^c, d_l^c) \qquad c\in\widehat{\mathbb{C}}\cup\{0\},\ l=0,...,N-1 \tag{6.23b}$$

$$z_0^c = x_k \qquad c\in\widehat{\mathbb{C}} \tag{6.23c}$$

$$v_l^c = v_l^{c'} \quad \{(c,c')|z_l^c = z_l^{c'}\} \quad c,c'\in\widehat{\mathbb{C}}\cup\{0\} \tag{6.23d}$$

$$z_l^c\in\mathbb{X}, v_l^c\in\mathbb{U}, z_N^c\in\mathbb{X}_f \tag{6.23e}$$

where $\widehat{\mathbb{C}}$ and $\overline{\mathbb{C}}$ are critical and non-critical scenarios, respectively, and $\Delta z_l^c, \Delta v_l^c$ are obtained from the sensitivity step (6.12) based on the nominal scenario.

The NLP problem (6.23) can be considered as a partially linearized version of problem (6.1), where the non-critical scenarios are no longer constrained by the inequalities, nonlinear equalities and NAC constraints. Instead, the state and control variables of non-critical scenarios are represented approximately as a combination of state and control variables of nominal scenarios and the corresponding perturbation terms calculated by (6.16). Yet, NACs pertinent to non-critical scenarios are still satisfied under (6.12). Additionally, the weighted sum of stage and terminal costs of each scenario in (6.1) still remains such that samNMPC optimizes an expected performance, the same way as the exact multistage problem. By this controller design, non-critical scenarios only appear in the objective function and do not increase the number of variables and constraints in the NLP problem. Thus, it reduces the problem size while still finding an approximate solution to multistage NMPC.

6.1.6 The overall approach for samNMPC

At time step k,

1. Solve Problem (6.17) to get the nominal solution and evaluate $\mathbf{K}_0$.

2. Solve (6.12) using $\mathbf{K}_c = \mathbf{K}_0$ to get $[\Delta z^c, \Delta v^c]^T$.

3. Find the worst case d^{wc} from (6.22) and form the critical scenarios set $\widehat{\mathbb{C}}$

4. Solve (6.23) with all scenarios, where the stage costs for non-critical scenarios are represented by sensitivity solutions

5. Set $u(k) = v_0^c,\ c \in \widehat{\mathbb{C}} \cup \{0\}$ and inject into the plant

6. $k = k + 1$, and go to Step 1

6.1.7 Implementation

The samNMPC method is implemented in CasADi [63]. CasADi is an open-source software toolkit for dynamic optimization problems with state-of-the-art implementation of automatic differentiation (AD). It has become widely used among developers of algorithms for nonlinear optimization problems because of its flexibility over algebraic modeling languages like AMPL. Similar to AMPL, it also includes high-level interfaces to numerical codes for nonlinear programming such as IPOPT. CasADi is written in self-contained C++ code and it also has front-ends to scripting languages Python and Matlab for easy prototyping.

At each step, after solving the nominal NLP using IPOPT [57], the optimal nominal solution is stored as well as the factorization of the nominal KKT matrix $\mathbf{K}_0$. In addition, a group of sparse matrices $\{N_c, c = 0, 1, ..., n_c\}$ needs to be generated based on the number of NACs. The factorization of $\mathbf{K}_0$ can be reused to solve (6.13) - (6.16). Using Schur-complement decomposition avoids solving a large linear system by decoupling into smaller ones with possibility of further parallelization. The linear systems incurred by sensitivity calculations are solved by MA27 from the Harwell Subroutine Library (HSL) [83], and the interface to MA27 is available in CasADi under linear solver class linsol.

6.2 Case studies

The scenario generation algorithm has been applied to a benchmark CSTR example.

$$\begin{aligned}
\frac{dc_A}{dt} &= F(c_{A0} - c_A) - k_1 c_A - k_3 c_A^2 \\
\frac{dc_B}{dt} &= -Fc_B + k_1 c_A - k_2 c_B \\
\frac{dT_R}{dt} &= F(T_{in} - T_R) + \frac{k_W A}{\rho c_p V_R}(T_K - T_R) - \\
&\quad \frac{k_1 c_A \Delta H_{AB} + k_2 c_B \Delta H_{BC} + k_3 c_A^2 \Delta H_{AD}}{\rho c_p} \\
\frac{dT_K}{dt} &= \frac{1}{m_K c_{pK}}(\dot{Q}_K + k_W A(T_R - T_K))
\end{aligned} \tag{6.24}$$

The system has four states $[c_A, c_B, T_R, T_k]$ and two controls $[F, \dot{Q}_K]$. The control objective is to track the setpoint of c_B as $c_B = 0.5 mol/L$ in the first 20 steps, and $c_B = 0.7 mol/L$ for the rest. The stage cost is computed as $L = \sum_l (c_{Bl} - c_B^{ref})^2 + r_1 * (F_l - F_{l-1})^2 + r_2 * (\dot{Q}_{Kl} - \dot{Q}_{Kl-1})^2$. The uncertainty parameters in this case study are $E_{A,3}$ and c_{A0}. Note that the uncertain parameter may change between time steps, but can only choose among a finite set of three values.

