Development of Efficient Algorithms for Model Predictive Control of Fast Systems

Dissertation

Zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt der Fakultät für Informatik und Automatisierung
der Technischen Universität Ilmenau

von Msc. Jasem Aburajabaltamimi
geboren am 02.01.1979 in Hebron-Palästina

1. Gutachter: Prof. Dr.-Ing. habil. P. Li
2. Gutachter: Prof. Dr.-Ing. habil. Ch. Ament
3. Gutachter: Prof. Dr.-Ing. R. Findeisen

Tag der Einreichung: 10.01.2011
Tag der wissenschaftlichen Aussprache: 30.03.2011

urn:nbn:de:gbv:ilm1-2011000068

Abstract

Nonlinear model predictive control (NMPC) has been considered as a promising control algorithm which is based on a real-time solution of a nonlinear dynamic optimization problem. Nonlinear model equations and controls as well as state restrictions are treated as equality and inequality constraints of the optimal control problem. However, NMPC has been applied mostly in relatively slow processes until now, due to its high computational expense. Therefore, computation time needed for the solution of NMPC leads to a bottleneck in its application to fast systems such as mechanical and/or electrical processes.

In this dissertation, a new solution strategy to efficiently solve NMPC problems is proposed so that it can be applied to fast systems. This strategy combines the multiple shooting method with the collocation on finite elements method. The multiple shooting method is used for transforming the nonlinear optimal control problem into nonlinear program (NLP) problem using discretization and parametrization techniques. To solve this NLP problem the values of state variables and their gradients at the end of each shooting need to be computed. We use collocation on finite elements to carry out this task, thus, a high precision approximation of the state variables and their sensitivities in each shoot are achieved. As a result, the advantages of both the multiple shooting and the collocation method can be employed and therefore the computation efficiency can be considerably enhanced.

Due to the nonlinear and complex optimal control problem formulation, in general, it is difficult to analyze the stability properties of NMPC systems. In this dissertation we propose a new formulation of the optimal control problem to ensure the stability of the NMPC problems. It consists the following three features. First, we introduce auxiliary states and linear state equations into the finite horizon dynamic optimization problem. Second, we enforce system states to be contracted with respect to the auxiliary state variables by adding inequality constraints. Thus, the stability features of the system states will conform to the stability properties of the auxiliary states, i.e. the system states will be stable, if the auxiliary states are stable. Third, the eigenvalues of the linear state equations introduced will be determined to stabilize the auxiliary states and at the same time make the optimal control problem feasible. This is achieved by considering the eigenvalues as optimization variables in the optimal control problem. Moreover, features of this formulation are analyzed at the stationary point of the system model.

To show the effectiveness and performance of the proposed algorithm and the new optimal control problem formulation we present a set of NMPC case studies. We use the numerical algorithm group (NAG) library Mark 8 to solve numerically linear and nonlinear systems that resulted from the collocation on finite elements to compute the states and sensitivities, in addition, the interior point optimizer (IPOPT) and in C/C++ environment. Furthermore, to show more applicability, the proposed algorithm is applied to control a laboratory loading bridge.

Keywords

Optimal Control Problem, Nonlinear Model Predictive Control, Direct Methods, Indirect Methods, Direct Multiple Shooting, Collocation on Finite Elements, Sequential Quadratic Programming, Interior Point Method, Newton Methods, Sensitivity Analysis, Prediction Horizon, Control Horizon, Optimal Feedback Control, Stability Analysis.

Zusammenfassung

Die nichtlineare modellprädiktive Regelung (NMPC) ist ein vielversprechender Regelungsalgorithmus, der auf der Echtzeitlösung eines nichtlinearen dynamischen Optimierungsproblems basiert. Nichtlineare Modellgleichungen wie auch die Steuerungs- und Zustandsbeschränkungen werden als Gleichungs- bzw. Ungleichungsbeschränkungen des Optimalsteuerungsproblems behandelt. Jedoch wurde die NMPC wegen des recht hohen Rechenaufwandes bisher meist auf relativ langsame Prozesse angewendet. Daher bildet die Rechenzeit bei Anwendung der NMPC auf schnelle Prozesse einen gewissen Engpass wie z. B. bei mechanischen und/oder elektrischen Prozessen.

In dieser Arbeit wird eine neue Lösungsstrategie für dynamische Optimierungsprobleme vorgeschlagen, wie sie in NMPC auftreten, die auch auf sog. schnelle Systeme anwendbar ist. Diese Strategie kombiniert Mehrschieß-Verfahrens mit der Methode der Kollokation auf finiten Elementen. Mittels Mehrschieß-Verfahren wird das nichtlineare dynamische Optimierungsproblem in ein hochdimensionales statisches Optimierungsproblem (nonlinear program problem, NLP) überführt, wobei Diskretisierungs- und Parametrisierungstechniken zum Einsatz kommen. Um das NLP-Problem zu lösen, müssen die Zustandswerte und ihre Gradienten am Ende jedes Diskretisierung-Intervalles berechnet werden. In dieser Arbeit wird die Methode der Kollokation auf finiten Elementen benutzt, um diese Aufgabe zu lösen. Dadurch lassen sich die Zustandsgrößen und ihre Gradienten am Ende jedes Diskretisierungs-Intervalls auch mit großer Genauigkeit berechnen. Im Ergebnis können die Vorteile beider Methoden (Mehrschieß-Verfahren und Kollokations-Methoden) ausgenutzt werden und die Rechenzeit lässt sich deutlich reduzieren.

Wegen des komplexen Optimierungsproblems ist es im Allgemeinen schwierig, eine Stabilitätsanalyse für das zugehörige NMPC durchzuführen. In dieser Arbeit wird eine neue Formulierung des Optimalsteuerungsproblems vorgeschlagen, durch die die Stabilität des NMPC gesichert werden kann. Diese Strategie besteht aus den folgenden drei Eigenschaften. Zunächst wird ein Hilfszustand über eine lineare Zustandsgleichung in das Optimierungsproblem eingeführt. Die Zustandsgleichungen werden durch Hilfszustände ergänzt, die man in Form von Ungleichungsnebenbedingungen einführt. Wenn die Hilfszustände stabil sind, lässt sich damit die Stabilität des Gesamtsystems sichern. Die Eigenwerte der Hilfszustände werden so gewählt, dass das Optimalsteuerungsproblem lösbar ist. Dazu benutzt man die Eigenwerte als Optimierungsvariable. Damit lassen sich die Stabilitätseigenschaften in einem stationären Punkt des Systemmodells untersuchen.

Leistungsfähigkeit und Effektivität des vorgeschlagenen Algorithmus werden an Hand von Fallstudien belegt. Die Bibliothek Numerische Algorithmus Group (NAG), Mark 8, wird eingesetzt, um die linearen und nichtlinearen Gleichungen, die aus der Kollokation resultieren, zu lösen. Weiterhin wird zur Lösung des NLP-Problems der Löser IPOPT für C/C++- Umgebung eingesetzt. Insbesondere wird der vorgeschlagene Algorithmus zur Steuerung einer Verladebrücke im Labor des Institutes für Automatisierungs- und Systemtechnik angewendet.

Acknowledgments

I would like to express my deep gratitude to my supervisor Professor Pu Li from Ilmeau University of Technology for his invaluable supervision, guidance as well as many discussions and continuous encouragements during my PhD work.

I would like to thank Professor Christoph Ament from Ilmeau University of Technology for his comments on my thesis.

Also I would like to thank Professor Rolf Findeisen from Otto-von-Guericke University-Magdeburg for many comments and suggestion to improve my thesis.

Many thanks to my friends and colleagues especially from Simulation and Optimal Processes group at Ilmeau University of Technology, namely, Prof. Horst Puta, Dr. Siegbert Hopfgarten, Dr. Abebe Geletu, Mr. Stefan Röll, Mr. Martin Bartl, Mr. Quoc Dong Vu, Mr. Hui Zhang, Mr. Ines Mynttinen, Mr. Aouss Gabash, Mr. Michael Klöppel, Mr. Duc Dai Pham, Mr. Wolfgang Heß and Mrs. Rita Helm for making my stay at TU-Ilmenau a wonderful experience.

I am gratefully acknowledge the financial support of Deutscher Akademischer Austausch Dienst (DAAD).

Also many thanks to my parents, brothers, sister and wife for their love, support and continues encouragements.

Finally, I would like to thank Ilmeau University of Technology and Palestine Polytechnic University for giving me the opportunity to complete this work at TU Ilmenau.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Goals

Dynamic optimization problems, also called optimal control problems, are found in many industrial fields and many control frameworks. The aim is to find a control strategy which minimizes a given cost functional (performance measure) and, at the same time, satisfies a set of constraints. These constraints can be system dynamics and/or constraints on states and controls. Besides in process engineering, optimal control has also been applied successfully in many other disciplines such as economics, management etc.

For simple systems, an optimal control problem can be directly solved and the resulting optimal control law applied on-line when the states are continuously measured and feeded back to the system. If the process dynamics do not change and there is no disturbance, the derivation of the controller can be done in advance. In the case of changing dynamics or disturbances an adaptive control has to be employed, then the computation has to be carried out at every sampling time. However, for many systems, the optimal control law has to be computed off-line due to high computation time, which means that the optimal control problem is solved before the real process operations begin.

In general, except for the simplest cases where the problem can be solved analytically, the solution of an optimal control problem in both on-line and off-line cases is treated numerically by using an optimization method. Simple optimal control problems were solved previously by the so-called *indirect* approach which is based on the first order optimality conditions [156]. Within the indirect methods, the optimal control problem can be solved through several approaches such as slack variable method, interior point method and penalty function method.

In the penalty function method, for example, inequality constraints will be converted to equality constraints and the optimal control problem is transformed into a two-point-boundary-value problem. All indirect methods are grouped within a strategy of *"first optimize then discretize"*. Therefore, most of indirect methods are based on finding the solution that satisfies either the Hamiltonian-Jacobi- Bellman equation or the Euler-Lagrange equation and then treat these equations numerically [33, 132]. However, indirect methods have several disadvantages: a) it is very difficult to solve the Hamiltonian-Jacobi-Bellman equation, b) we need to introduce and treat so-called costate variables, c) the solution of the optimal control problem is not robust, and d) we have to have a deep insight into a physical and mathematical nature of the optimal control problem [93].

Since 1980s, many researchers have proposed more efficient methods and algorithms to solve optimal control problems which follow the strategy of *"first discretize and then optimize"* to overcome the drawbacks of the indirect methods. In these methods the solution can be achieved by reformulating "or approximating" the dynamic optimal control problem to a finite dimensional nonlinear programming (NLP) problem. The reformulation can be done by using discretization and parametrization techniques such as control parametriza-

tion, control and state parametrization. After the discretization or parametrization the resulting nonlinear optimization problem can be solved by a nonlinear programming algorithm like sequential quadratic programming [21]. In this way, highly nonlinear complex optimal control problems with inequality and equality constraints can be treated.

To solve nonlinear optimal control problems using a direct method, the control trajectory will be parameterized, while the state trajectory can be handled with two approaches: sequential or simultaneous approach. In the sequential approach, the state vector is handled implicitly with the control vector and initial value vector, and the ordinary differential equations (ODEs) is addressed as an initial value problem using an ODE solver like the Runge-Kutta or Euler algorithm. Thus, the simulation and optimization will be sequentially handled in each iteration of the NLP solver. The degree of freedom of the NLP problem is only composed of the discretized control parameters. The direct single shooting method is an example of the sequential method. In the simultaneous approach, the state trajectory will be parameterized, too. All of the parameterized variables (both states and controls) are considered as optimization variables in the NLP problem. The discretization method commonly used is either collocation on finite elements [18] or multiple shooting [26].

One of the most popular model-based control techniques is *model predictive control* (MPC). The idea of this control technique is to determine the control law by solving online a (nonlinear) optimal control problem on the prediction horizon. The results from this solution are the optimal control profiles and the corresponding state trajectories (predicted states). Only the first part of the optimal controls will be fed to the plant. At the next sampling time the system states are measured or observed and sent to the optimal control solver. These new measurements act as initial conditions to the dynamic system and thus enable the optimal control problem to be solved on the new prediction horizon and the procedure will be repeated. Linear model predictive control (LMPC) is based on a linear model as well as linear constraints on states and controls which nowadays has been widely applied in process industries. On the other hand, more accurate nonlinear models can be expected to represent system dynamics more accurately. Therefore, nonlinear model predictive control (NMPC) is needed in situations where the system behavior is strongly nonlinear.

MPC presents a set of advantages over other control strategies since it can robustly handle multivariable control problems with inequality constraints both on control and state variables. However, the computation of MPC is more complex than classical controllers due to demanding on-line computational requirements of the repetitive solution of the optimal control problem. Thus MPC is often only be applied to "slow" systems with a large sampling rate since the required time to solve the optimization problem is limited. Therefore, the implementation of MPC or NMPC to a system with fast dynamics has not been widespread. For systems with very small time constants, the MPC implementation needs a small sampling time and this costs prohibitive computing expense although much efforts have been made in pervious studies for enhancing the computation efficiency [69, 121, 182]. The difficulty to maintain the balance between the computational cost and optimal performance is one of the main reasons why MPC did not do far replace classical controllers [68, 148]. However, MPC has more advantages than classical controllers when a certain kind of optimality and constraint satisfaction are required. In this case it is desired to solve the optimal control problem as fast as possible and, at the same time, it can be easily implemented on existing or simple micro-controller hardware [138, 148].

The importance to enhance computational performance of MPC can be shown by the control of a five-link robot arm [62]. It is an optimal control problem that minimizes the time for a moving robot arm from a point to another. The minimum time is 0.15 second by solving this optimal control problem using one of the available numerical algorithms. But it takes 20 CPU seconds to run the computation using a fast computer. According to this example, it is clear that the required time to calculate the optimal controller is much more than the actual time, it means such MPC algorithms are not applicable for real time application. Therefore it is necessary to develop new MPC algorithms for fast systems.

Furthermore, due to the nonlinearities and complexities of system dynamics and constraints, it is very difficult to analyze the stability of closed-loop systems by using MPC. Therefore, the stability analysis of MPC systems has long been a challenge. Many schemes to ensure the closed-loop stability of the MPC can be found in the literature such as stabilizing the MPC system by using a terminal equality constraint, a terminal cost function etc [124]. These stabilization methods have some limitations or shortcomings, e.g., it is very difficult to define a proper Lyapunov function.

In summary, although many progresses have been made in the development of MPC algorithms in previous studies, furtherer investigations are needed especially in the areas of efficient computing and stabilization analysis. These are the major motivations and goals of this thesis.

1.2 Thesis Contribution

The contribution of this thesis can be briefly summarized as follows:

- A new approach to the solution of optimal control problems resulting from NMPC problems is proposed. This approach is based on the principle of direct methods to solve the nonlinear optimal control problem by using a combination of multiple shooting with collocation on finite elements. We first use multiple shooting to convert the nonlinear optimal control problem into a nonlinear programming (NLP) problem. The finite time horizon will be discretized into subintervals and then the controls in each subinterval will be parameterized. At the same time the initial values of the states at the beginning of each subinterval will be parameterized as well. Then collocation on finite elements is used for the integration of the model equations and the computation of the gradients required. Due to this combinations, the proposed control strategy possesses a higher computation efficiency; it requires a smaller amount of computation expense compared with the existing NMPC algorithms.

  The main contribution of the proposed approach to the existing multiple shooting approaches is the employment of collocation on finite elements to calculate state values and its derivatives instead of using a standard ODE solver and internal numerical differentiation, respectively. Moreover, in comparison to a pure collocation method, using multiple shooting the size of the resulted NLP problem is much smaller. Furthermore, since in multiple shooting the computation of each shoot is independent, this makes a parallel computation possible. Therefore, the increase in the number of shoots (i.e. finite elements) will not lead to more computation expense if a parallel computation is implemented.

- We propose a new approach to ensure the stability of NMPC systems by introducing auxiliary state variables and corresponding linear state equations. It leads to a new formulation of NMPC problems, in which a new term is added to the performance index to penalize the auxiliary states. System states are enforced to contract with respect to the auxiliary state variables by adding inequality constraints. Thus the stability properties of system states will conform to those of the auxiliary states, i.e. the system states will be stable, if the auxiliary states are stable. The eigenvalues of the linear state equations introduced will be determined to stabilize the auxiliary state variables and at the same time make the optimal control problem feasible. This is achieved by considering the eigenvalues as optimization variables in the optimal control problem. Therefore, the solution of the optimal control problem guarantees the feasibility, stability and optimality of the NMPC system.