The inequalities for this problem are the variable bounds for the state and control variables, where the bounds on control variables are hard constraints. From the sensitivity analysis on each state constraint, we observe that c_B and T_K are insensitive to both uncertainties $E_{A,3}$ and c_{A0}, which means that the perturbation of both parameters will not affect the value of c_B and T_K (within the robust horizon). On the other hand, for the first step only, c_A is sensitive to c_{A0} and T_R is sensitive to both $E_{A,3}$ and c_{A0}. This implies that when $E_{A,3}$ is uncertain, the worst $E_{A,3}$ value is determined by the sign of $\frac{dT_R}{d(E_{A,3})}|_{(x_l,u_l,d_l^0)}$. When c_{A0} is uncertain, the worst parameter value is determined by the signs of both $\frac{dc_A}{d(c_{A0})}|_{(x_l,u_l,d_l^0)}$ and $\frac{dT_R}{d(c_{A0})}|_{(x_l,u_l,d_l^0)}$. Also, $N_r = 1$ for all the cases studied here.

6.2.1 Single uncertain parameter $d = E_{A,3} \pm 10\%$

For this case, the only uncertain parameter in the system is the activation energy $d = E_{A,3}$ and its three possible values are $\{\max, \text{nom}, \min\}$. From the sensitivity analysis we know that the only affected state is T_R and the sensitivity $\frac{dT_R}{d(E_{A,3})}|_{(x_l,u_l,d_l^0)}$ is computed to determine the worst case parameter value. From the case study, the sign of $\frac{dT_R}{d(E_{A,3})}|_{(x_l,u_l,d_l^0)}$ is usually negative, which renders the worst case value for $d^{wc} = E_{A,3}^{min}$. In this case, scenario generation multistage has only two scenarios (instead of three scenarios in conventional multistage and min-max problems).

Table 6.1 provides an overall performance comparison for different robust NMPC schemes by averaging 10 random parameter realizations in the plant. The integrated error is computed as the integral of tracking errors for realized states with respect to time. The CPU seconds are recorded as wall time for different algorithms. Fig. 6.4 plots trajectories of state and control variables, and objective functions for different NMPC controllers in one sample run.

Fig. 6.4 shows a close resemblance of tracking performance between multistage and sensitivity-assisted multistage (samNMPC), especially in terms of state trajectories and objective values where two lines overlap almost entirely. This observation is also supported by the similar tracking errors of conventional and approximate multistage in Table. 6.1 for the one parameter case. At the same time, samNMPC requires less computation than conventional multistage NMPC. In particular, the sensitivity step, obtained by solving a linear system, only comprises 3% of the total computational time which presents a very light computational footprint compared to solving NLPs.

For the other controller schemes, nominal NMPC seems to have a lower tracking error on average along with fast computing, but it often suffers from non-robust control performance. For instance, Fig. 6.4 shows that both c_A and T_R violate the upper bound of state constraints. On the other hand, min-max NMPC is able to stay robust, but performs more conservatively, which is also corroborated by the trajectories of objective functions shown in Fig. 6.4.

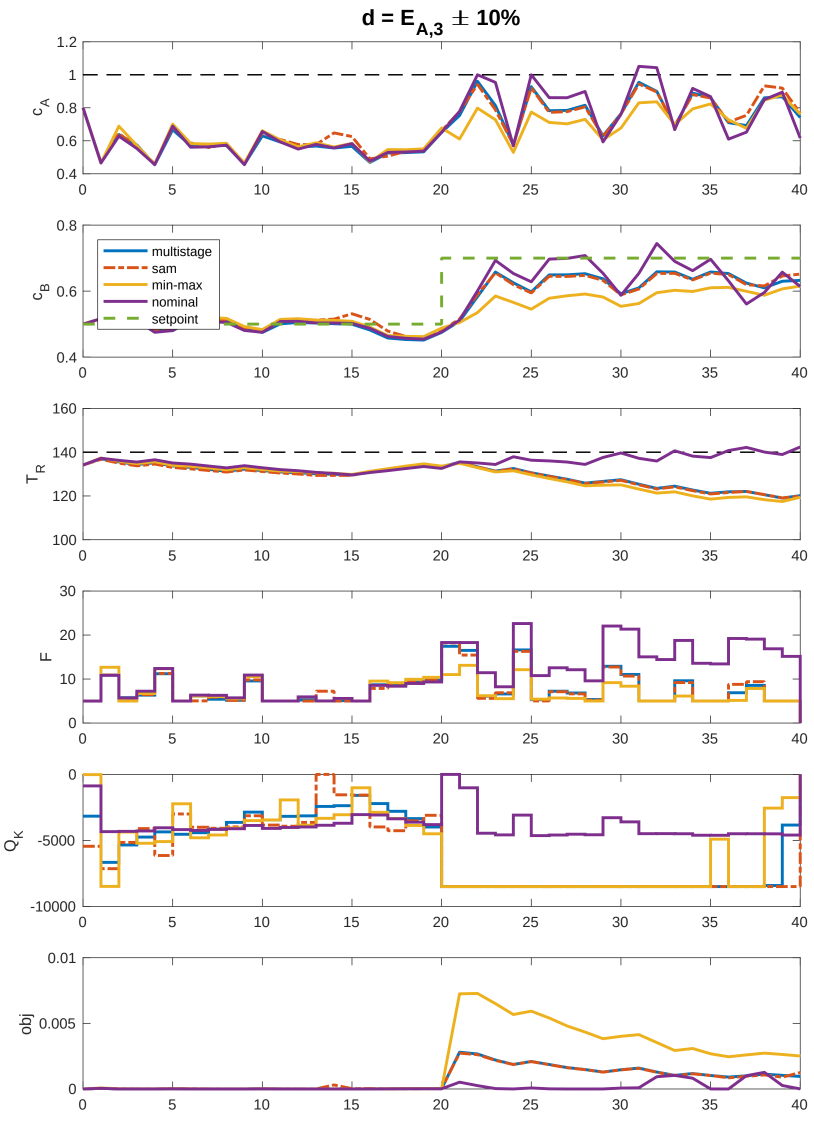


Figure 6.4: Trajectories for state and controls for different robust NMPC schemes with $d = E_{A,3}$.

Table 6.1: Average performances (10 random runs) of robust NMPC schemes with robust horizon $N_r = 1$, $d = [E_{A,3} \pm 10\%]$ (top) and $d = [c_{A0} \pm 30\%, E_{A,3} \pm 10\%]$ (bottom).

		min-max	multistage	nominal	samNMPC
3c	Error	0.3077	0.1435	0.1238	0.1406
	CPUs	0.367	0.251	0.0611	0.178
9c	Error	0.2158	0.2429	0.3135	0.1457
	CPUs	0.839	0.926	0.0806	0.366

6.2.2 Two uncertain parameters $d = E_{A,3} \pm 10\%, d = c_{A,0} \pm 30\%$

For this section, both uncertainties are considered, which renders a scenario tree of 9 scenarios. Similarly, the sensitivity analysis leaves the problem size of sensitivity-assisted multistage (samNMPC) the same as with 3 scenarios. For the min-max and multistage formulation, the problem size is as large as 6500 variables due to the consideration of all 9 scenarios so that the computational time is relatively long.

Again, for the two uncertain parameter case, sensitivity-assisted multistage (samNMPC) and conventional multistage perform closely in terms of state and control trajectories. Most importantly, samNMPC achieves a robust tracking performance with only a fraction of computational resources (again, linear algebra calculations only cost 0.023 second, which contributes to less than 10% of the total computational time). Unsurprisingly, nominal NMPC suffers from large constraint violations, while min-max NMPC also behaves quite close to multistage NMPC in this example.

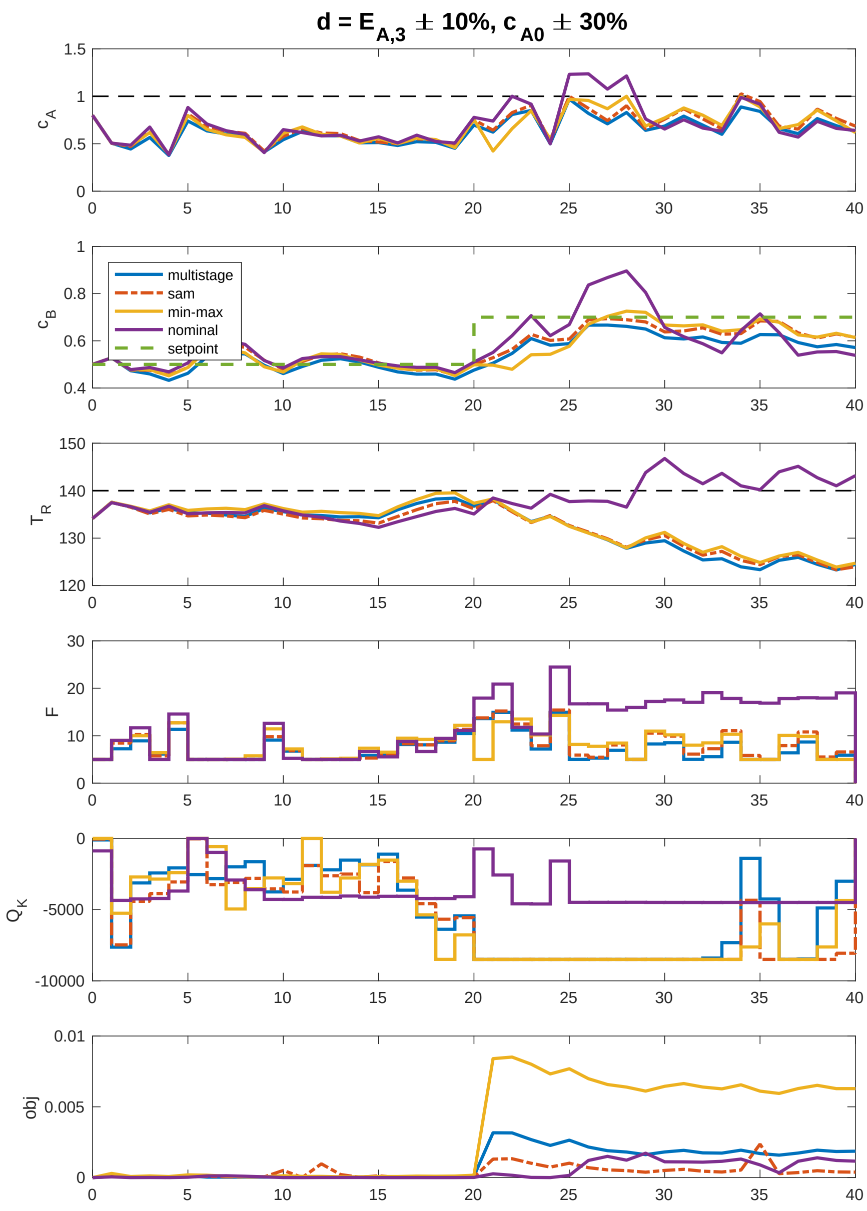


Figure 6.5: Trajectories for state and controls for different robust NMPC schemes with $d = E_{A,3}, c_{A0}$.