  We analyze the features of this new formulation of NMPC at the terminal region and equilibrium point. It can be proven that if the prediction horizon is chosen such that the system and auxiliary states reach a neighborhood of the equilibrium point, the optimal eigenvector and auxiliary states can shift instable poles of the linearized system to a stable region.

- The efficiency of the proposed approach is demonstrated through several case studies. The computation time taken to solve these control problems is in the order of mseconds. Therefore this NMPC algorithm can be applied to fast systems. In addition it is successfully implemented to control a laboratory loading bridge. Satisfactory control performance of a moving cart with a single pendulum between two points with highly nonlinear dynamics is achieved.

1.3 Thesis Structure

The remaining chapters of this thesis are organized as follows:

Chapter 2 gives an overview of the optimal control theory. It reviews some analytical and numerical methods to solve optimal control problems of continuous time system. A nonlinear optimal control problem is formulated in general form and methods that are used to solve this problem are reviewed. These methods are classified into three groups: dynamic programming, indirect methods and direct methods. We review the basic procedures of dynamic programming which converts an optimal control problem into a Hamilton-Jacobi-Bellman (HJB) equation. Indirect methods are based on the first order optimality conditions of variations of the infinite optimal control problem and the solution of an optimal control problem will be transformed to a solution of a two-point-boundary-value problem which can be solved numerically by several methods: gradient methods, shooting methods, segmentation methods. We also review the direct methods and classify them into two groups: sequential method, like in direct single shooting method, and simultaneous method like in collocation on finite elements method. Finally, we introduce the principle and the structure of the MPC.

Chapter 3 reviews the direct multiple shooting method and the parameterized nonlinear programming problem that will be treated in this thesis. We review the basic theory

of the optimality conditions for a constrained and parameterized NLP problem. The sequential quadratic programming (SQP) method as well as the quadratic programming (QP) methods are described. We also present other methods to solve the NLP problems, such as, sparse nonlinear optimizer (SNOPT) and interior point optimizer (IPOPT).

Chapter 4 presents some approaches that are used to solve differential algebraic equations (DAEs) that are normally addressed as initial value problems (IVPs). We review the Euler method, Runge-Kutta method and collocation on finite elements approach which will be used to solve the IVPs of the parameterized NLP problem. An error analysis of the reviewed DAE solvers will be also given and the additional efficiency of the collocation on finite elements solver will be provided.

Chapter 5 presents the main algorithm in this thesis. It presents the main procedures that are needed to solve the parameterized NLP. We call this method *combined multiple shooting collocation method*. The elementary steps that lead to the solutions of the DAEs, in the parameterized NLP, using collocation on finite elements will be described. In addition, the steps and the equations that lead to the computation of the variables' sensitivities will be also presented. In Section 5.3 we present the proposed algorithm in pseudocode.

Chapter 6 focuses on stability analysis of NMPC. We review the receding horizon control principle as well as Lyapunov stability theorems. Section 6.3 reviews some schemes that are used to ensure the stability of NMPC systems. In Section 6.4 we propose a new approach to ensure the stability of NMPC systems. We use a quasi-infinite method to enhance our proposed approach. Moreover, the features of the proposed approach at the equilibrium point is analyzed

Chapter 7 presents some case-studies to show the efficiency of the proposed algorithm. We apply the proposed algorithms to different types of case-studies on-line and off-line. The proposed algorithm is successfully implemented to the NMPC system of a loading bridge unit at Ilmeau University of Technology .

Finally, Chapter 8 contains the conclusions of this thesis and recommendations for future developments.

2 Optimal Control Theory: Review

2.1 Optimal Control Problem of Continuous Systems

Optimal control problems have been studied in many previous works [4, 33, 67, 97, 102, 109, 151]. The main goal of optimal control is to determine an open-loop optimal control $u^*(t)$ or an optimal feedback control $u^*(x^*(t), z^*(t), a^*, t)$ that forces the system to satisfy a set of physical constraints and, at the same time, minimizes an objective functional. This chapter reviews a general class of optimal control problems and existing solution methods on which this dissertation is built.

An optimal control problem consists of an objective functional or performance index, model equations usually described by a set of differential-algebraic equations (DAEs) and path and boundary constraints of state as well as control variables.

2.1.1 Differential Algebraic Equation System

The corner stone of the optimal control problem is a mathematical model of a system. This model is described by state equations and can be written by a set of differential algebraic equations (DAEs) in the explicit form:

$$\begin{aligned} &\dot{x}(t) = f(x(t), z(t), u(t), a, t), \\ &g(x(t), z(t), u(t), a, t) = 0, \end{aligned}$$

where $x(t) \in \mathbb{R}^{n_x}$ and $z(t) \in \mathbb{R}^{n_z}$ denote the differential and algebraic state vectors, respectively, $u(t) \in \mathbb{R}^{n_u}$ and $a \in \mathbb{R}^{n_a}$ are the control vector and a time-invariant system parameters vector, respectively, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_z}$ are continuously differentiable with respect to all its arguments in time interval $t \in [t_0, t_f]$. If the model equations are written in the implicit form, we have the following representation

$$\hat{f}(\dot{x}(t), x(t), z(t), u(t), a, t) = 0, \tag{2.1a}$$

$$\hat{g}(x(t), z(t), u(t), a, t) = 0. \tag{2.1b}$$

In this thesis we consider only nonsingular cases, i.e., the Jacobian of $\hat{f}$ with respect to $\dot{x}$ (denoted by $\partial \hat{f}/\partial \dot{x} = \hat{f}_{\dot{x}}$) is nonsingular. These cases are normally addressed as *index one* DAEs [30].

2.1.2 Objective Functional

An optimal control problem enables the designer to select the "best" control vector by minimizing (or maximizing) a given objective functional (performance index). The general

Table 2.1: Typical objective functionals.

Problem subclass	Lagrange term	Mayer term
Minimum tracking error	$(x(t)-x_d(t))^T Q(x(t)-x_d(t))$ $+(z(t)-z_d(t))^T \hat{Q}(z(t)-z_d(t))$	-
Minimum control effort	$u^T R u$	-
General optimal control	$(x(t)-x_d(t))^T Q(x(t)-x_d(t))$ $+(z(t)-z_d(t))^T \hat{Q}(z(t)-z_d(t))$ $+u(t)^T R u(t)$	-
Minimum terminal	-	$(x(t_f)-x_d(t_f))^T S(x(t_f)-x_d(t_f))$ $+(z(t_f)-z_d(t_f))^T \hat{S}(z(t_f)-z_d(t_f))$
Combined minimization	$(x(t)-x_d(t))^T Q(x(t)-x_d(t))$ $+(z(t)-z_d(t))^T \hat{Q}(z(t)-z_d(t))$ $+u(t)^T R u(t)$	$(x(t_f)-x_d(t_f))^T S(x(t_f)-x_d(t_f))$ $+(z(t_f)-z_d(t_f))^T \hat{S}(z(t_f)-z_d(t_f))$
Minimum fuel	$\sum_{i=1}^{n_u} \beta_i \lvert u_i(t) \rvert$	-
Minimum time	1	-

Bloza type objective functional can be expressed:

$$J(x(t), z(t), u(t), a, t) = E(x(t_f), z(t_f), a, t_f) + \int_{t_0}^{t_f} L(x(t), z(t), u(t), a, t) dt.$$

where E and L are scalar functions, continuously differentiable in all arguments and often called Mayer term and Lagrange term, respectively. The above cost functional is very general and can cover a large class of practical problems in many control applications. Performance indices of important sub-classes of optimal control problems can be found in Table 2.1, where R is a positive definite matrix ($R > 0$), Q, $\hat{Q}$, S and $\hat{S}$ are positive semi-definite matrices ($Q, \hat{Q}, S, \hat{S} \geq 0$) and $\beta_i > 0$ [111, 132].

2.1.3 Path Constraints and Boundary Conditions

The solution of the optimal control problem must satisfy restrictions and boundary conditions on the state and control variables on the time horizon. A general form of path constraints is

$$s(x(t), z(t), u(t), a, t) \geq 0, \qquad t \in [t_0, t_f].$$

where $s : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_s}$.

The typical form of path constraints are lower and upper bounds on states and controls

$$\begin{aligned} x_{\min} &\leq x(t) \leq x_{\max}, \\ z_{\min} &\leq z(t) \leq z_{\max}, \\ u_{\min} &\leq u(t) \leq u_{\max}, \\ a_{\min} &\leq a \leq a_{\max}. \end{aligned}$$

The initial state and terminal equality as well as inequality constraints are considered as boundary conditions of the optimal control problem

$$\begin{aligned} x(t_0) &= x_0, \\ r_e(x(t_f), z(t_f), a, t_f) &= 0, \\ r_i(x(t_f), z(t_f), a, t_f) &\geq 0, \end{aligned}$$

where the vector functions $\{r_e : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_a} \times \mathbb{R} \to \mathbb{R}^{n_{r_e}}\}$ and $\{r_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_a} \times \mathbb{R} \to \mathbb{R}^{n_{r_i}}\}$

2.2 Optimal Control Problem Formulation

According to the above definitions, a general formulation of an optimal control problem starts by minimizing or "maximizing" the objective functional. The controls and the states that minimize an objective functional must satisfy the model equations, the path constraints and the boundary conditions. Therefore, the constrained optimal control problem can be formulated as

$$\min_{u(t),x(t),z(t),a} J = E(x(t_f), z(t_f), a, t_f) + \int_{t_0}^{t_f} L(x(t), z(t), u(t), a, t)dt. \tag{2.2a}$$

subject to

$$\dot{x}(t) = f(x(t), u(t), z(t), a, t), \qquad t \in [t_0, t_f], \tag{2.2b}$$
$$g(x(t), u(t), z(t), a, t) = 0, \qquad t \in [t_0, t_f], \tag{2.2c}$$
$$s(x(t), z(t), u(t), a, t) \geq 0, \qquad t \in [t_0, t_f], \tag{2.2d}$$
$$x(t_0) = x_0, \tag{2.2e}$$
$$r_e(x(t_f), z(t_f), a, t_f) = 0, \tag{2.2f}$$
$$r_i(x(t_f), z(t_f), a, t_f) \geq 0, \tag{2.2g}$$
$$x_{\min} \leq x(t) \leq x_{\max}, \tag{2.2h}$$
$$z_{\min} \leq z(t) \leq z_{\max}, \tag{2.2i}$$
$$u_{\min} \leq u(t) \leq u_{\max}, \tag{2.2j}$$
$$a_{\min} \leq \quad a \quad \leq a_{\max}. \tag{2.2k}$$

If this optimal control problem is solved, we obtain an optimal objective functional J^* with an optimal control $u^*(t)$ and optimal system parameter a^* with corresponding state trajectories $x^*(t)$ and $z^*(t)$ for $t \in [t_0, t_f]$.

2.3 Optimal Control Problem Solution

Solving optimal control problems is highly motivated nowadays, since these solutions are very important in almost all industrial fields such as chemical, electrical, mechanical, and economical systems. Several methods are available to solve optimal control problems[62]:

- Dynamic Programming: The Hamilton-Jacobin-Bellman (HJB) equation and a partial differential equation (PDE) in state space are solved.
- Indirect methods: Characterized as first optimization and then discretization.
- Direct methods: Characterized as first discretization and then optimization.

2.3.1 Dynamic Programming

The principle of dynamic optimization usually is known as *dynamic programming* [15, 17]. The idea of dynamic programming was proposed by Bellman [15]. It converts an optimal control problem into a so-called Hamilton-Jacobi-Bellman (HJB) equation [4, 17, 33, 97]. Here we review the basic procedure in dynamic programming to solve a sub-class of the general optimal control problem defined above. We consider the following optimal control problem

$$\min_{u(t)} J = E(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt. \tag{2.3a}$$

subject to

$$\dot{x}(t) = f(x(t), u(t), t), \qquad t \in [t_0, t_f], \tag{2.3b}$$
$$x(t_0) = x_0, \tag{2.3c}$$
$$r_e(x(t_f), t_f) = 0. \tag{2.3d}$$

The HJB equation is given by

$$\frac{\partial H^*(x(t), t)}{\partial t} = -\min_{u(t)}[L(x(t), u(t), t) + (\frac{\partial H^*}{\partial x})^T f(x(t), u(t), t)]. \tag{2.4}$$

The solution of Eq.(2.4) leads to the control law

$$u^* = \Upsilon(\frac{\partial H^*}{\partial t}, x(t), t) \tag{2.5}$$

Substituting Eq.(2.5) in Eq.(2.4), the following partial differential equation is obtained

$$-\frac{\partial H^*(x(t), t)}{\partial t} = L(x(t), \Upsilon(t), t) + (\frac{\partial H^*}{\partial x})^T f(x(t), \Upsilon(t), t). \tag{2.6}$$

Now the gradient $\frac{\partial H^*}{\partial x}$ will be computed and then substituted in Eq.(2.5), thus we obtain the control law

$$u^* = \hat{\Upsilon}(x(t), t) \tag{2.7}$$

For more detail on derivation of HJB, see for example [4, 17, 33, 97, 132]. The HJB equation solution is a sufficient condition for optimality. In the case of a linear quadratic optimal problem in Eqs. (2.3a)-(2.3d), the HJB equation is reduced to a Riccati differential equation which can be solved analytically [132]. However, in the nonlinear case it is difficult to solve Eq.(2.6) analytically. In addition, we may not be able to easily solve the nonlinear PDE by numerical techniques. Moreover, inequality constraints can not be considered. That is why the research in this direction has not been followed.

2.3.2 Indirect Methods

Indirect methods are based on the first order optimality conditions of variations of the infinite optimal control problem. Inequality constraints will be converted into equality constraints and the optimal control problem will be transformed to a two-point-boundary-value problem (TPBVP) which must be solved numerically [21, 155]. We consider the following constrained optimal control problem

$$\min_{u(t),x(t)} J = E(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt. \tag{2.8a}$$

subject to

$$\dot{x}(t) = f(x(t), u(t), t), \qquad t \in [t_0, t_f], \tag{2.8b}$$
$$s(x(t), u(t), t) \geq 0, \qquad t \in [t_0, t_f], \tag{2.8c}$$
$$x(t_0) = x_0, \tag{2.8d}$$
$$r_e(x(t_f), t_f) = 0. \tag{2.8e}$$

This problem can be solved using indirect methods using three approaches [70, 129]: slack variables, interior point or penalty function methods. In the penalty function method for example,inequality constraints will be converted to equality constraints and the solution of the optimal control problem will be transformed to the solution of a TPBVP which will be solved numerically. Several kinds of numerical methods are based on the Euler-Lagrange differential equation (EL-DEQ), are available to solve the TPBVP. One may classify these numerical methods according to the particular approach used [1]:

Gradient methods

In these methods the optimal control problem is solved by minimizing the Hamiltonian equation subject to the boundary-value problem [33]. In each iteration the state Eq. (2.8b) is numerically integrated forwards in the time while the costate is integrated backwards.

Shooting methods

It is sometimes called *initial value method* and is the commonly used method for the solution of the boundary-value problem [146]. Shooting method tries to determine the unknown initial values of the costate by the given terminal condition. Then the problem can be solved as an initial value problem by using, e.g., the Runge-Kutta method.

Segmentation methods

These methods are applied for solving TPBVP numerically. The time interval $[t_0, t_f]$ is segmented into subintervals and the differential equations are approximated over these segments. There are several segmentation methods according to the size of segments and approximation procedures.

- Segmented initial value method or multiple shooting method: It is one of the most powerful numerical methods for solving TPBVP. The principle of multiple shooting is to discretize the time interval $[t_0, t_f]$ by introducing additional grid points

  $$t_0 < t_1 < ... < t_N = t_f$$

  For the nonlinear-boundary-value problem, the shooting method calculates the values of the states and costates at all grid points using the Newton-Raphson method or by any roots-finding technique. More details of multiple shooting can be found in [6, 7, 34, 35].

- Methods based on piecewise polynomial functions (spline function, Hermite interpolation, collocation on finite elements etc.) The idea of these methods is approximating the differential equations in each segment with these functions.

  The principle of the collocation method is to choose a finite-dimensional space of candidate solutions (polynomials with a certain degree) and a number of points in the time segment (called collocation points), then these solutions which satisfy given equations at collocation points are determined. Collocation has been applied more successfully in direct methods [7, 34, 35, 149]. In this thesis, we apply the collocation method in Chapter 5 to solve ODEs.

- Finite difference methods: For a high numerical accuracy the size of each segment is taken relatively small. The use of uniform segment lengths is preferred in general, although an adaptive segment length is possible.