6.3 Conclusions

We present an approximation algorithm (samNMPC) that resembles the robust multi-stage NMPC performance but with far less computation effort. The conventional multi-stage NMPC problem size grows exponentially with respect to the number of uncertain parameters and robust horizon; hence it becomes difficult to solve online. By separating all scenarios into critical and non-critical categories, one can contain the size of the optimization problem. Only the critical scenarios are added to the problem, whereas the non-critical scenarios are represented by the nominal scenario with sensitivity corrections. In this way, the optimization problem size is only determined by the number of critical scenarios, which is much smaller than the entire scenario tree in traditional multistage NMPC formulations. We apply this approximation strategy to a CSTR benchmark problem, where two case studies with one and two uncertain parameters are explored. SamNMPC achieves robustness and similar tracking performance with respect to conventional multistage NMPC with only a fraction of computational effort.

Chapter 7

Conclusions

7.1 Summary and contributions

This thesis has pushed forward the state-of-the-art of robust NMPC under two distinct types of uncertainties. A cohesive and structured framework has been created to design a robust NMPC strategy that is performance-driven, stability-guaranteed, and computationally efficient. To achieve that, optimal control theory and nonlinear optimization theory are intertwined to help further the application of robust NMPC and make better decisions.

Chapter 1 briefly covers the history of MPC and the general background of advanced control in a process industry. Specifically, many research efforts have been directed to design robust MPC controllers in the past decades. The main challenges are robustness, performance and computation. The meaning of robustness is twofold: robust stability and robust to constraints. As a robust controller, it also needs to achieve the desired tracking performance, same as any other controller, but under the influence of uncertainties. And all of these should be accomplished with a tight computational budget. To make the goal possible, advanced toolboxes are required.

Chapter 2 establishes the fundamental definitions and concepts of stability analysis and basic NMPC ideas. In particular, Lyapunov stability theory is introduced and discussed, as well as the nominal stability and robust stability definitions, which are the building blocks for examining stability properties of different variants of NMPC. Additionally, for the reason of computational convenience, infinite horizon MPC evolves to finite horizon

MPC and becomes the canonical form of MPC that is widely used at the current time.

Chapter 3 describes the general class of dynamic optimization problem and several aspects of solution strategies. Dynamic optimization problems appear in a variety of industries and applications, but they cannot be solved directly with integrals or differential equations. A common practice is to discretize the continuous-time dynamics equation to an algebraic form and send that to an NLP solver instead. In particular, the discretization method that uses orthogonal collocation over finite elements has shown high accuracy and numerical stability, and it is used throughout this dissertation. To understand an NLP problem, basic NLP concepts are introduced. In this thesis, IPOPT has been applied as the default NLP solver. Additionally, NLP sensitivity strategies are introduced to obtain fast approximate solutions for perturbed problems.

Chapter 4 proposes a new design for a robust NMPC controller - advanced-step multistage NMPC. Two types of uncertainty are present in the system: Type 1 uncertainty is the model parameter that can be realized within one sampling time and is handled by the multistage structure; Type 2 uncertainty is the process or measurement noise that is generally not realized and is corrected by the NLP sensitivity updates. A scenario tree is formed to monitor the Type 1 uncertainty evolution, with non-anticipativity constraints (NAC) within robust horizons as complicating constraints. The sensitivity-based method enables multistage NMPC with the adjustment with respect to Type 2 uncertainty. It also allows the control action to be computed one time step ahead, which gives the solver more time to solve for a large multistage problem. Several robust NMPC schemes are also implemented in comparison: standard NMPC, ideal multistage NMPC, min-max NMPC, NMPC with back-off constraints, and perfect-information NMPC. The major contributions of this chapter are listed as follows:

- Proposed a parallelizable advanced-step multistage NMPC (as-msNMPC) framework

- Compared ideal multistage NMPC with state-of-the-art robust NMPC approaches for 3-scenario models under uniformly distributed parameter for a CSTR example
- Demonstrated the effect of different robust horizon lengths and different parameter distribution
- Conducted a feasibility study for robust NMPC schemes with the constraint satisfaction metrics
- Studied the tracking and computational performances for a quadtank case study with both Type 1 only and Type 1+2 uncertainty
- Demonstrated a similar tracking performance between ideal multistage NMPC and advanced-step multistage NMPC, and a significant improvement in online computational efforts by using as-msNMPC over ideal-msNMPC for both CSTR and quadtank examples

Chapter 5 extends the stability analysis to ideal multistage NMPC and advanced-step multistage NMPC. When there is no plant-model mismatch or disturbance in the system, ideal standard NMPC is proven to be asymptotically stable and the system converges to the origin. However, assume Type 2 uncertainty enters the system, the system is only able to converge to a compact neighborhood of the origin, in which the size of the neighborhood is dependent on the size of the Type 2 uncertainty. On the other hand, assume Type 1 and 2 uncertainties are both present, one needs to introduce a new set of definitions of robust stability to handle both types. The contributions of Chapter 5 are listed as follows:

- Analyzed the nominal and robust stability for standard NMPC and advanced-step NMPC with respect to Type 2 uncertainty
- Adapted and proved recursive feasibility property of ideal multistage NMPC
- Proposed an extended Input-to-state practical stability (ISpS) definitions under Type

1 and 2 uncertainties

- Proved that the ideal multistage NMPC is ISpS stable under Type 1 only and Type 1 + 2 uncertainty
- Proved that the advanced-step multistage NMPC is ISpS stable under Type 1 + 2 uncertainty

Chapter 6 presents an approximation algorithm to multistage NMPC problems. Multistage NMPC are known to have a large NLP problem due to its multi-scenario structure. More importantly, non-anticipativity constraints (NAC) become the only coupling constraints that prevent scenarios from being independent and separable. Once the scenarios are decoupled, the problem of each invidual scenario becomes parallelizable and also more manageable. SamNMPC takes advantage of the block-bordered-diagonal structure that comes naturally with multistage NMPC problems, and solves its linear sensitivity system with Schur complement decomposition. After obtaining the Schur-complement, the sensitivity of each scenario can be solved separately and in parallel. This sensitivity result is combined with the worst-case scenarios generated by performing scenario criticality check, such that an approximate multistage NLP is formed with much fewer scenarios and variables.