However, indirect methods have several disadvantages. They require significant efforts to convert the optimal control problem into the TPBVP. For complex systems the derivation of the TPBVP is even not possible. In addition, inequality constraints can not be handled with indirect methods.

2.3.3 Direct Methods

More complicated optimal control problems are normally solved by direct methods which transform the optimal control problem into a nonlinear programming (NLP) problem of the form [18, 28, 87, 101, 114]:

$$\min_{w} A(w) \tag{2.9a}$$

subject to

$$B(w) = 0, \tag{2.9b}$$

$$C(w) \geq 0, \tag{2.9c}$$

where a finite dimensional vector w represents the decision variables of this optimization problem and the functions A, B and C are corresponding the objective function, equality constraints and inequality constraints, respectively, after the transformation. This NLP problem can be solved iteratively by using a sequential quadratic programming (SQP) method. This leads to a numerical solution task, that is why direct methods are regarded as numerical methods. In this way, inequality and equality constraints as well as highly

nonlinear complex optimal control problems can be treated [54, 156]. In all direct methods, discretization and parametrization methods are used for the transformation. Basically two different classifications of direct methods are found in the literature to solve the optimal control problem [19, 21, 75].

Sequential Method

In this approach control variables are discretized over the time horizon and model equations are solved "exactly" in every iteration of the NLP solver by a numerical integration method [94]. That means only control trajectories will be parameterized and then considered as optimization variables. The solution of the model is implicitly performed during the integration of ODEs which are addressed as an initial value problems using one of the available integration methods like Runge-Kutta or Euler algorithms [101, 155]. An example of sequential methods is direct single shooting which will be briefly addressed in Section 2.3.4.

Simultaneous Method

In this method, control as well as state trajectories will be discretized and parameterized over the time horizon $[t_0, t_f]$. Thus the finite dimensional vector w will represent both parameterized states and controls. The ODEs will be transformed into algebraic equality constraints either with collocation on finite elements [20, 111] which is sketched in Section 2.3.5 or with multiple shooting [61] as shown in Chapter 3.

2.3.4 Direct Single Shooting Method

Direct single shooting reformulates the optimal control problem[1] into a finite dimensional NLP problem (2.9) [94, 101]:

$$\min_{u(t),x(t)} J = E(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt. \tag{2.10a}$$

subject to

$$\dot{x}(t) = f(x(t), u(t), t), \qquad t \in [t_0, t_f], \tag{2.10b}$$
$$s(x(t), u(t), t) \geq 0, \qquad t \in [t_0, t_f], \tag{2.10c}$$
$$x(t_0) = x_0, \tag{2.10d}$$
$$r_e(x(t_f), t_f) = 0. \tag{2.10e}$$

First, the time interval $[t_0, t_f]$ is divided into equal subintervals (segments) $[t_i, t_{i+1}]$, such that

$$t_0 < t_1 < ... < t_N = t_f$$

where N is the total number of subintervals. Then the control vector is transformed into a parameterized finite dimensional control vector $u(t, q)$ that depends on the finite dimensional parameter vector $q \in \mathbb{R}^N$. There are several parametrization schemes [21], for

[1] Formulation (2.8) is repeated here for duality of presentation.

example Fig 2.1 shows how the piecewise constant representation is used to parameterize a control variable. As a sequential method a numerical simulation routine is used for solving

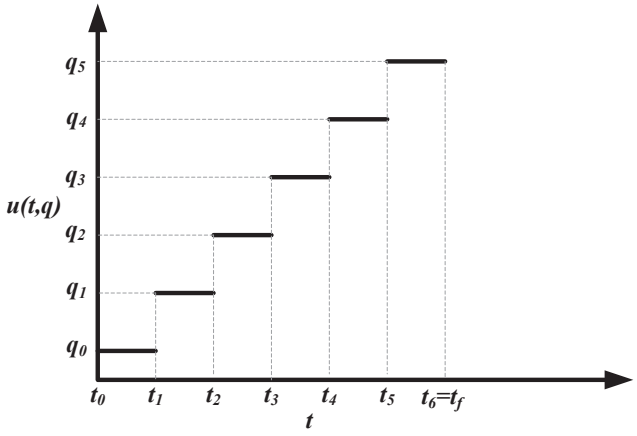

Figure 2.1: Piecewise constant parametrization of a control ($N = 6$). Control intervals are given as $[t_i, t_{i+1}]$ for $i = 0, ..., 5$ with intermediate time points $t_1, ..., t_5$.

the initial value problem (IVP)

$$\dot{x}(t) = f(x(t), u(t, q), t), \qquad x(t_0) = x_0, \qquad t \in [t_0, t_f],$$

which is solved to yield the state vector $x(t, q)$ in the time interval $[t_0, t_f]$ as shown in Fig 2.2. It means the state vector depends on parametrization of the control vector. Since the simulation is done over the whole time horizon this method is called direct single shooting method. Due to the numerical simulation the model equations are eliminated from (2.10). The path constraints are also discretized. Thus the optimal control problem Eq. (2.10) is rewritten as

$$\min_q J = E(x(t_f), q) + \int_{t_0}^{t_f} L(x(t, q), u(t, q), t)dt, \tag{2.11a}$$

subject to

$$s(x(t_i, q), u(t_i, q)) \geq 0, \qquad t \in [t_0, t_f], \tag{2.11b}$$

$$r_e(x(t_f), q) = 0. \tag{2.11c}$$

Fig 2.3 shows the framework of single shooting method. Here the simulation phase is done at every iteration of the NLP solver. After the convergence of the NLP algorithm the optimal control vector is obtained from the optimizer, while the corresponding optimal state vector is obtained from the simulator.

It can be seen that the single shooting method depends mainly on solutions of DAEs. That means, the solution accuracy depends on the accuracy of DAE solver and only a 'suitable' initial guess for the control vector is needed for the optimization. On the other hand, since the number of optimization variables is only determined by the parameterized control vector the size of the NLP problem will be small even when we deal with a large DAE system. This method, however, has disadvantages in comparison to other direct methods. The knowledge of the initialization of state trajectories cannot be used, especially

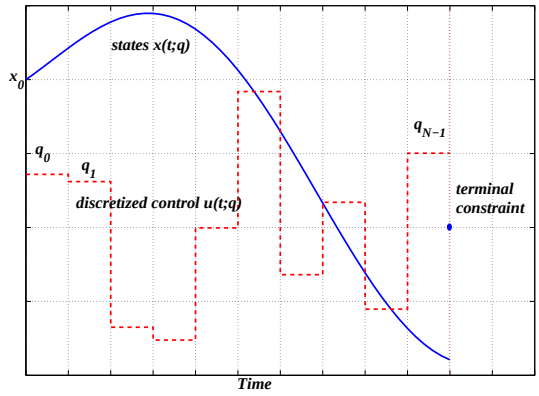

Figure 2.2: Illustration of single shooting.

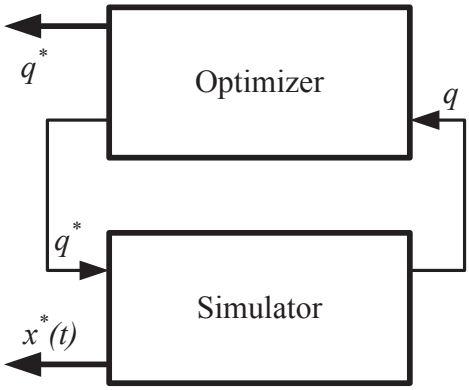

Figure 2.3: Framework of single shooting method.

in tracking problems. It has difficulties to treat instable systems. The information of the violation of state trajectories cannot be obtained. In addition, this method can lead to DAEs which depend on a nonlinear parameter q.

2.3.5 Collocation Method

In the collocation method both control and state trajectories will be discretized and parameterized [18, 38, 52, 78, 89, 177]. We consider the optimal control problem (2.10). The control vector is normally parameterized with piecewise constant vector q_i in each subinterval $[t_i, t_{i+1}]$. The state trajectories are interpolated with intermediate collocation points within the subinterval $[t_i, t_{i+1}]$. Therefore the size of the NLP problem will be the parameter vector q, values of the state vector at the grid points w_i (i.e. parameterized states at the grid points) as well as additional variables $\hat{w}_i$ which represent the states at the collocation points in each subinterval.

Fig 2.4 shows the framework of the collocation method. Using collocation the continuous system model is transferred into equality constraints. In this simultaneous treatment, equality constraints are not necessarily satisfied in each NLP iteration. When the NLP

solver converges, the equality constraints will be satisfied and both optimal control trajectories and corresponding optimal state trajectories are obtained from the optimizer.

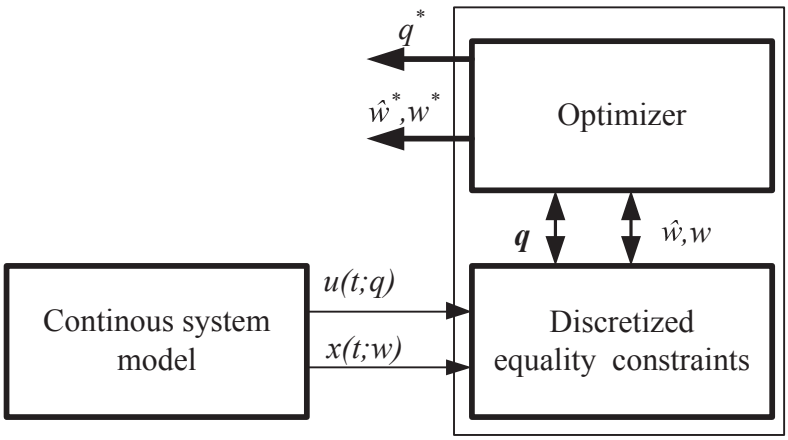

Figure 2.4: Framework of the collocation method.

In the collocation method, the system model

$$\dot{x}(t) = f(x(t), u(t), t), \qquad t \in [t_0, t_f].$$

is replaced by finite equality constraints

$$C_i(q_i, w_i, \hat{w}_i, w_{i+1}) = 0, \qquad i = 0, ..., N-1.$$

Similarly, the objective functional is discretized and approximated in each subinterval

$$L_i(q_i, w_i, \hat{w}_i, w_{i+1})$$

Then we obtain the following large-scale and sparse NLP problem[51]:

$$\min_{q,w,\hat{w}} \quad E(w_N) + \sum_{i=0}^{N-1} L_i(q_i, w_i, \hat{w}_i, w_{i+1}) \tag{2.12a}$$

subject to

$$C_i(q_i, w_i, \hat{w}_i, w_{i+1}) = 0, \qquad i = 0, ..., N-1, \tag{2.12b}$$
$$s(q_i, w_i, \hat{w}_i, w_{i+1}) \geq 0, \qquad i = 0, ..., N-1, \tag{2.12c}$$
$$w_0 = x_0, \tag{2.12d}$$
$$r_e(w_N) = 0. \tag{2.12e}$$

Since the NLP problem obtained by the collocation method is very sparse, this structure can be utilized to reduce the computation time. The main advantages of the collocation method in comparison to single shooting method are that it uses the information of state trajectories in the initialization. In addition, an asymptotical stability can be obtained since it can easily handel path and terminal constraints on state variables.

However, due to the simultaneous treatment of both controls and states the dimension of the NLP problem can be too high to be solved efficiently. Moreover the dimension of

the NLP problem will be further increased if the discretization error needs to be updated. Therefore the collocation approach is normally not applied to optimal control problems, even it is applied to many practical optimal control problems off-line [38, 160, 166].

In this thesis, we use collocation on finite elements to approximate state trajectories in each subinterval which [91] is combined with direct multiple shooting briefly discussed in the following.

2.3.6 Direct Multiple Shooting

This method combines the advantages of simultaneous methods like collocation method with the main advantages of the single shooting method, so that it is sometimes called a *hybrid method* [65]. In this method the transcription of the optimal control problem (2.10) into a NLP problem starts with way similar to the single shooting method. The time interval $[t_0, t_f]$ is divided into N equal subintervals [28, 64, 156]. Then the control trajectory vector is transformed into a parameterized finite dimensional control vector $u(t, q)$ that depends on the finite dimensional vector $q \in \mathbb{R}^N$. In addition, state trajectories are discretized by the same grid points. The initial values of state trajectories in each subinterval are also parameterized. Through this discritization, the continuous model equations are transformed into discritized equality constraints, where the terminal value of state in each subinterval is equal the initial value of the state in next subinterval. Therefore, the equality constraints are needed to be solved for the computation of state variables on the grid points in each NLP iteration. Fig. 2.5 shows the framework of the direct multiple shooting method. The difference from the collocation method is that we need only the control and states at the grid points as optimization variables in the NLP formulation. In Chapter 3, we will discuses the direct multiple shooting method in detail.

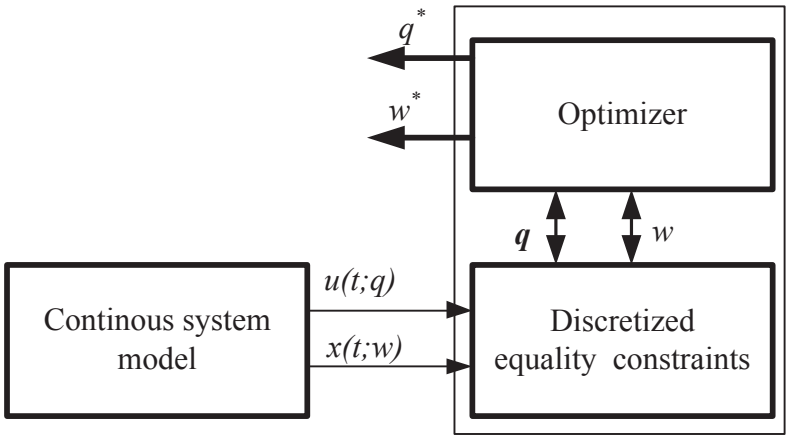

Figure 2.5: Framework of multiple shooting method.

Fig. 2.6 summarizes some direct and indirect methods discussed above for solving optimal control problems.

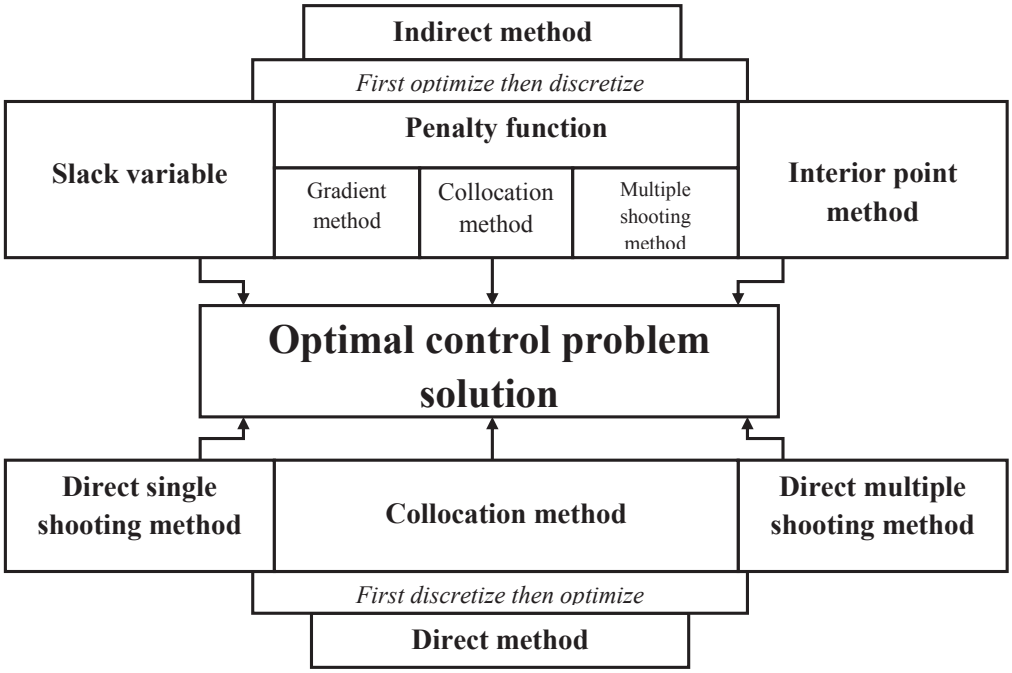

Figure 2.6: Computation methods of optimal control problems using direct and indirect approaches.

2.4 Model Predictive Control (MPC)

The principle of model predictive control (MPC) is to create a formulation that solves online the finite optimal control problem subject to model equations and constrains involving states and controls [13, 46, 76, 99, 115, 116, 122, 128, 140, 142, 169] as described, e.g., by Eq. (2.2).

The model is used to predict the process output at a future time horizon by finding a control sequence that minimizes a certain objective functional using the theory of the optimal control [36]. We assume that the system is described by a set of index one DAEs (2.2b) with initial condition the $x(t_0) = x_0$. Here we define the *prediction* horizon over which the optimal control problem is solved ($T_p = t_f - t_0$). Using MPC, first the optimal control $u(t)$ is applied to the plant over the control horizon ($T_c < T_p$). If there are no disturbances and no model-plant mismatch then we can apply the input $u(t)$ over the time horizon $t \in [t_0, t_f]$. But when disturbances are presented and model-plant mismatches, the computed control strategy will lead to poor performance. Therefore, a feedback mechanism will be implemented, that means, the open loop computed input function will be implemented only until the next measurement is available. Fig. 2.7 shows the basic principle of model predictive control, Fig. 2.8 shows the basic structure of MPC, as applied in closed-loop. We use the system model to predict the future plant outputs, based on past and current values and optimal future controls as well as the system constraints. Accordingly, the MPC methodology can be summarized:

1. Measure and/or estimate state variables at time instant t.

2. Compute an optimal control vector by solving an open-loop optimal control problem

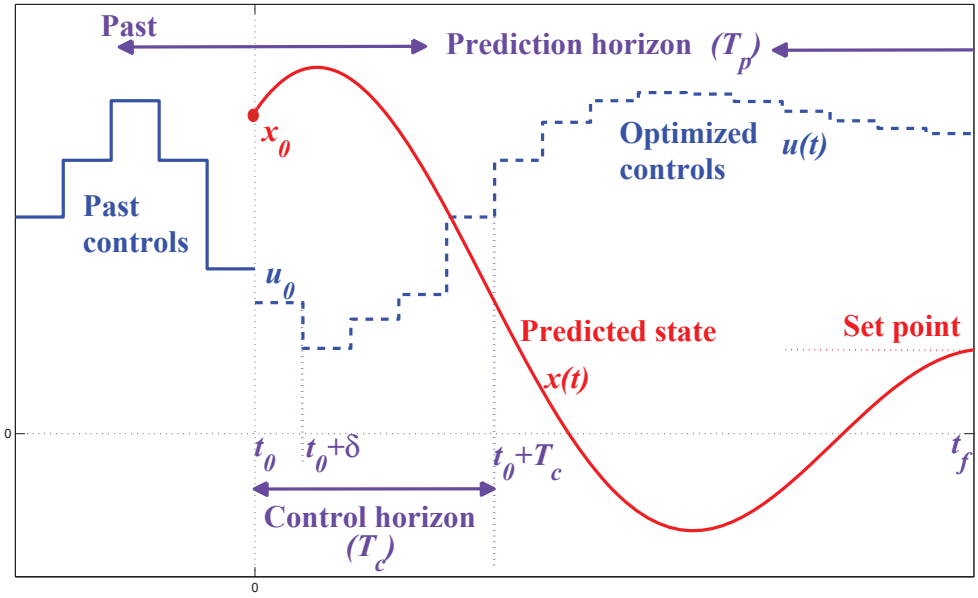

Figure 2.7: Principle of model predictive control (MPC).

over a future prediction horizon T_p subject to model equations, constrains on states and controls based on measured and/or estimated state variables.

3. Apply the first part of the computed optimal controls until the new state measurements and/or estimations are available.

4. Continue with step 1.

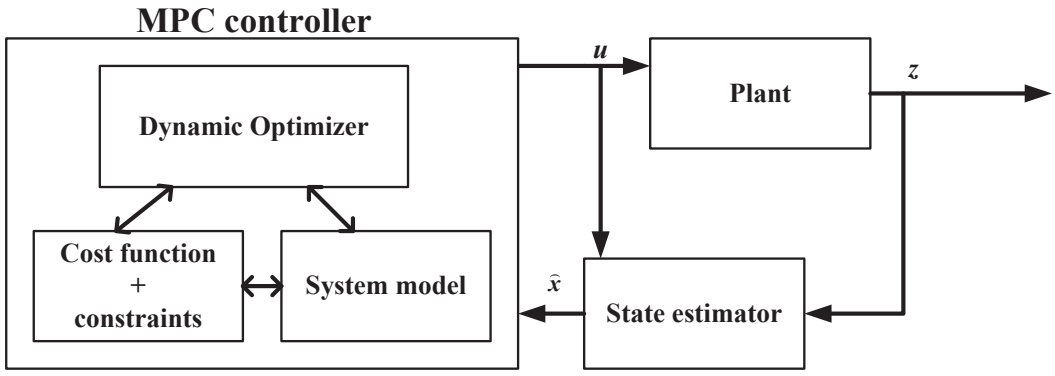

Figure 2.8: Basic MPC control loop.

MPC (also called *long-rang Predictive Control* (LRPC) [36]) has been considered as a promising control algorithm in recent years. Many different algorithms have been presented in the literature to discuss MPC. MPC has been used in many complex applications such as control of robot manipulators [108, 175], clinical anaesthesia [112], cement plant [49], chemical plants [16, 48, 50] and steam generators [144, 145].

Moreover, MPC is very attractive especially for plant operators with limited knowledge of control since MPC concepts are easy to understand and tuned. The advantages of MPC

can be summarized as follows: 1) It can be used to control different complex processes, even systems that include long time delays as well as systems that are instable. 2) It solves the open-loop optimal control problem on-line. 3) It can handle linear and nonlinear systems. 4) It can compensate dead time. 5) It introduces feed forward control in a natural way to compensate for the measurable disturbances.

However, MPC has also disadvantages. The derivation of the resulting control law is more complex than that of classical controllers (e.g. PID controller). If the process dynamics do not change, the derivation of the controller can be done in advance. But in the case of an adaptive control all the computations have to be carried out at every sample time. When the inequality constraints are considered, an iterative solution scheme has to be used and then the expense of the required computation is even higher. Thus MPC has been only implemented to 'slow' dynamic systems and for 'large sampling rates' [68] despite the advances of the computational tools.

2.5 Nonlinear Model Predictive Control (NMPC)

Linear MPC is an efficient algorithm which handles intrinsic multivariable and constraints. In many cases the selection of the desired operating range coupled with nonlinear process behaviors can degrade the process performance and destabilize the closed loop system. Nonlinear MPC (NMPC) can relieve this performance degradation.

The concept of NMPC is similar to linear MPC where the control problem is formulated to solve on-line a finite open-loop optimal control problem considering *nonlinear* model equations of the system and possibly nonlinear constraints on controls and states. However, an infinite horizons problem is very difficult in the nonlinear case, so that a *moving horizon* technique is used instead. That means a constant control horizon T_C is chosen. If the control horizon T_C is sufficiently large, the computed optimal control, state and output variables are supposed to have a 'good' approximation of the exact solutions for the infinite horizon problem [123].

Several theoretical and computational aspects of NMPC were discussed in the literature. In Chapter 6 we will review NMPC approaches that guarantee the closed-loop stability and propose an approach to stabilize the NMPC systems.

2.6 Summary

In this chapter the basic optimal control problem is formulated. To the solution of the optimal control problem two important strategies (direct and indirect) are reviewed. Within indirect method, the optimal control problem can be solved by slack variable method, penalty function method or interior point method. Within direct methods the problem will be transformed in nonlinear programming (NLP) problem using direct single shooting problem, collocation method or direct multiple shooting, then NLP problem will be solved by using a sequential quadratic programming (SQP) method. Moreover, the basic principle of (nonlinear) model predictive control (MPC) is reviewed.

3 Direct Multiple Shooting

In this chapter, we present and analyze a numerical method for solving the optimal control problem (2.2). This method, which is called *direct multiple shooting*, originally proposed by Bock and Plitt [28]. It converts a continuous nonlinear optimal control problem into a NLP problem. The resulting NLP problem can be solved using a sequential quadratic programming (SQP) method, interior point optimizer (IPOPT) or sparse nonlinear optimizer (SNOPT). A new algorithm is proposed to solve this NLP problem. This algorithm is realized in the framework of the numerical algorithm group (NAG) library 8 [88] and IPOPT [178] for the SQP and in C/C++ for the rest of the computations.

3.1 Problem Transformation

In this section, the optimal control problem (2.2) is converted into a finite dimensional nonlinear programming (NLP) problem using the direct multiple shooting approach. This algorithm can be summarized by the following steps [63, 72, 73, 107, 110, 120, 152, 162]:

1. Divide the time horizon into equal subintervals.
2. Parameterize the control function in each subinterval.
3. Parameterize the initial conditions of the states in each subinterval.
4. Calculate the state trajectories in each subinterval and the values of the states at the end of each subinterval using the parameterized values of the states at the beginning of each subinterval.
5. Define continuity constraints to maintain continuity of the states between the subintervals.
6. Evaluate the objective function in each subinterval and formulate the NLP problem
7. Solve the NLP problem.

In the following, we explain these steps in detail.

Controls and States Discretization

In all direct methods, a discretiztion strategy is required. It means, the optimal control problem over the time horizon $[t_0, t_f]$ will be discretized. First, we divide the time horizon $[t_0, t_f]$ into equal subintervals $[t_i, t_{i+1}]$, such that

$$t_0 < t_1 < ... < t_N = t_f \tag{3.1}$$

where N is the total number of subintervals.

At the same time, the control and the state variables are discretized according to the time subintervals, such that

$$u_i(t) \qquad \forall \ t \in [t_i, t_{i+1}), \tag{3.2a}$$
$$x_i(t) \qquad \forall \ t \in [t_i, t_{i+1}), \tag{3.2b}$$
$$z_i(t) \qquad \forall \ t \in [t_i, t_{i+1}). \tag{3.2c}$$

where $i = 0, 1, ..., N-1$. Where t_i is the node between two neighboring subintervals.

Control Parametrization

Each piecewise control vector $u_i(t)$ is parameterized by a parameter vector v_i in each subinterval $[t_i, t_{i+1}]$ as shown in Fig. 3.1.

$$u_i(t) = u_i(t, v_i) \qquad \forall \ t \in [t_i, t_{i+1}). \tag{3.3}$$

where $i = 0, 1, ..., N-1$.

For the control parametrization piecewise constants, linear or either polynomial functions can be used. Here, for simplicity, we parameterize the control trajectories $u_i(t)$ by the piecewise *constant* representation, i.e.

$$u_i(t) = v_i, \qquad \forall \ t \in [t_i, t_{i+1}), \qquad i = 0, 1, ...N-1, \tag{3.4a}$$
$$u(t_N) = v_N = v_{N-1}. \tag{3.4b}$$

Now we have $N+1$ parameterized control vectors, $v_0, v_1, ..., v_N$, where the vector $v_i \in \mathbb{R}^{n_u}$. We notice from the piecewise constant parametrization that continuities of the control trajectories are not guaranteed. However, if these continuities are desired, the control variables can be parameterized by piecewise polynomials and treated as additional differential state variables whose time derivatives can be controlled [60].

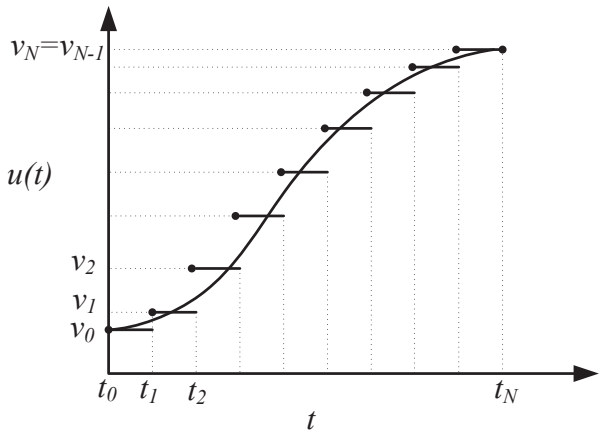

Figure 3.1: Control parametrization.

State Parametrization

For the state parametrization, the differential state vector $x(t)$ is decoupled from the algebraic state vector $z(t)$ in the solution of DAEs (2.2b) on the N subintervals. We

parameterize the initial condition of each state vector in each subinterval

$$x_i(t_i) = h_i^x, \tag{3.5a}$$
$$z_i(t_i) = h_i^z. \tag{3.5b}$$

where $i = 0, 1, ..., N-1$. Since the value of state variables at the end of the last subinterval should be considered the time and state indices will be $i = 0, 1, ..., N$. Therefore we have $2(N+1)$ state vectors $h_0^x, h_1^x, ..., h_N^x$ where $h_i^x \in \mathbb{R}^{n_x}$ as shown in Fig. 3.2 and $h_0^z, h_1^z, ..., h_N^z$ where $h_i^z \in \mathbb{R}^{n_z}$. For the compactness, we combine the vectors h_i^x and h_i^z in one vector $h_i = [h_i^{xT}\ h_i^{zT}]^T$.

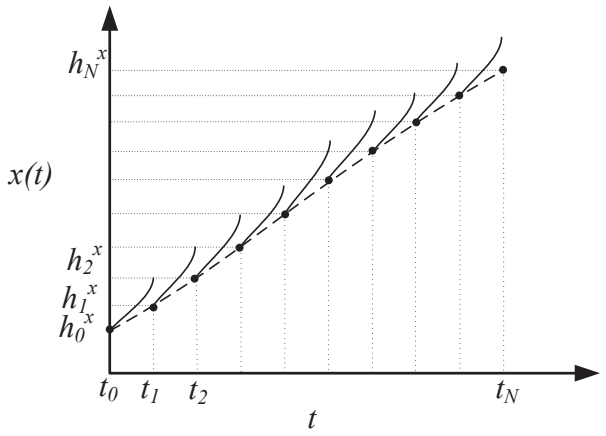

Figure 3.2: State parametrization.

Now, all control and state variables we will combine in one vector

$$w = [v_0^T\ \ h_0^{xT}\ \ h_0^{zT}\ \ v_1^T\ \ h_1^{xT}\ \ h_1^{zT}\ ...\ v_N{}^T\ \ h_N^{x\,T}\ \ h_N^{z\,T}\ \ a^T]^T,$$

where $w \in \mathbb{R}^{(n_u+n_x+n_z)(N+1)+n_a}$. For the dimensions of the variables see Eq. (2.2).

State Trajectory Solution

The state trajectories in each subinterval need to be solved. These solutions can be considered as solutions of initial value problems (IVPs) in each subinterval when the control vector $u(t)$ and the parameter vector a are parameterized by piecewise constants

$$\dot{x}_i(t) = f(x_i(t), z_i(t), v_i, a, t), \qquad \forall t \in [t_i, t_{i+1}), \tag{3.6a}$$
$$0 = g(x_i(t), z_i(t), v_i, a, t), \tag{3.6b}$$
$$x_i(t_i) = h_i^x, \qquad i = 0, 1, ..., N-1. \tag{3.6c}$$

Note that the solutions $x_i(t)$ and $z_i(t)$ are usually independent in each subinterval $[t_i, t_{i+1}]$, thus the notations $x_i(h_i, v_i, t)$ and $z_i(h_i, v_i, t)$ are used to represent this independence. In existing multiple shooting algorithms like the *multiple shooting code* (MUSCOD) [104, 131] a so-called *internal numerical differentiation* (IND) [25, 27] approach is used to approximate the state trajectories in each subinterval. The original version of this code uses a Runge-Kutta-Fehlberg method of order 7/8 [71]. In this thesis we use the method

of *collocation on finite elements* [1, 7, 35, 52, 53, 149] to approximate the solutions of the IVPs. In Chapter 4 we will present methods commonly used to solve these IVPs and briefly discuss the accuracy of each method in comparison to that of the method used in this thesis.

Algebraic Consistency Conditions

The multiple sooting nodes with parameters h_i^x, h_i^z and v_i will be imposed in the algebraic Eq. (2.2c) to construct a relaxed algebraic consistency conditions

$$g(h_i^x, h_i^z, v_i, a) = 0, \qquad i = 0, 1, ..., N.$$

It simply means that the algebraic state equation must be satisfied at joint points.