- Proposed an algorithm sensitivity-assisted multistage NMPC (samNMPC) to approximate solutions to the original multistage NMPC
- Extended the application of NLP sensitivity beyond advanced-step NMPC and perturbations on initial conditions of states, to any number of parameters in the model
- Applied Schur complement decomposition to solve multistage NMPC problems
- Applied a scenario generation technique to select critical scenarios whose number is only proportional to the number of active inequality constraints
- Compared the tracking performance and computational time between samNMPC

and exact multistage NMPC with both one and two uncertain Type 1 parameters on a CSTR system

- Implemented a preliminary version of NLP sensitivity assistance tools in CasADi that helps prototyping multistage NMPC and samNMPC

7.2 Recommendations and future work

After summarizing the work that has been done, we now discuss some interesting topics within the general realm of this thesis that are worth exploring in the future.

7.2.1 Extensions on `samNMPC` framework

The current implementation of samNMPC is in CasADi, and with the extensibility and flexibility of CasADi there are many things that can be done.

By the time this thesis is written, `samNMPC` framework has been able to run two modes with multistage NMPC. Given a dynamic optimization model (e.g. CSTR example) and a control objective (e.g. a setpoint), `samNMPC` can use this information to generate four types of NMPC controllers: nominal NMPC (standard NMPC), min-max NMPC, exact multistage NMPC, and samNMPC, while the last three require a set of possible values for the uncertain parameters along with their probabilities. This encourages researchers to explore the possible gains in robustness and performance of using a multistage formulation without too much hassle.

However, as of right now the implementation has been preliminary and yet user-friendly. Future work is needed to extend `samNMPC` to a fully-functional multistage NMPC package. Compared with general-purpose modeling and optimization packages such as `Pyomo` [84] and `acados` [85], `samNMPC` focuses on NMPC under uncertainty with multistage formulations and its variations. The following features would be useful to add:

– A swift transition between exact and approximate multistage NMPC is desired. If an exact multistage NMPC is permitted within the time limit, an exact formulation will be performed. Otherwise, the approximate multistage NMPC can be employed.
– Implement advanced-step multistage NMPC within `samNMPC`, which provides another option for online computational performance.
– A new algorithm advanced-step samNMPC can be implemented, which is an interplay between advanced-step multistage NMPC of Chapter 4, and samNMPC of Chapter 6. In this case, the online computation remains almost the same, but the background solution is now provided with the approximate multistage NMPC with scenario generation technique. This would result in reductions in both online and background CPU time.

7.2.2 Terminal conditions of multistage NMPC

In Chapter 5, we have discussed that the robust stability of ideal multistage NMPC and advanced-step multistage NMPC can be guaranteed by applying terminal costs and terminal region. Additionally, in order to analyze the recursive feasibility property of ideal multistage NMPC, a common terminal region is assumed to exist for all scenarios. However, calculating such terminal constraints and costs is not straightforward.

One of the key assumptions to ensure recursive feasibility (e.g. Assumption 26) is that there exists a stablizing controller in the control invariant terminal region such that once the state of the system enters the terminal region, a control law can be applied to keep states stay within the terminal region. To compute such a controller, an infinite-horizon LQR to the linearized system has been proposed [86]. In this case the terminal cost is the cost-to-go function of LQR, i.e. $\phi(x) = x^T P x$, where P satisfies the discrete-time Riccati equation

$$A^T PA - P - (A^T PB)(B^T PB + R)^{-1}(B^T PA) + Q = 0 \tag{7.1}$$

where $A = \frac{\partial f(0,0)^T}{\partial x}$, $B = \frac{\partial f(0,0)^T}{\partial u}$ and Q and R are corresponding weights of states and controls in the objective. This also provides the gain matrix $K = (R + B^T PB)^{-1} B^T PA$ and the linear control law is $u_f(x) = -Kx$.

The terminal region is determined by the largest region centered around the origin where the LQR can stabilize the original nonlinear system. Recent developments have been working on finding the size of the terminal region by finding bounds of the higher order nonlinear terms of the system through simulation [45].

One can certainly extend the LQR approach to find the common terminal region for multistage NMPC. With the parametrized nonlinear functions in different scenarios, one may resort to the parametrized LQR.

7.2.3 Decision-dependent Type 1 uncertainty in multistage NMPC

By far, the nature of Type 1 uncertainty considered in this thesis has been *exogenous*, which means that the true value of parameters are revealed independent of control decisions. In general, only exogenous uncertainties have been studied in the field of multistage NMPC, and problems with exogenous uncertainties have also been the focus of the stochastic programming community [68, 87, 88]. Another class of Type 1 uncertainty are called *endogenous*, where the realizations of parameters are dependent on the decisions [89]. There are at least two ways that the decisons can influence the parameters: one by altering the probability distribution of the parameter, and the other by changing the time at which parameters are realized [90]. For a robust control problem, one may find it more relevant to solve the first case where the parameter probability distribution is dependent on the control variables.