Continuity Constraints

Since the continuities of differential state trajectories must be guaranteed, so-called node matching conditions are introduced which force the terminal values $x_i(h_i, v_i, a, t_{i+1})$ of each state trajectory in each subinterval $[t_i, t_{i+1}]$ to be equal parameterized initial conditions of state trajectories h_{i+1}^x in the next subinterval $[t_{i+1}, t_{i+2}]$. From Eqs. (3.4) to (3.6) the continuity constraints result to

$$h_{i+1}^x - x_i(h_i^x, h_i^z, v_i, a, t_{i+1}) = 0, \qquad i = 0, 1, ..., N-1.$$

The parameterized initial value of the differentiable state vector h_0^x is also needed to be equal to the initial value vector x_0 in the this formulation

$$h_0^x = x_0$$

Path Constraint Discretization

The path constraints on the states and controls are imposed pointwise at the nodes and the terminal constraints at the terminal point.

$$\begin{aligned} &s(h_i^x, h_i^z, v_i, a) \geq 0, \qquad i = 0, 1, ..., N-1, \\ &r_e(h_N^x, h_N^z, a) = 0, \\ &r_i(h_N^x, h_N^z, a) \geq 0, \end{aligned}$$

Objective Function Discretization

The vectors $x_i(t)$ and $z_i(t)$ as well as the parameter vectors v_i will be used to compute the Lagrange term of Eq.(2.2a) in the subinterval $[t_i, t_{i+1}]$

$$L(h_i^x, h_i^z, v_i, a) = \int_{t_i}^{t_{i+1}} L(x_i(t), z_i(t), v_i, a, t)dt$$

where $i = 0, 1, ..., N-1$.

Furthermore, the value of the parameter $h_i = (h_i^x, h_i^z)$ is also used to compute the Mayer term in Eq.(2.2a)

$$E(x(t_f), z(t_f), a, t_f) = E(h_N^x, h_N^z, a, t_N).$$

Therefore the final form of a discretized objective function can be written as

$$J = E(h_N^x, h_N^z, a, t_N) + \sum_{i=0}^{N-1} L(h_i^x, h_i^z, v_i, a). \tag{3.7}$$

3.2 Resulting Nonlinear Programming Problem

Based on the above multiple shooting method, the finite dimensional NLP problem that results from the nonlinear optimal control problem (2.2) can be summarized:

$$\min_{\substack{v_0 \ldots, v_{N-1}, \\ h_0^x \ldots, h_N^x, \\ h_0^z \ldots, h_N^z, a}} J = E(h_N^x, h_N^z, a, t_N) + \sum_{i=0}^{N-1} L(h_i^x, h_i^z, v_i, a), \tag{3.8a}$$

subject to

$$g(h_i^x, h_i^z, v_i, a) = 0, \qquad i = 0, 1, ..., N, \tag{3.8b}$$
$$h_0^x - x_0 = 0, \tag{3.8c}$$
$$h_{i+1}^x - x_i(h_i^x, h_i^z, v_i, a, t_{i+1}) = 0, \qquad i = 0, 1, ..., N-1, \tag{3.8d}$$
$$s(h_i^x, h_i^z, v_i, a) \geq 0, \qquad i = 0, 1, ..., N, \tag{3.8e}$$
$$r_e(h_N^x, h_N^z, a) = 0, \tag{3.8f}$$
$$r_i(h_N^x, h_N^z, a) \geq 0. \tag{3.8g}$$

The NLP problem (3.8) can be rewritten in a shorter form:

$$\min_w A(w), \tag{3.9a}$$

subject to

$$B(w) = 0, \tag{3.9b}$$
$$C(w) \geq 0, \tag{3.9c}$$

with a finite dimensional vector $w \in \mathbb{R}^{n_w}$, $n_w = (N+1)(n_u+n_x+n_z)+n_a$, the scalar function $A(w) : \mathbb{R}^{n_w} \to \mathbb{R}$. The vector functions $\{B(w) : \mathbb{R}^{n_w} \to \mathbb{R}^{n_B}\}$, $n_B = (N+1)(n_x+n_z)+n_{r_e}$, $\{C(w) : \mathbb{R}^{n_w} \to \mathbb{R}^{n_C}\}$, $n_C = (N+1)n_s + n_{r_i}$, where

$$w = [v_0^T \;\; h_0^{xT} \;\; h_0^{zT} \;\; v_1^T \;\; h_1^{xT} \;\; h_1^{zT} \;\; ... \;\; v_N{}^T \;\; h_N^{x\,T} \;\; h_N^{z\;T} \;\; a^T]^T,$$

$$A(w) = E(h_N^x, h_N^z, a) + \sum_{i=0}^{N-1} L(h_i^x, h_i^z, v_i, a),$$

$$B(w) = \begin{bmatrix} h_0^x - x_0 \\ h_{i+1}^x - x_i(t_{i+1}; h_i^x, h_i^z, v_i, a) \\ g(h_i^x, h_i^z, v_i, a) \\ g(h_N^x, h_N^z, v_N, a) \\ r_e(h_N^x, h_N^z, a) \end{bmatrix}$$

and

$$C(w) = \begin{bmatrix} s(h_i^x, h_i^z, v_i, a) \\ s(h_N^x, h_N^z, v_N, a) \\ r_i(h_N^x, h_N^z, a) \end{bmatrix}$$

where $i = 0, 1, ..., N-1$.

Fig. 3.3a shows a graphical description of an initialization of a simple optimal control problem using direct multiple shooting in 4 subintervals. In this figure, initial guesses for v_i and h_i^x are given inside the boundary values, i.e. the initial guess should be feasible. The profiles of the state variables are computed by solving the model equations. When the NLP problem (3.9) is solved, optimal values of control and state trajectories will be obtained as shown schematically in Fig. 3.3b. We can see from Fig. 3.3 that an integration of the state trajectory over the whole time interval $[t_0, t_f]$ is not required, but this integration is done separately over the subintervals $[t_i, t_{i+1}]$. Moreover using the optimal state and control trajectories the equality and inequality constraints will be satisfied, and the performance index $A(w)$ will be minimized. In Section 3.3 several methods to solve the NLP problem (3.9) will be presented.

3.3 NLP Problem Solution

The goal of this section is to present numerical methods that solve the NLP problem (3.9). We first define a gradient vector $\nabla A(w) = [\frac{\partial A}{\partial w_0}, ..., \frac{\partial A}{\partial w_{n_w-1}}]$, Jacobian matrices $\nabla_w B(w) = \frac{\partial B}{\partial w}$ and $\nabla_w C(w) = \frac{\partial C}{\partial w}$ with dimensions $n_w \times n_B$ and $n_w \times n_C$, respectively. We assume that Hessian matrices $\nabla^2_{ww} A$, $\nabla^2_{ww} B$ and $\nabla^2_{ww} C$ are available. In the following, basic definitions and theorems are given.

Definition 3.1 (Feasibility):
A point $w^* \in \mathbb{R}^{n_w}$ is **feasible** *in the NLP problem (3.9) if $B(w^*) = 0$ and $C(w^*) \geq 0$ (i.e. all equality and inequality constraints are satisfied).*

We note that the vector w^* may not minimize the function $A(w)$ although it is a feasible point.

Definition 3.2 (Local Optimality):
A point $w^* \in \mathbb{R}^{n_w}$ is a **local minimizer** *(or a local minimum solution) of the NLP problem (3.9) if:*

- *w^* is a feasible point and*
- *w^* satisfies $A(w^*) \leq A(w)$ for all feasible neighborhoods of w^*.*

Definition 3.3 (Global Optimality):
A point $w^* \in \mathbb{R}^{n_w}$ global minimizer of the NLP problem (3.9) if:

- *w^* is a local minimizer of the NLP problem (3.9) and*
- *for all existing $w^* \in \mathbb{R}^{n_w}$ such that $A(w^*) \leq A(w)$ for all feasible points $w \neq w^*$.*

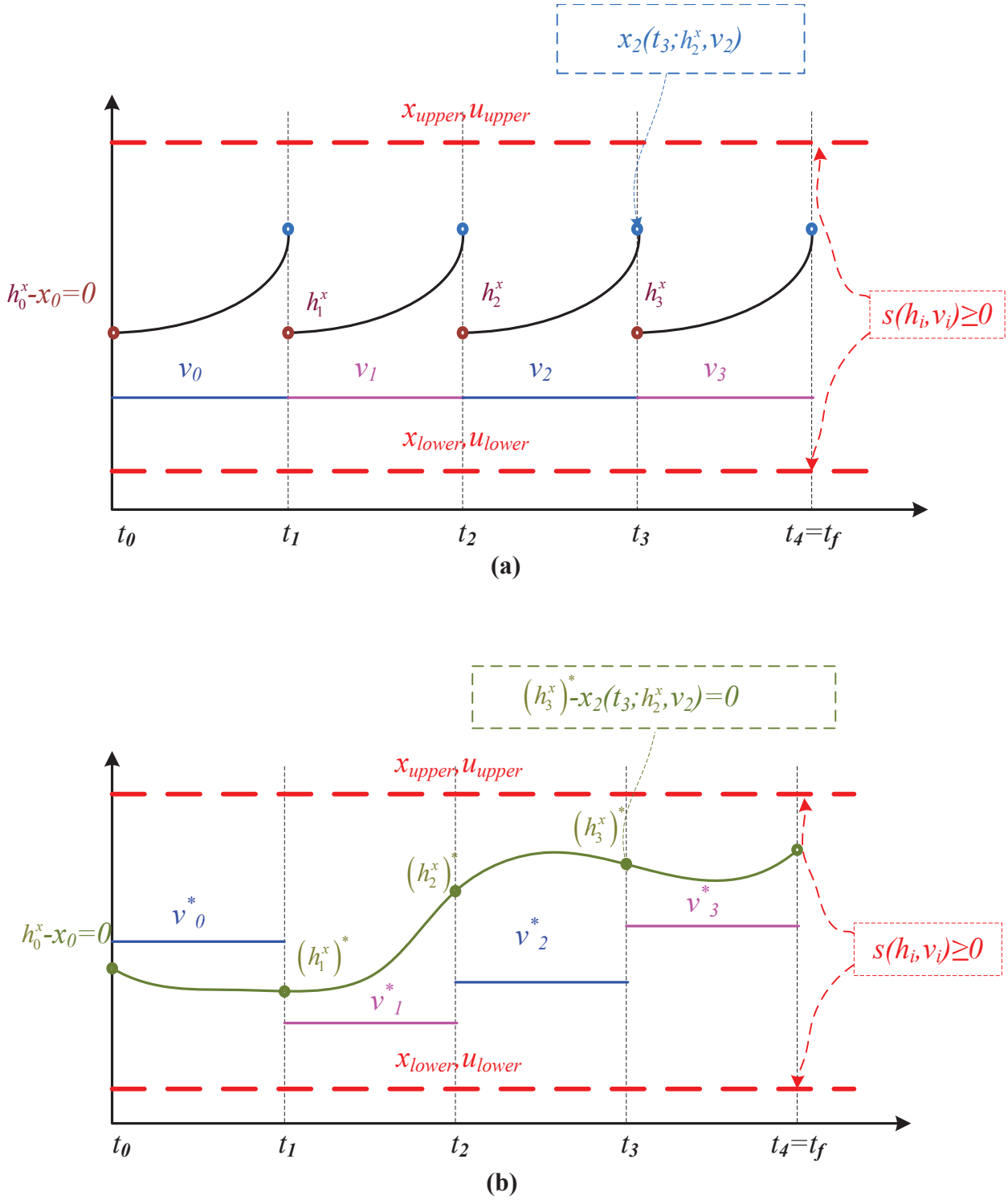

Figure 3.3: Multiple shooting method- initial and finial trajectory (N=4).

Definition 3.4 (Lagrangian Function and Lagrangian Multipliers):
A Lagrangian function is defined to investigate the local optimality of the NLP problem (3.9) by introducing Lagrangian multiplier vectors $\lambda \in \mathbb{R}^{n_B}$ *and* $\mu \in \mathbb{R}^{n_C}$ *which correspond*

one-to-one to the constraint functions B and C, respectively, such that,

$$L(w, \lambda, \mu) = A(w) - \lambda^T B(w) - \mu^T C(w). \tag{3.10}$$

Theorem 3.1 (Karush-Kuhn-Tucker Conditions):
Consider the NLP problem (3.9) has a Lagrangian function (3.10). *Then w^* is a* local optimum *of the NLP problem (3.9) if the triple (w^*, λ^*, μ^*) satisfies the following necessary conditions:*

- *Stationarity condition:*

$$\nabla L(w^*, \lambda^*, \mu^*) = \nabla A(w^*) - \frac{\partial B}{\partial w^*}\lambda^* - \frac{\partial C}{\partial w^*}\mu^* = 0. \tag{3.11a}$$

- *Primal feasibility conditions:*

$$B(w^*) = 0, \tag{3.11b}$$
$$C(w^*) \geq 0. \tag{3.11c}$$

- *Dual feasibility condition:*

$$\mu^* \geq 0, \tag{3.11d}$$

- *Complementary slackness condition:*

$$\mu_i^* C_i(w^*) = 0, \qquad i = 1, 2..., n_C. \tag{3.11e}$$

Definition 3.5 (Karush-Kuhn-Tucker Point):
A triple (w^*, λ^*, μ^*) is called a Karush-Kuhn-Tucker (KKT) point when it satisfies the condition (3.11) of Theorem 3.1.

Definition 3.6 (Weakly Active and Strongly Active Constraints [60]):
If the inequality constraints $C(w^*) \geq 0$ are active, i.e. $C(w^*) = 0$, then

- *the inequality constraints $C(w^*) \geq 0$ are **weakly** active constraints if the complementary condition (3.11e) is satisfied such that $\mu^* = 0$.*
- *the inequality constraints $C(w^*) \geq 0$ are **strongly** active constraints if the complementary condition (3.11e) is satisfied such that $\mu^* > 0$.*

3.3.1 Quadratic Programming (QP)

One of the most important classes of the NLP problems (3.9) is a NLP problem with *quadratic* objective function and *linear* constraints which is called *quadratic programming* (QP) problem [135]. This type of the NLP problems is very important since it is considered as a subproblem in methods for general constrained optimization such as sequential quadratic programming method. The quadratic programming (QP) problem can be formulated as

$$\min_w \frac{1}{2} w^T A_{QP} w + a_{QP}^T w, \tag{3.12a}$$

subject to

$$B_{QP}w + b_{QP} = 0, \tag{3.12b}$$
$$C_{QP}w + c_{QP} \geq 0, \tag{3.12c}$$

where the matrices $A_{QP} \in \mathbb{R}^{n_w} \times \mathbb{R}^{n_w}$, $B_{QP} \in \mathbb{R}^{n_B} \times \mathbb{R}^{n_w}$ and $C_{QP} \in \mathbb{R}^{n_C} \times \mathbb{R}^{n_w}$ are predefined constant matrices and vectors $a_{QP} \in \mathbb{R}^{n_w}$, $b_{QP} \in \mathbb{R}^{n_B}$ and $c_{QP} \in \mathbb{R}^{n_C}$ are predefined constant vectors. A_{QP} is a symmetric and positive definite matrix.

Many algorithms have been developed to solve QP problems. Here we show one of the algorithms mostly used for the solution.

Definition 3.7 (Lagrangian of QP Problem):
The Lagrangian of the QP problem (3.12) is given by:

$$L_{QP}(w, \lambda_{QP}, \mu_{QP}) = w^T A_{QP} w + a_{QP} - \lambda_{QP}^T (B_{QP} w + b_{QP}) - \mu_{QP}^T (C_{QP} w + c_{QP}) \tag{3.13}$$

where $\lambda_{QP} \in \mathbb{R}^{n_B}$ *and* $\mu_{QP} \in \mathbb{R}^{n_C}$.

Applying the KKT conditions (3.11) in Theorem 3.1 to a KKT point $(w^*, \lambda_{QP}^*, \mu_{QP}^*)$, we find that

$$A_{QP}w^* + a_{QP} - B_{QP}^T \lambda_{QP}^* - C_{QP}\mu_{QP}^* = 0, \tag{3.14a}$$
$$b_{QP} + B_{QP}w^* = 0, \tag{3.14b}$$
$$c_{QP} + C_{QP}w^* \geq 0, \tag{3.14c}$$
$$\mu_{QP}^* \geq 0, \tag{3.14d}$$
$$\mu_{QP_i}^* (c_{QP_i} + C_{QP_i} w^*) = 0, \quad i = 1, 2, ..., n_C. \tag{3.14e}$$

This problem is usually solved by the active-set method [5]. A unique solution, i.e. KKT point, of the QP problem (3.12) can be found for inequality constraints (3.12b) by assuming that the QP problem (3.12) is feasible and the combined constraint matrix (B_{QP}^T, C_{QP}) has full rank $n_B + n_C$.