Let's start with the dynamics model with Type 1 uncertainty that has the following form:

$$x_{k+1} = \hat{f}(x_k, u_k, d_k) \tag{7.2}$$

Assuming we know a priori d_k can be expressed as some function of u_k as $d_k = d(u_k)$, then we can rewrite (7.2) as,

$$x_{k+1} = f_d(x_k, u_k) \tag{7.3}$$

In the case with standard NMPC (i.e. with only one scenario), there is no apparent difference between solving exogenous and endogenous nominal NMPC. The stability properties of ideal NMPC should also be extended to the decision-dependent standard NMPC.

For multistage NMPC, as with classic multistage NMPC of exogenous Type 1 uncertainty, a finite number of representative values (for instance, 3) are taken from the probability distribution, and they are denoted as $d_k \in \{\underline{d}(u_k), d(u_k), \bar{d}(u_k)\}$. In this case, three models can be represented as $\{f_{\underline{d}}(\cdot), f_d(\cdot), f_{\bar{d}}(\cdot)\}$. Here, decision-dependent multistage NMPC can be treated as multi-model multistage NMPC, in which each model may not share the functional form with others. Computationally, decision-dependent ideal multistage NMPC can be solved using the same solution strategy with decision-independent multistage NMPC. In terms of controller stability, it remains an open question whether robust stability can be extended considering that the recursive feasibility may not be ensured. For discussions regarding multi-model in MPC, readers are referred to [91].

Bibliography

[1] K.-U. Klatt and S. Engell, "Gain-scheduling trajectory control of a continuous stirred tank reactor," *Computers & Chemical Engineering*, vol. 22, no. 4, pp. 491–502, 1998.

[2] S. Lucia, *Robust Multi-stage Nonlinear Model Predictive Control*. Doctoral Thesis, Technical University of Dortmund, Shaker, 2014.

[3] T. Raff, S. Huber, Z. K. Nagy, and F. Allgower, "Nonlinear model predictive control of a four tank system: An experimental stability study," in *Control Applications, 2006. CCA'06. IEEE International Conference on*, pp. 237–242, IEEE, 2006.

[4] J. M. Maciejowski, *Predictive control: with constraints*. Pearson education, 2002.

[5] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[6] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[7] S. Yu, M. Reble, H. Chen, and F. Allgöwer, "Inherent robustness properties of quasi-infinite horizon nonlinear model predictive control," *Automatica*, vol. 50, no. 9, pp. 2269–2280, 2014.

[8] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings, "On the inherent robustness of optimal and suboptimal nonlinear mpc," *Systems & Control Letters*, vol. 106, pp. 68–78, 2017.

[9] A. Bemporad and M. Morari, “Robust model predictive control: A survey,” in *Robustness in identification and control*, pp. 207–226, Springer, 1999.

[10] P. J. Campo and M. Morari, “Robust model predictive control,” in *American Control Conference, 1987*, pp. 1021–1026, IEEE, 1987.

[11] D. Q. Mayne, M. M. Seron, and S. Raković, “Robust model predictive control of constrained linear systems with bounded disturbances,” *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[12] P. Scokaert and D. Mayne, “Min-max feedback model predictive control for constrained linear systems,” *IEEE Transactions on Automatic control*, vol. 43, no. 8, pp. 1136–1142, 1998.

[13] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, “Tube-based robust nonlinear model predictive control,” *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[14] D. Bernardini and A. Bemporad, “Scenario-based model predictive control of stochastic constrained linear systems,” in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 6333–6338, IEEE, 2009.

[15] R. Huang, S. C. Patwardhan, and L. T. Biegler, “Multi-scenario-based robust nonlinear model predictive control with first principle models,” in *Computer Aided Chemical Engineering*, vol. 27, pp. 1293–1298, Elsevier, 2009.

[16] S. Lucia, T. Finkler, and S. Engell, “Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty,” *Journal of Process Control*, vol. 23, no. 9, pp. 1306–1319, 2013.

[17] J. Puschke and A. Mitsos, "Robust feasible control based on multi-stage enmpc considering worst-case scenarios," *Journal of Process Control*, vol. 69, pp. 8–15, 2018.

[18] R. Kumar, J. Jalving, M. J. Wenzel, M. J. Ellis, M. N. ElBsat, K. H. Drees, and V. M. Zavala, "Benchmarking stochastic and deterministic mpc: A case study in stationary battery systems," *AIChE Journal*, p. e16551, 2019.

[19] R. Findeisen and F. Allgöwer, "Computational delay in nonlinear model predictive control," *IFAC Proceedings Volumes*, vol. 37, no. 1, pp. 427–432, 2004.

[20] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[21] C. Leidereiter, A. Potschka, and H. G. Bock, "Dual decomposition for qps in scenario tree nmpc," in *Control Conference (ECC), 2015 European*, pp. 1608–1613, IEEE, 2015.

[22] D. Krishnamoorthy, B. Foss, and S. Skogestad, "A distributed algorithm for scenario-based model predictive control using primal decomposition," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 351–356, 2018.

[23] D. Krishnamoorthy, B. Foss, and S. Skogestad, "A primal decomposition algorithm for distributed multistage scenario model predictive control," *Journal of Process Control*, vol. 81, pp. 162–171, 2019.

[24] E. Klintberg and S. Gros, "A parallelizable interior point method for two-stage robust mpc," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2087–2097, 2017.