3.3.2 Sequential Quadratic Programming (SQP)

***S**equential **q**uadratic **p**rogramming* or ***s**uccussive **q**uadratic **p**rogramming* (SQP) [5, 170, 174], is considered one of the the most effective and robust algorithms for solving NLP problems. We consider the short form of the resulting NLP problem, as described in Section 3.2,

$$\min_w A(w), \tag{3.15a}$$

subject to

$$B(w) = 0, \tag{3.15b}$$
$$C(w) \geq 0, \tag{3.15c}$$

where $w \in \mathbb{R}^{n_w}$ is a finite dimensional vector, $A(w) : \mathbb{R}^{n_w} \to \mathbb{R}$ is a scalar function , $\{B(w) : \mathbb{R}^{n_w} \to \mathbb{R}^{n_B}\}$, $\{C(w) : \mathbb{R}^{n_w} \to \mathbb{R}^{n_C}\}$ are vector functions and at least one of the constraints is nonlinear.

Solving the NLP problem using SQP is based on solving a *series* of quadratic programming subproblems that are designed to minimize a quadratic approximation of the objective function subject to linearized constraints. SQP starts with a point, $w_k, \quad k = 0, 1, ...,$ in the NLP problem (3.15) by a quadratic programming (QP) subproblem, and then uses the solution to the subproblem to construct a better approximation with a new point w_{k+1}. This process is iterated to create a sequence of approximations that will converge to a solution w^*. If the problem is unconstrained SQP reduces to Newton's method for finding a point where the gradient of the objective vanishes. If the problem has only equality constraints the SQP method, which uses an exact penalty function, is equivalent to applying Newton's method to the first-order optimality conditions, or KKT conditions, of the problem [4, 135].

Lemma 3.1 (Exact Penalty Function):
Suppose that the triple (w^*, λ^*, μ^*) satisfies the KKT conditions (3.11) to the problem (3.15), furthermore, the functions A and B are convex and the function C is affine. If a penalty function is defined as

$$P_e(w) = A(w) + \rho \left(\sum_i^{n_w} \max\left(0, B_i(w)\right) + \sum_i^{n_w} |B_i(w)| \right), \tag{3.16}$$

where $\rho > 0$, and ρ is large enough such that

$$\rho \geq \max\{\mu^*, |\lambda|\},$$

then the vector w^* is also a global minimum of the function $P_e(w)$:

Proof. A proof of this lemma can be found in [5].

Definition 3.8 (Search Direction and Relaxation Factor):
Consider the NLP problem (3.15) with a Lagrangian $L(w, \lambda, \mu)$. Newton's method takes a step in a direction towards a near optimal solution of the problem. If the Newton's method begins with initial guess (w_k, λ_k, μ_k) then the characterization of the next iterate $(w_{k+1}, \lambda_{k+1}, \mu_{k+1})$ is given by

$$(w_{k+1}, \lambda_{k+1}, \mu_{k+1}) = (w_k, \lambda_k, \mu_k) + \alpha_k(\Delta w_k, \Delta\lambda_k, \Delta\mu_k) \tag{3.17}$$

where $\alpha \in (0, 1]$ is the relaxation factor of the Newton's method [5, 24, 104, 158] and $(\Delta w_k, \Delta\lambda_k, \Delta\mu_k) \in (\mathbb{R}^{n_w}, \mathbb{R}^{n_B}, \mathbb{R}^{n_C})$ solves a second-order approximation of a stationary point condition for the Lagrange

$$\nabla^2 L(w_k, \lambda_k, \mu_k) \begin{bmatrix} \Delta w_k \\ \Delta\lambda_k \\ \Delta\mu_k \end{bmatrix} = -\nabla L(w_k, \lambda_k, \mu_k). \tag{3.18}$$

From Eq. (3.18), we yield:

$$\begin{bmatrix} \nabla^2_{ww} L(w_k, \lambda_k, \mu_k) & \nabla B(w_k) & \nabla C(w_k) \\ \nabla B(w_k) & 0 & 0 \\ \nabla C(w_k) & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta w_k \\ \Delta\lambda_k \\ \Delta\mu_k \end{bmatrix} = - \begin{bmatrix} \nabla_w L(w_k, \lambda_k, \mu_k) \\ B(w_k) \\ C(w_k) \end{bmatrix}. \tag{3.19}$$

System (3.19) of linear equations has a KKT point corresponds to the quadratic programming problem:

$$\min_{w} \frac{1}{2}\Delta {w_k}^T H_k \Delta w_k + \nabla A(w_k)\Delta w_k, \tag{3.20a}$$

subject to

$$\begin{aligned} B(w_k) + \nabla B(w_k)\Delta w_k &= 0, & (3.20b)\\ C(w_k) + \nabla C(w_k)\Delta w_k &\geq 0. & (3.20c) \end{aligned}$$

where $H_k = \nabla^2_{ww} L(w_k, \lambda_k, \mu_k)$ is the Hessian matrix of the Lagrangian L or its approximation. This problem is a second-order approximation of the Lagrangian with respect to the primal vector w, and the original constraints have been replaced by their first-order approximations at w_k.

Theorem 3.2 (Convergence of the SQP Method):
The SQP algorithm terminates at a KKT point for the problem (3.15) for every $w_k \in \mathbb{R}^{n_w}$, a symmetric and positive definite Hessian $\nabla^2_{ww} L(w_k, \lambda_k, \mu_k)$ or its approximation if

- *the QP problem (3.20) has a unique solution,*
- *the unique Lagrangian multiplier vectors λ and μ satisfying $\rho \geq \max\{\lambda, \mu\}$ where $\rho > 0$ is the penalty parameter (see Eq. (3.16)) and*
- *the sequence of the Hessian or its approximation is bounded and that every point of this sequence is positive definite.*

Proof. The proof of Theorem 3.2 can be found in [5].

Approximation of the Hessian

The analytical computation of the Hessian of the Lagrange is very expensive, thus in many SQP implementations, the identity matrix I is used as initial Hessian approximation H_0. This approximation, however, is quit unsatisfactory. A more 'reasonable' approximation of Hessian [45, 105, 139] has been studied.

The Hessian matrix H_k has n_w^2 elements and is hence under-determined by these n_w equations. Additional requirements, such as making H_k symmetric and positive definite, can be a result in particular quasi-Newton methods. Starting from $H_0 = I$, H_{k+1} is calculated from H_k using a rank-one or rank-two matrix update. In particular, this allows us to update the factorization of H_k to efficiently obtain the factorization of H_{k+1} using standard algorithms in linear algebra [5, 135]. There are many algorithms that can be used to compute the approximation H_{k+1}. One of the most famous and effective algorithms is the so-called Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [32, 79, 85, 163] where

$$H_{k+1} = H_k - \frac{H_k \sigma_k (H_k \sigma_k)^T}{\sigma_k^T H_k \sigma_k} + \frac{\phi_k \phi_k^T}{\phi_k^T \phi_k}, \tag{3.21}$$

with $\sigma_k = w_{k+1} - w_k$ and $\phi_k = \nabla A(w_{k+1}) - \nabla A(w_k)$.

Summary of SQP Algorithm

To summarize the SQP method, the NLP problem (3.15) is solved starting from an initial guess w_0, then each iteration k will result in

$$w_{k+1} = w_k + \alpha_k \Delta w_k, \qquad k = 0, 1, ... \tag{3.22}$$

where $\alpha_k \in (0, 1]$ is the step length factor and Δw_k is the search direction. The search direction is derived from the solution of the quadratic programming (QP) sub-problem:

$$\min_w \frac{1}{2} \Delta {w_k}^T H_k \Delta w_k + \nabla A(w_k) \Delta w_k, \tag{3.23a}$$

subject to

$$B(w_k) + \nabla B(w_k) \Delta w_k = 0, \tag{3.23b}$$
$$C(w_k) + \nabla C(w_k) \Delta w_k \geq 0. \tag{3.23c}$$

where H_k is the BFGS approximation of the Hessian of the Lagrangian function (3.10) and can be given by:

$$H_k = \begin{cases} I & \text{if } k = 0, \\ H_{k-1} - \frac{H_{k-1}\sigma_{k-1}(H_{k-1}\sigma_{k-1})^T}{\sigma_{k-1}^T H_{k-1}\sigma_{k-1}} + \frac{\phi_{k-1}\phi_{k-1}^T}{\phi_{k-1}^T \phi_{k-1}} & \text{if } k > 0. \end{cases} \tag{3.24}$$

where $I \in \mathbb{R}^{n_w} \times \mathbb{R}^{n_w}$ is the identity matrix , $\sigma_k = w_{k+1} - w_k$ and $\phi_k = \nabla A(w_{k+1}) - \nabla A(w_k)$. Algorithm 1 [5, 135] is normally used to solve the NLP problem in many dynamic optimization frameworks, e.g. MUSCOD [103, 106].

Algorithm 1 SQP Algorithm

Initialization: given (w_0, λ_0, μ_0), H_0 and an objective function.
Step 0: ***set*** $k = 0$.
Step 1: ***Formulate and solve the QP*** problem (3.23) to obtain (Δw_k) and the multipliers $(\Delta\lambda_k, \Delta\mu_k)$.
Step 2: ***Choose*** the step length factor α_k.
Step 3: ***Set*** $(w_{k+1}, \lambda_{k+1}, \mu_{k+1}) = (w_k, \lambda_k, \mu_k) + \alpha_k(\Delta w_k, \Delta\lambda_k, \Delta\mu_k)$.
Step 4: ***Stop if*** $(w_{k+1}, \lambda_{k+1}, \mu_{k+1})$ converges.
Step 5: ***Compute*** H_{k+1}.
Step 6: ***Set*** $k = k + 1$, ***go to*** *Step* 1.

3.3.3 Sparse Nonlinear Optimizer (SNOPT)

One of the most effective algorithms in the optimization community is the *sparse nonlinear optimizer* (SNOPT) [82–84]. Here we summarize the main features of the SQP algorithm used in SNOPT. The basic structure of the SQP method involves major and minor iterations. The major iterations generate a sequence of iterates w_k that satisfy the constraints within both the equality $B(w)$ and the inequality $C(w)$ constraints and converge to a point that satisfies the nonlinear constraints and the first-order optimality conditions. Therefore

the linear constraints are decoupled from the nonlinear constraints, and both equality and inequality constraints can be written as inequality constraints. The NLP problem (3.15) can be rewritten as

$$\min_{w_{\min} \leq w \leq w_{\max}} A(w), \tag{3.25a}$$

subject to

$$\begin{aligned} D_{\min} &\leq D(w) \leq D_{\max}, & (3.25b)\\ F_{\min} &\leq Fw \leq F_{\max}, & (3.25c) \end{aligned}$$

where $D(w) \in \mathbb{R}^{n_D}$ is used to construct the nonlinear inequality constraints (3.25b) from the nonlinear functions in both $B(w)$ and $C(w)$. We assume that $D(w)$ is smooth and differentiable. $F \in \mathbb{R}^{n_F} \times \mathbb{R}^{n_F}$ is a sparse matrix and used to construct linear inequality constraints (3.25c) from both $B(w)$ and $C(w)$. Note that the boundaries $D_{\min}$, $D_{\max}$, $F_{\min}$, and $F_{\max}$ are chosen carefully to convert the equality constraints in Eq. (3.15b) into inequality constraints. SNOPT then converts the general constraints (3.25b) and (3.25c) to equalities by introducing a vector of slack variables $s \in \mathbb{R}^{n_D+n_F}$. Thus, the NLP problem (3.25) will be:

$$\min_{w,s} A(w), \tag{3.26a}$$

subject to

$$\begin{aligned} D(w) - s_D &= 0, & (3.26b)\\ Fw - s_F &= 0, & (3.26c) \end{aligned}$$

with the slack variables $s_D \in \mathbb{R}^{n_D}$, $s_F \in \mathbb{R}^{n_F}$ and $s = [s_D^T \; s_F^T]^T$. We note that the slack variables are introduced in both sides of Eqs. (3.25c) to (3.25b). In each *major* iteration k a QP subproblem is used to generate a search direction for the next iterate w_{k+1}. The constraints of the subproblem are consist from the linear constraints $Fw - s_F = 0$ and the linearized constraints

$$D(w_k) + \nabla D(w_k)(w - w_k) - s_D = 0,$$

consequently, the QP subproblem can be written as [82]:

$$\min_{w,s_D,s_F} \frac{1}{2} \Delta {w_k}^T H_k \Delta w_k + \nabla A(w_k) \Delta w_k, \tag{3.27a}$$

subject to the linear and linearized equality constraints

$$\begin{aligned} \nabla D(w_k)w - s_D &= -D(w_k) + \nabla D(w_k)w_k, & (3.27b)\\ Fw - s_F &= 0, & (3.27c) \end{aligned}$$

where $\nabla A(w_k)$ and H_k are a gradient and a quadratic approximation to the Hessian of the Lagrangian L_S, respectively, where

$$L_S(w, s_D, s_F, \lambda_S, \mu_S) = A(w) - \lambda_S^T(D(w) - s_D) - \mu_S^T(Fw - s_F),$$

where a vector and $\lambda_S \in \mathbb{R}^{n_D}$ and $\mu_S \in \mathbb{R}^{n_F}$ are Lagrangian multiplier vectors, and $\nabla D(w_k)$ is a Jacobian matrix, whose elements are first derivatives of $D(w)$ evaluated at

w_k. A BFGS update is applied after each major iteration. If some of the variables enter the Lagrangian linearly the Hessian will have some zero rows and columns. If the nonlinear constraints appear, then only the leading n_D rows and columns of the Hessian need to be approximated. Inside each major iteration, the QP subproblem (3.27) will be also solved iteratively [84]. In each QP iteration, which is called minor iteration, a system of linear equations will be solved.

The main advantages of the SNOPT are

- SNOPT is suitable for large-scale linear and quadratic programming and for linearly constrained optimization, as well as for general nonlinear programs. That means, it is more efficient on large NLP problems if only some variables enter nonlinearly.
- SNOPT finds solutions that are locally optimal solutions that are often global solutions. Ideally any nonlinear functions should be smooth with pre-provided gradients, but discontinuities in the function gradients can often be tolerated if they are not too close to an optimum.
- SNOPT uses a sequential quadratic programming (SQP) algorithm and the search directions are obtained from QP subproblems that minimize a quadratic model of the Lagrangian function subject to linearized constraints.
- SNOPT requires relatively few evaluations of the problem functions. Thus it is especially effective if the objective or constraint functions (and their gradients) are expensive to evaluate.

The main steps of the SNOPT algorithm and more information can be found in [83].

3.3.4 Interior Point Optimizer (IPOPT)

The *interior point* or barrier SQP method is also one of the most efficient algorithms for solving large-scale NLP problems. In this thesis, we use IPOPT, a software package developed by Wächter and Bigeler [178] based on the interior point method, to solve the resulting NLP problem (3.9). IPOPT uses the general and typical NLP problem form that frequently arises and is treated in the literature [18, 137, 180, 181, 183] :

$$\min_{w} A(w), \tag{3.28a}$$

subject to

$$B(w) = 0, \tag{3.28b}$$

$$\underline{c} \leq C(w) \leq \overline{c}, \tag{3.28c}$$

where ($\underline{c}$ and $\overline{c}$) $\in \mathbb{R}^{n_C}$ are vectors of upper and lower bound values for the inequality constraints $C(w)$, respectively. Since the structure of the resulting NLP problem is already defined in Eq. (3.9) the vectors of upper and lower bound values are $\underline{c} = [\underline{c}_0, ..., \underline{c}_{n_C-1}]^T = [0, ..., 0]^T$ and $\overline{c} = [\overline{c}_0, ..., \overline{c}_{n_C-1}]^T = [\infty, ..., \infty]^T$.

In IPOPT the inequality constraints are transformed into equality constraints. This is done by adding a logarithmic barrier term to the objective function and non-negative slack

variables $(\underline{s}, \overline{s})$ to the inequality constraints. Thus the NLP problem (3.28) is reformulated as:

$$\min_{w} A(w) - \zeta_l \sum_{i=0}^{n_C-1} \ln(\underline{s}_i + \overline{s}_i), \tag{3.29a}$$

subject to

$$\begin{aligned} B(w) &= 0, && (3.29b)\\ C(w) - \underline{s} - \underline{c} &= 0, && (3.29c)\\ C(w) + \overline{s} - \overline{c} &= 0, && (3.29d) \end{aligned}$$

where $\underline{s} = [\underline{s}_0 \ ... \ \underline{s}_{n_C-1}]^T$, $\overline{s} = [\overline{s}_0 \ ... \ \overline{s}_{n_C-1}]^T$ and a barrier parameter $\zeta_l > 0$ is enforced to decrease towards zero iteratively with index l. As ζ_l tends zero, as the iterations progress, the solution of problem (3.29) approaches a local solution w^*. The IPOPT code applies the Newton method to KKT conditions derived from problem (3.29). Then a following sparse linear system needs to be solved at every iteration k [137, 172]:

$$\begin{bmatrix} M_{1k} & 0 & \underline{S}_k & 0 & 0 & 0 \\ 0 & M_{2k} & 0 & \overline{S}_k & 0 & 0 \\ -\underline{S}_k & 0 & 0 & 0 & \nabla C^T(w_k) & 0 \\ 0 & -\overline{S}_k & 0 & 0 & -\nabla C^T(w_k) & 0 \\ 0 & 0 & \nabla C(w_k) & -\nabla C(w_k) & \nabla^2_{ww} l_\zeta(w_k) & -\nabla B^T(w_k) \\ 0 & 0 & 0 & 0 & -\nabla B(w_k) & 0 \end{bmatrix} \begin{bmatrix} \Delta\underline{s} \\ \Delta\overline{s} \\ \Delta\mu_1 \\ \Delta\mu_2 \\ \Delta w \\ \Delta\lambda \end{bmatrix}$$

$$= - \begin{bmatrix} \underline{S}_k\mu_{1k} - \zeta_l e \\ \overline{S}_k\mu_{2k} - \zeta_l e \\ -(C(w_k) - \underline{s}_k - \underline{c}_k) \\ -(C(w_k) + \overline{s_k} - \overline{c_k}) \\ \nabla A(w_k) - \nabla B(w_k)\lambda_k - \nabla C(w_k)\mu_{1k} - \nabla C(w_k)\mu_{2k} \end{bmatrix}, \tag{3.30}$$

where λ, μ_1 and μ_2 are the Lagrangian multipliers, $M_1 = \mathrm{diag}(\mu_1)$, $M_2 = \mathrm{diag}(\mu_2)$, $\underline{S} = \mathrm{diag}(\underline{s})$, $\overline{S} = \mathrm{diag}(\overline{s})$, $e = [1 \ ... \ 1]^T$ and $\nabla^2_{ww} l_\zeta(w)$ is the Hessian of the Lagrangian L_ζ,

$$L_\zeta(w, \lambda, \overline{s}, \underline{s}, \mu_1, \mu_2) = A(w) - \lambda^T B(w) - \mu_1^T(C(w) - \underline{s} - \underline{c}) - \mu_2^T(C(w) + \overline{s} - \overline{c}). \tag{3.31}$$

By solving the linear equation (3.30) we can obtain a direction of the Newton method. Rewriting Eq.(3.30) in a reduced form

$$\begin{bmatrix} H_k & -\nabla B^T(w_k) \\ -\nabla B(w_k) & 0 \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \Gamma \\ B(w) \end{bmatrix} \tag{3.32}$$

we compute for Δw and $\Delta\lambda$, where $H_k = \nabla^2_{ww} L_\zeta + \nabla C(w_k)(\underline{S}_k^{-1}\mu_1 - \overline{S}_k^{-1}\mu_2)\nabla C^T(w_k)$, and $\Gamma = \nabla_w L_\zeta - \nabla C(w_k)(\underline{S}_k^{-1}\mu_1 - \overline{S}_k^{-1}\mu_2)\nabla C^T(w_k) + \underline{S}_k^{-1}\nabla_{\underline{s}} L_\zeta - \overline{S}_k^{-1}\nabla_{\overline{s}} L_\zeta$. From Δw and $\Delta\lambda$ we compute $\Delta\underline{s}$, $\Delta\overline{s}$, $\Delta\mu_1$ and $\Delta\mu_2$.

$$\begin{aligned} \Delta\underline{s} &= \nabla C^T(w_k)\Delta w - \nabla_{\mu_1} L_\zeta, & \Delta\overline{s} &= \nabla C^T(w_k)\Delta w - \nabla_{\mu_2} L_\zeta,\\ \Delta\mu_1 &= \underline{S}_k^{-1}(-\mu_{1k} \cdot \Delta\underline{s} - \nabla_{\underline{s}} L_\zeta), & \Delta\mu_2 &= \overline{S}_k^{-1}(-\mu_{2k} \cdot \Delta\overline{s} + \nabla_{\overline{s}} L_\zeta). \end{aligned}$$

The primal variables w, $\underline{s}$ and $\overline{s}$ and the dual variables can be updated from

$$\begin{aligned} w_{k+1} &= w_k + \alpha_k^p \Delta w, & \lambda_{k+1} &= \lambda_k + \alpha_k^d \Delta\lambda, \\ \underline{s}_{k+1} &= \underline{s}_k + \alpha_k^p \Delta\underline{s}, & \mu_{1k+1} &= \mu_{1k} + \alpha_k^d \Delta\mu_1, \\ \overline{s}_{k+1} &= \overline{s}_k + \alpha_k^p \Delta\overline{s}, & \mu_{2k+1} &= \mu_{2k} + \alpha_k^d \Delta\mu_2, \end{aligned} \tag{3.33}$$

where scalars $\alpha^p,\ \alpha^d \in (0, 1]$ are step length values of the primal and the dual parameters, respectively [171].

The IPOPT code [180] applies Algorithm 2 to solve the NLP problem. More information on the IPOPT method can be found in [29, 39, 134, 137, 178]. The advantages of the IPOPT method over other SQP methods can be summarized as follows:

- IPOPT uses a simple computation of search directions.
- The reduced linear system (3.30) can be solved by several approaches using state-of-the-art linear algebra packages.
- IPOPT considers a number of alternative Merit functions to find the line search.
- Jacobian and Hessian in the IPOPT can be sparse, see (3.30).
- In the implementation of IPOPT, the program stores only nonzero terms in the sparse matrix to save memory space.

Algorithm 2 IPOPT algorithm [179]

Step 0: ***Initialization***
Set $k = 0$, define a barrier parameter $\zeta_0 > 0$ and choose a starting initial guess $(\underline{s}_0, \overline{s}_0, \mu_{10}, \mu_{20}, w_0, \lambda_0)$ that satisfies strict positivity conditions.

Step 1: ***Compute the Newton direction:***
Obtain the Newton system Eq. (3.30) at the current point and solve for a Newton direction $(\Delta\underline{s}, \Delta\overline{s}, \Delta\mu_1, \Delta\mu_2, \Delta w, \Delta\lambda)$.

Step 2: ***Update variables:***
Compute step length factors α_k^p and α_k^d in a direction of $(\Delta\underline{s}, \Delta\overline{s}, \Delta w, \Delta\lambda, \Delta\mu_1, \Delta\mu_2)$ and update the variables using Eq. (3.33).

Step 3: ***Test for convergence:***
If the new point ***satisfies*** the convergence criteria, stop.
Otherwise set $k = k + 1$, update the barrier parameter ζ_l, and ***go*** to *Step* 1.

Table 3.1 shows a comparison between SNOPT and IPOPT. From this comparison we conclude that if the large-scale NLP problem contains many linear constraints it is more efficient to solve it using SNOPT. However, using multiple shooting, the nonlinear model equations and constrains (2.2b) to (2.2k) cause nonlinear constraints (3.9b) to (3.9c), and thus it can be solved more efficiently using IPOPT. For this reason, we use IPOPT to solve the NLP problems.

Table 3.1: Comparison between SNOPT and IPOPT.

	SNOPT	IPOPT
Large NLP problem	Yes	Yes
NLP with many linear constraints	More efficient	Less efficient
NLP with many nonlinear constraints	Less efficient	More efficient
Transformation inequalities into equalities	Through slack variables	Through slack variables and barrier term
Each iteration solves	QP problem	System of linear equations
Convergence method	KKT	KKT
CPU time	Less in case of many linear constraints	Less in case of many nonlinear constraints
Sparsity of Jacobian and Hessian	Yes	Yes
Memory reserved in the program	More	Less

Numerical Implementation for Solving NLP problems

All NLP solvers are based on iterative algorithms. In each iteration, all the information of the NLP problem must be computed or approximated and then be provided to the NLP solver. These informations are the objective function $A(w)$ and its gradient with respect to the vector w, $\nabla_w A(w)$, the equality, $B(w)$, and inequality, $C(w)$, functions (see, e.g., Eq. (3.15)) and their Jacobians with respect to the vector w, $\nabla_w B(w)$ and $\nabla_w C(w)$, respectively, and the Hessian (or its approximation) of the Lagrangian $\nabla^2_{ww} L_{NLP}$.

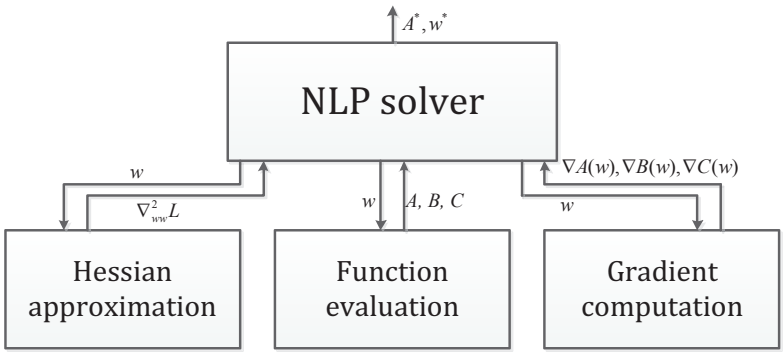

Figure 3.4: Informations follow SQP iteration.

The output of each NLP solver iteration is an objective function value and near converged variable vector w as shown in Fig. 3.4. When the vector w has converged (i.e. the KKT conditions are satisfied) then the iteration will be quitted. The sensitivity information, i.e. $\nabla_w A(w)$, $\nabla_w B(w)$ and $\nabla_w C(w)$, plays an important role in each iteration and its computation requires much CPU-time. In the available multiple shooting algorithms, e.g. MUSCOD, it is done by the integration of DAEs by means of a DAE solver and then the chain-rule for the sensitivity computation is used. In this work we employ the method

of collocation on finite elements to carry out the DAE integration and compute these sensitivities for each shooting as will be described in Chapter 5.

4 Solution of Differential Algebraic Equations

4.1 Introduction

In Chapter 3 the nonlinear optimal control problem was transformed into a large and spares NLP problem. The transformation of the optimal control problem into a NLP problem implies that the equality constraints of this NLP problem guarantees the continuity property of state trajectories. Each parameterized initial condition in each subinterval must be equal to a terminal value of the state in the previous subinterval. To find these terminal values we have to compute the state trajectories in each subinterval independently. In other words, we have to solve DAEs of the system model in each subinterval.

After parameterizing controls and algebraic states, we solve the following *initial value problem* (IVP):

$$\dot{x}_i(t) = f(x_i(t), z_i(t), v_i, a, t), \qquad \forall t \in [t_i, t_{i+1}), \tag{4.1a}$$

$$0 = g(x_i(t), z_i(t), v_i, a, t), \tag{4.1b}$$

$$x_i(t_i) = h_i^x \qquad i = 0, 1, ..., N-1. \tag{4.1c}$$

In the the following, some general approaches for the solution of DAEs are briefly summarized. Specifically, available multiple shooting algorithms for the solution of IVP (4.1) are given attention. In the existing multiple-shooting algorithms like MUSCOD [66, 103], typical Euler and Runge-Kutta methods are used to solve the IVP. In this work, we use collocation on finite elements (CFE) to solve the ODE integration problem. In addition, sensitivities that are needed in each shoot will be also computed using the same method. The accuracy and the speed of this new approach will be compared with other DAE solvers.

In general and for simplicity of presentation, it is assumed that the DAE (4.1) can be rewritten in an explicit, or a normal form

$$\dot{y}_i = f(y_i(t), t), \tag{4.2}$$

where $y_i(t) \in \mathbb{R}^{n_y}$. To rewrite the IVP (4.1) in the form of (4.2), we consider the following reformulation. For this, we differentiate the constraint equation (4.1b) with respect to t to get

$$\dot{x}_i(t) = f(x_i(t), z_i(t), v_i, a, t), \tag{4.3a}$$

$$\nabla_{x_i} g \dot{x}_i + \nabla_{z_i} g \dot{z}_i = -\dot{g}(x_i(t), z_i(t), v_i, a, t), \tag{4.3b}$$

$$x_i(t_i) = h_i^x. \tag{4.3c}$$

If ∇g_{z_i} is nonsingular, the system (4.3) is an explicit ODE then it is called index one system. Then, the vectors $x_i(t)$ and $z_i(t)$ can be merged into one differentiable vector $y_i(t)$. If this is

not the case, using algebraic manipulations and coordinate transformations, the problem (4.3) can be written in the form of (4.1) with possibly different x and z vectors. We differentiate the resulting constraint equations again. If an explicit DAE results, we say that the original problem has index two. If the new system is not an explicit DAE, we repeat the process again. This process is finite and is repeated until explicit differential equations are obtained corresponding to the variables z. The number of differentiation steps required is known as the *index* of the DAE [30].

After coupling the state vectors $x(t)$ and $z(t)$ into a single differentiable vector, a piecewise polynomial function is used to parameterize the control $u_i(t)$ and piecewise constant functions for the parameters a, for a consistent representation (it may not be necessary if we consider a given $u(t)$ and a). These can be simply treated as additional polynomial states. Therefore, we can define a new state vector $y_i(t) \in \mathbb{R}^{(n_x+n_u+n_z+n_a)}$ so that:

$$\begin{pmatrix} \dot{x}_i(t) \\ \dot{z}_i(t) \\ \dot{u}_i(t) \\ \dot{a}_i(t) \end{pmatrix} = \begin{pmatrix} f(x_i(t), z_i(t), u_i, a, t) \\ \hat{g}(x_i(t), z_i(t), v_i, a, t) \\ \dot{p}_i^u(t) \\ 0 \end{pmatrix} \Longrightarrow \begin{pmatrix} \dot{y}_i(t) = \hat{f}(y_i(t), t) \\ y_i(t_i) = [h_i^x \;\; h_i^z \;\; v_i \;\; a(t_0)]^T \end{pmatrix} \tag{4.4}$$

where $t \in [t_i, t_{i+1}]$, $p_i^{(.)}$ denotes a piecewise polynomial function representation and $\hat{g}$ is a reformulated function which is derived from the function g (e.g. like the implicit equation (4.3b)). Since, in general, it is difficult to find an analytic solution of the IVP (4.4), the solution is usually done numerically. Therefore, many numerical algorithms have been developed for the solution of DAEs [8]. The effectiveness of these algorithms depends on the type of discretization strategy used in the algorithms.

For sake of convenience, on each interval $[t_i, t_{i+1}], i = 0, ..., N-1$ the following typical ODE

$$\begin{aligned} &\dot{y}_i(t) = f(y_i(t), t), t \in [t_i, t_{i+1}], \\ &y_i(t_{i0}) = y_{i0}, \end{aligned} \tag{4.5}$$

is considered, where $t_{i0} = t_i$ and $y_i, y_{i0} \in \mathbb{R}^{n_y}$ for $i = 0, ..., N-1$.

Definition 4.1 (ODE Discretiztion):
Given a constant $M > 1$, the time interval $[t_i, t_{i+1}]$ is divided into M equal subintervals $[t_{ik}, t_{i(k+1)}], k = 0, ..., M$ such that

$$t_i = t_{i0} < t_{i1} < ... < t_{iM} = t_{(i+1)0}. \tag{4.6}$$

The consideration of the ODE (4.5) at the time points $t_i, t_{i0}, t_{i1}, ..., t_{iM}$ is called as a sub-discretization of the ODE on the interval $[t_i, t_{i+1}]$.

The satisfaction of the *Lipschitz condition* is frequently required for error estimation and convergence statements of ODE discretization methods. And it is defined as:

Definition 4.2 (Lipschitz Condition and Lipschitz Constant):
A function $y(t)$ is said to satisfy the Lipschitz condition *at a point t_a if there is a constant $L > 0$ such that*

$$|y(t) - y(t_a)| \leq L|t - t_a|, \tag{4.7}$$

for all t in some neighborhood of t_a. The constant L is called the Lipschitz constant *for $y(t)$ in a neighborhood of t_a.*

In the following sections some well-known ODE discretization strategies are briefly summarized.