[25] W. Daosud, P. Kittisupakorn, M. Fikar, S. Lucia, and R. Paulen, "Efficient robust nonlinear model predictive control via approximate multi-stage programming: A neural

networks based approach," in *Computer Aided Chemical Engineering*, vol. 46, pp. 1261–1266, Elsevier, 2019.

[26] F. Holtorf, A. Mitsos, and L. T. Biegler, "Multistage nmpc with on-line generated scenario trees: Application to a semi-batch polymerization process," *Journal of Process Control*, vol. 80, pp. 167–179, 2019.

[27] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, vol. 10. Siam, 2010.

[28] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[29] E. D. Sontag, "Smooth stabilization implies coprime factorization," *IEEE transactions on automatic control*, vol. 34, no. 4, pp. 435–443, 1989.

[30] E. D. Sontag, "Further facts about input to state stabilization," *IEEE Transactions on Automatic Control*, vol. 35, no. 4, pp. 473–476, 1990.

[31] Z.-P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.

[32] D. Limon, T. Alamo, D. Raimondo, D. M. De La Peña, J. Bravo, A. Ferramosca, and E. Camacho, "Input-to-state stability: a unifying framework for robust model predictive control," in *Nonlinear model predictive control*, pp. 1–26, Springer, 2009.

[33] E. D. Sontag and Y. Wang, "New characterizations of input-to-state stability," *IEEE Transactions on Automatic Control*, vol. 41, no. 9, pp. 1283–1294, 1996.

[34] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, "Input to state stability of min–

max mpc controllers for nonlinear systems with bounded uncertainties," *Automatica*, vol. 42, no. 5, pp. 797–803, 2006.

[35] R. R. Bitmead, M. Gevers, and V. Wertz, *Adaptive optimal control: the thinking man's GPC*. Prentice Hall New York, 1990.

[36] J. B. Rawlings and K. R. Muske, "The stability of constrained receding horizon control," *IEEE Transactions on Automatic Control*, vol. 38, no. 10, pp. 1512–1516, 1993.

[37] G. Pannocchia, J. B. Rawlings, D. Q. Mayne, and G. M. Mancuso, "Whither discrete time model predictive control?," *IEEE Transactions on Automatic Control*, vol. 60, no. 1, pp. 246–252, 2014.

[38] S. a. Keerthi and E. G. Gilbert, "Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations," *Journal of Optimization Theory and Applications*, vol. 57, no. 2, pp. 265–293, 1988.

[39] G. De Nicolao, L. Magni, and R. Scattolini, "Stability and robustness of nonlinear receding horizon control," in *Nonlinear model predictive control*, pp. 3–22, Springer, 2000.

[40] A. Jadbabaie, J. Yu, and J. Hauser, "Unconstrained receding-horizon control of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 5, pp. 776–783, 2001.

[41] D. Mayne, "An apologia for stabilising terminal conditions in model predictive control," *International Journal of Control*, vol. 86, no. 11, pp. 2090–2095, 2013.

[42] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 38, no. 11, pp. 1623–1633, 1993.

[43] L. Grüne, "Economic receding horizon control without terminal constraints," *Automatica*, vol. 49, no. 3, pp. 725–734, 2013.

[44] L. Grüne and J. Pannek, "Nonlinear model predictive control," in *Nonlinear Model Predictive Control*, pp. 45–69, Springer, 2017.

[45] D. W. Griffith, L. T. Biegler, and S. C. Patwardhan, "Robustly stable adaptive horizon nonlinear model predictive control," *Journal of Process Control*, vol. 70, pp. 109–122, 2018.

[46] L. T. Biegler, A. M. Cervantes, and A. Wächter, "Advances in simultaneous strategies for dynamic process optimization," *Chemical engineering science*, vol. 57, no. 4, pp. 575–593, 2002.

[47] M. Vidyasagar, "Nonlinear systems theory," 1993.

[48] L. S. Pontryagin, V. Boltyanskii, R. Gamkrelidze, and E. Mishchenko, "The mathematical theory of optimal processes," *New York*, 1962.

[49] A. Bryson and Y.-C. Ho, "Applied optimal control," *New York*, 1975.

[50] A. Cervantes and L. T. Biegler, "Optimization strategies for dynamic systems," *Encyclopedia of optimization*, pp. 1886–1897, 2001.

[51] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*, vol. 19. Siam, 2010.

[52] A. Flores-Tlacuahuac, L. T. Biegler, and E. Saldívar-Guerra, "Dynamic optimization of hips open-loop unstable polymerization reactors," *Industrial & engineering chemistry research*, vol. 44, no. 8, pp. 2659–2674, 2005.

[53] H. G. Bock and K.-J. Plitt, “A multiple shooting algorithm for direct solution of optimal control problems,” *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[54] H. Bock, M. Diehl, D. Leineweber, and J. Schlöder, “A direct multiple shooting method for real-time optimization of nonlinear dae processes,” in *Nonlinear model predictive control*, pp. 245–267, Springer, 2000.

[55] U. M. Ascher, R. M. Mattheij, and R. D. Russell, *Numerical solution of boundary value problems for ordinary differential equations*, vol. 13. Siam, 1994.

[56] L. T. Biegler and I. E. Grossmann, “Retrospective on optimization,” *Computers & Chemical Engineering*, vol. 28, no. 8, pp. 1169–1192, 2004.

[57] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[58] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[59] A. V. Fiacco, *Introduction to sensitivity and stability analysis in nonlinear programming*. Elsevier, 1983.

[60] A. V. Fiacco, “Sensitivity analysis for nonlinear programming using penalty methods,” *Mathematical programming*, vol. 10, no. 1, pp. 287–311, 1976.

[61] J. Jäschke, X. Yang, and L. T. Biegler, “Fast economic model predictive control based on nlp-sensitivities,” *Journal of Process Control*, vol. 24, no. 8, pp. 1260–1272, 2014.

[62] H. Pirnay, R. López-Negrete, and L. T. Biegler, “Optimal sensitivity based on ipopt,” *Mathematical Programming Computation*, pp. 1–25, 2012.

[63] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “Casadi: a software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[64] V. M. Zavala and L. T. Biegler, “The advanced-step nmpc controller: Optimality, stability and robustness,” *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[65] Z. J. Yu and L. T. Biegler, “Advanced-step multistage nonlinear model predictive control: Robustness and stability,” *Journal of Process Control*, vol. 84, pp. 192–206, 2019.

[66] Z. J. Yu and L. T. Biegler, “Sensitivity-assisted robust nonlinear model predictive control with scenario generation,” *IFAC-PapersOnLine*, submitted.

[67] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[68] R. M. Apap and I. E. Grossmann, “Models and computational strategies for multistage stochastic programming under endogenous and exogenous uncertainties,” *Computers & Chemical Engineering*, vol. 103, pp. 233–274, 2017.

[69] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[70] J. A. Paulson and A. Mesbah, “Nonlinear model predictive control with explicit backoffs for stochastic systems under arbitrary uncertainty,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 523–534, 2018.

[71] J. Shi, L. T. Biegler, I. Hamdan, and J. Wassick, “Optimization of grade transitions in polyethylene solution polymerization process under uncertainty,” *Computers & Chemical Engineering*, vol. 95, pp. 260–279, 2016.

[72] D. Mayne, “Nonlinear model predictive control: Challenges and opportunities,” in *Nonlinear model predictive control*, pp. 23–44, Springer, 2000.

[73] L. Magni and R. Scattolini, “Robustness and robust design of mpc for nonlinear discrete-time systems,” in *Assessment and future directions of nonlinear model predictive control*, pp. 239–254, Springer, 2007.

[74] X. Yang, *Advanced-multi-step and economically oriented nonlinear model predictive control*. PhD thesis, figshare, 2015.

[75] D. W. Griffith, *Advances in Nonlinear Model Predictive Control forLarge-Scale Chemical Process Systems*. PhD thesis, Carnegie Mellon University, 2018.

[76] L. Magni, D. M. Raimondo, and R. Scattolini, “Regional input-to-state stability for nonlinear model predictive control,” *IEEE Transactions on Automatic Control*, vol. 51, no. 9, pp. 1548–1553, 2006.

[77] W.-H. Chen, D. J. Ballance, and J. O’Reilly, “Model predictive control of nonlinear systems: Computational burden and stability,” *IEE Proceedings-Control Theory and Applications*, vol. 147, no. 4, pp. 387–394, 2000.

[78] L. O. Santos, P. A. Afonso, J. A. Castro, N. M. Oliveira, and L. T. Biegler, “On-line implementation of nonlinear mpc: an experimental case study,” *Control Engineering Practice*, vol. 9, no. 8, pp. 847–857, 2001.

[79] M. Maiworm, T. Bäthge, and R. Findeisen, “Scenario-based model predictive control: Recursive feasibility and stability,” *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 50–56, 2015.

[80] D. M. Raimondo, D. Limon, M. Lazar, L. Magni, and E. F. ndez Camacho, “Min-max model predictive control of nonlinear systems: A unifying overview on stability,” *European Journal of Control*, vol. 15, no. 1, pp. 5–21, 2009.

[81] Y. Zhu, S. Legg, and C. D. Laird, “Optimal operation of cryogenic air separation systems with demand uncertainty and contractual obligations,” *Chemical Engineering Science*, vol. 66, no. 5, pp. 953–963, 2011.

[82] D. P. Word, J. Kang, J. Akesson, and C. D. Laird, “Efficient parallel solution of large-scale nonlinear dynamic optimization problems,” *Computational Optimization and Applications*, vol. 59, no. 3, pp. 667–688, 2014.

[83] A. HSL, “collection of fortran codes for large-scale scientific computation,” *See http://www. hsl. rl. ac. uk*, 2007.

[84] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo-optimization modeling in python*, vol. 67. Springer, 2017.

[85] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl, “acados: a modular open-source framework for fast embedded optimal control,” *arXiv preprint arXiv:1910.13753*, 2019.

[86] H. Chen and F. Allgöwer, “A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability,” *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[87] J. R. Birge, “State-of-the-art-surveystochastic programming: Computation and applications,” *INFORMS journal on computing*, vol. 9, no. 2, pp. 111–133, 1997.

[88] N. V. Sahinidis, “Optimization under uncertainty: state-of-the-art and opportunities,” *Computers & Chemical Engineering*, vol. 28, no. 6-7, pp. 971–983, 2004.

[89] T. W. Jonsbråten, “Optimization models for petroleum field exploitation,” 1998.

[90] V. Goel and I. E. Grossmann, “A class of stochastic programs with decision dependent uncertainty,” *Mathematical programming*, vol. 108, no. 2-3, pp. 355–394, 2006.

[91] F. Di Palma and L. Magni, “A multi-model structure for model predictive control,” *Annual Reviews in Control*, vol. 28, no. 1, pp. 47–52, 2004.