4.2 Euler Method

One of the simplest approaches for the numerical integration of ODEs is the *Euler* method which is classified as a *one-step method* and computes a discrete sequence of y at the arguments $t_{ik},\ k = 0, 1..., M,$ and $i = 0, 1, ..., N,$ (see Definition 4.1), using a difference equation [8, 9, 157, 167]

$$y_{i(k+1)} = y_{ik} + \xi f(y_{ik}, t_{ik}), \tag{4.8a}$$

or

$$y_{i(k+1)} = y_{ik} + \xi f(y_{i(k+1)}, t_{i(k+1)}), \tag{4.8b}$$

where $\xi = t_{i(k+1)} - t_{ik} = (t_{i+1} - t_i)/M$. Here, Eqs. (4.8a) and (4.8b) are called explicit and implicit Euler equations, respectively. If a fast ODE solution is needed, a small number M should be used but this causes an inaccurate approximation of $y_i(t)$.

Taking the first two terms of the Taylor series expansion of $y_i(t_{i(k+1)}) = y_i(t_k + \xi)$, it follows that

$$y_i(t_{ik} + \xi) = y(t_{ik}) + \xi f(t_{ik}, y(t_{ik})) + O(\xi^2), \tag{4.9}$$

where values y_{ik} and $y_{i(k+1)}$ are numerical approximations of $y_i(t_{ik})$ and $y_i(t_{ik} + \xi)$, respectively. The expression $O(\xi^2)$ represents the error term in the Euler method.

In general, the explicit one-step method can be written as

$$\begin{aligned} &y_{i(k+1)} = y_{ik} + \xi\Phi(t_{ik}, y_{ik}; \xi), \\ &y_i(t_{i0}) = y_{i0}, \\ &i = 0, 1, ..., N-1, \qquad k = 0, 1, ..., M-1. \end{aligned} \tag{4.10}$$

Definition 4.3 (Global Error [30]):
The global error from the approximation of the solution of $y_i(t)$ in Eq. (4.5) is defined by

$$e_{ik} = y_i(t_{ik}) - y_{ik},\ i = 0, 1, ..., N-1,\ k = 0, 1, ..., M. \tag{4.11}$$

Definition 4.4 (Truncation Error):
If a general one step method is used to approximate the function $y_i(t)$ in Eq. (4.5), then the truncation error of the one step method, $T_{ik}, i = 0, 1, ..., N,\ k = 0, 1, ..., M$, is defined by [92]

$$E_{ik} = \frac{y_i(t_{i(k+1)}) - y_i(t_{ik})}{\xi} - \Phi(t_{ik}, y_{ik}; \xi). \tag{4.12}$$

Definition 4.5 (Truncation Error of Euler method):
If the Euler method is used to approximate $y_i(t)$ in Eq. (4.5) using a step size $\xi \in [0, t_{i+1} - t_i]$, then the truncation error is given by:

$$E_{ik} = O(\xi) \tag{4.13}$$

This means that the truncation error of Eular method is order one.

Theorem 4.1 (Global Error in Terms of the Truncation Error):
If the function $f(y(t),t)$, $t \in [t_i, t_{i+1}]$; $i = 0, 1..., N$, is continuous of its arguments, Φ in Eq. (4.10) satisfies a Lipschitz condition with respect to y that is, there exists a positive constant L_Φ such that, for $0 \leq \xi \leq \xi_0$ and for all (t, α) and (t, β) in rectangular

$$D = \{(t, y) : t_0 \leq t \leq E_M, |y - y_0| \leq C\}$$

we have that

$$|\Phi(t, y; \Phi) - \Phi(t, \beta; \xi)| \leq L_\Phi |\alpha - \beta|.$$

Then assuming that $|y_k - y_0| \leq C$, $k = 1, 2..., M$, it follows that

$$|e_k| \leq \frac{E}{L_\Phi}(e^{L_\Phi(t_k - t_0)} - 1), \qquad k = 0, 1, ..., M. \tag{4.14}$$

where $E = \max_{0 \leq k \leq M-1} |E_k|$.

Proof. The proof of Theorem 4.1 can be found in [92] or [167]. This Theorem enables to estimate the global error based on the truncation error which can be computed.

4.3 Runge-Kutta Methods

Runge-Kutta (RK) methods are also considered as one-step methods, but have a high order truncation error of the Taylor series. Eq. (4.10) characterizes the general form of the one-step method. In simple Euler methods, only a first order of accuracy is obtained and we need to evaluate the function f only at one time point at (t_{ik}, y_{ik}). The objective of RK methods is to achieve higher accuracy by abdicating from the efficiency of Euler's method through re-evaluating the value of f at the intermediate points between t_{ik} and $t_{i(k+1)}$.

One of the most frequently used RK methods is known as a *classical fourth-order* method (RK4) which is characterized by

$$y_{i(k+1)} = y_{ik} + \frac{1}{6}\xi(\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4). \tag{4.15}$$

where the coefficients

$$\begin{aligned} \kappa_1 &= f(t_{ik}, y_{ik}) \\ \kappa_2 &= f(t_{ik} + \frac{1}{2}\xi, y_{ik} + \frac{1}{2}\xi\kappa_1) \\ \kappa_3 &= f(t_{ik} + \frac{1}{2}\xi, y_{ik} + \frac{1}{2}\xi\kappa_2) \\ \kappa_4 &= f(t_{ik} + \xi, y_{ik} + \xi\kappa_3) \end{aligned} \tag{4.16}$$

where κ_2 and κ_3 represent approximations to the derivative $\dot{y}$ at points on the solution curve, intermediate between t_{ik} and $t_{i(k+1)}$. For more detail on RK methods you can refer to [8, 9, 30, 34, 92, 157].

Definition 4.6 (Truncation Error of RK Methods):
If a RK method of order ρ is used to approximate $y_i(t)$ in Eq. (4.5) using a step size $\xi \in [0, t_{i+1} - t_i]$, this RK method has an order of accuracy ρ and the truncation error is given by:

$$T_{ik} = O(\xi^{\rho}) \tag{4.17}$$

In MUSCOD [103, 106] a Runge-Kutta method[1] is used to integrate the DAEs. Moreover, since RK methods divide the time subinterval (i.e. shooting time) into a finite number of small subintervals as shown in Fig. 4.1.

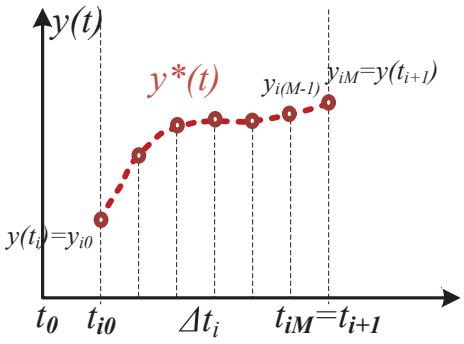

Figure 4.1: Integration using Runge-Kutta.

In the following sections we discuss the error and the efficiency of RK methods. We present a more accurate and well-known method to solve the DAEs which called *collocation on finite elements*. This higher accuracy leads to a high accuracy of state trajectories and their sensitivities. The sensitivity of terminal point of each shooting with respect to the initial point of the shooting in MUSCOD (i.e. $\frac{\partial y_{i+1}}{\partial y_i}$) can be computed using the chain-rule

$$\frac{\partial y_{i+1}}{\partial y_i} = \frac{\partial y_{iM}}{\partial y_{i0}} = \frac{\partial y_{iM}}{\partial y_{i(M-1)}} \times \frac{\partial y_{i(M-1)}}{\partial y_{i(M-2)}} \times \ldots \times \frac{\partial y_{i1}}{\partial y_{i0}}.$$

The RK method which is used in MUSCOD will be compared with the collocation method which will be used in this thesis to solve the state variables and their sensitivities.

4.4 Collocation on Finite Elements

A high precision approximation of $y_i(t)$ will be achieved using a collocation on finite elements (CFE) method [7]. In this method the time interval $[t_i, t_{i+1}]$ is considered as one *element* and the function $y_i(t)$ is approximated by a polynomial. The approximating polynomial in each element has to be satisfied exactly at a finite number of intermediate points which are called *collocation* points. That means the CFE method approximates a DAE solution in a time interval using an interpolating function with a number of intermediate

[1]The of RK method depends on the version of this software.

points. The derivative of this interpolating function should be equal to the derivative of the differential state $y_i(t)$ at the collocation points. As a result, we will obtain a system of nonlinear equations with anonymous variables of $y_i(t)$ at collocation points. We solve this system of nonlinear equations using one of the roots-finding techniques such as a Newton-Raphson method. To sketch the CFE method, the following definitions and theorems are needed.

Definition 4.7 (Time Segmentation):
The time interval $[t_i, t_{i+1}]$ is divided or segmented into M subintervals 'segments' $[t_{ik}, t_{i(k+1)}]$, therefore, each time point has two indices (i and k) such that

$$t_i = t_{i0} < t_{i1} < ... < t_{iM} = t_{(i+1)0} \tag{4.18}$$

where M is the number of collocation points and $k = 1, ..., M$. Fig 4.2 shows the time segmentation of the time horizon. Each interval $[t_i, t_{i+1}]$ is called a finite element with three collocation points, each finite element has M collocation points and the total number of mesh points in the time horizon are $(N + 1)$.

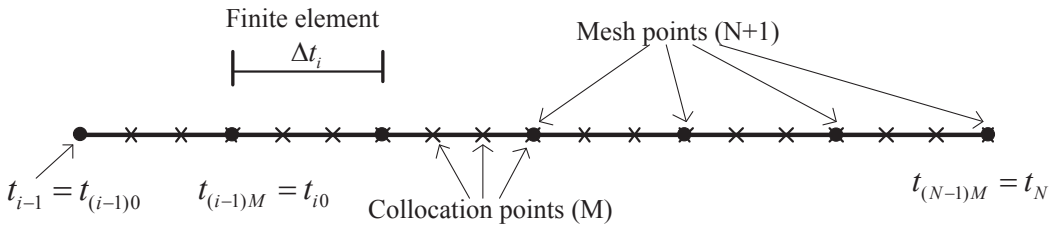

Figure 4.2: Time segmentation using collocation on finite elements.

We define the beginning of each finite element to be equal to the last collocation point of the previous element, that means

$$\begin{aligned} &t_{i0} = t_{(i-1)M}, \quad i = 1, ...N, \\ &t_{00} = t_0, \\ &t_{(N-1)M} = t_N = t_f. \end{aligned}$$

Theorem 4.2 (Weierstrass Approximation Theorem):
Suppose that $y(t)$ is continuous on $[t_i, t_{i+1}]$. For each $e > 0$, there exists a polynomial $P(t)$, with a property

$$|y(t) - P(t)| < e, \quad \forall t \in [t_i, t_{i+1}]$$

Proof. The proof of Theorem 4.2 can be found in most elementary texts on numerical analysis such as [14].

Theorem 4.3 (Orthogonal Function Set):
Suppose a set $S = \{p_0(t), p_1(t), ..., p_M(t)\}$ *has linearly independent functions* $p_k(t)$ *on* $[t_{i0}, t_{iM}]$, ω *is a weight function such that* $\omega(t) \neq 0,\ t \in [t_{i0}, t_{iM}]$. *A least-square approximation of the function* $y(t)$ *on* $[t_{i0}, t_{iM}]$ *is given by*

$$P(t) = \sum_{k=0}^{M} y_{ij} p_k(t)$$

if the set S *is an orthogonal set such as*

$$\int_{t_{i0}}^{t_{iM}} \omega(t) p_j(t) p_k(t) = \begin{cases} 0 & \text{if } k \neq k, \\ \alpha_k > 0 & \text{if } k = k. \end{cases} \tag{4.19}$$

Proof. The proof of Theorem 4.3 can be found in [34].

Theorem 4.4 (Interpolating Lagrange Polynomial):
If a function $y(t),\ \ t \in [t_{i0}, t_{iM}]$ *has* $M+1$ *distinct values at the numbers,* $t_{i0}, t_{i1}, ..., t_{iM}$, *then a unique polynomial* $P(t),\ \ t \in [t_{i0}, t_{iM}]$ *of degree* M *exists with*

$$y(t_{ik}) = P(t_{ik}), \qquad \text{for each } k = 0, 1, ...M.$$

where this polynomial is given by

$$P(t) = \sum_{j=0}^{M} y_i(t_{ij}) \cdot T_{ij}(t). \tag{4.20}$$

where T_{ij} *is a Lagrange polynomial for each* $j = 0, 1, ..., M$,

$$T_{ij}(t) = \prod_{\substack{k=0 \\ k \neq j}}^{M} \frac{t - t_{ik}}{t_{ij} - t_{ik}} \tag{4.21}$$

here indices $j = 0, 1, ..., M$ *denote the Lagrange polynomials and* $k = 0, 1, ..., M$ *denote the collocation points in each finite element* $[t_{i0}, t_{iM}]$.

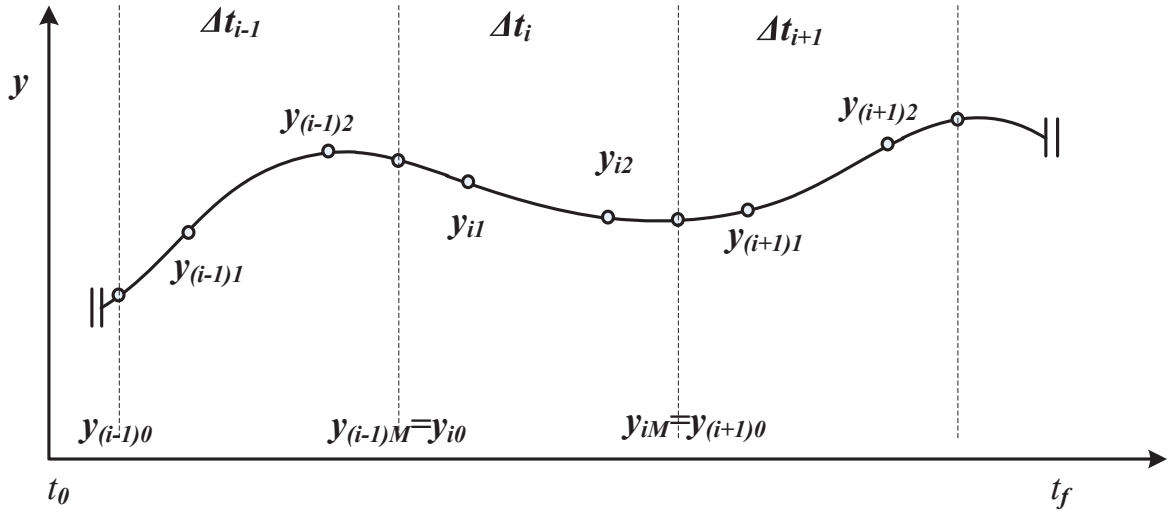

Figure 4.3: Principle of collocation on finite elements.

We use Theorem 4.4 to approximate the solution of the IVP (4.5). Fig. 4.3 shows the principle of collocation on 3 elements, where the number of collocation points in one element is $M = 3$. We assume that the number of finite elements is equal to the number of shoots for discretizing the dynamic system over the time horizon. An element means a time interval in which both the collocation and shooting will be carried out for the discretization.

Theorem 4.5 (Error Term of the Interpolating Lagrange Polynomial):
Suppose a function $y(t)$ is $M+1$ times differentiable and has the values at the collocation points, $t_{i0}, t_{i1}, ..., t_{iM}$, then for each $t \in [t_{i0}, t_{iM}]$, there is $\xi(t) \in [t_{i0}, t_{iM}]$ such as

$$y(t) = P(t) + \frac{f^{(M+1)}(\xi(t))}{(M+1)!} \prod_{k=0}^{M} (t - t_{ik}) \tag{4.22}$$

where $P(t)$ is the interpolating polynomial given in Eq. (4.20) in Theorem 4.4. The error formula of the Lagrangian polynomial of degree M is given by

$$e(t) = \frac{f^{(M+1)}(\xi(t))}{(M+1)!} \prod_{k=0}^{M} (t - t_{ik}) \tag{4.23}$$

Proof. The proof of Theorem 4.5 can be found in [9, 34].

To compute the approximating function $P(t)$ we have to determine the Lagrange polynomial T_{ij} and to solve the state values at the collocation points t_{ij}. We take the roots of Legendre polynomials as collocation points [9, 34, 77, 141]. In addition, to determine the values $y_i(t_{ij})$ from the IVP (4.5), we need to rewrite Eq. (4.20) and combine Eq. (4.5)

with Eq. (4.20), then we get

$$(\sum_{j=1}^{M} \dot{T}_{ij}(t).y_i(t_{ij})) + \dot{T}_{i0}(t).y_i(t_{i0}) - f(y_i(t), t) = 0 \tag{4.24}$$

Substituting the M collocation points in Eq. (4.24) and writing it in a matrix form leads to:

$$F(Y_{ik}, Y_{i0}) = \dot{W}_{ik} Y_{ik} + \text{diag}\left(\dot{W}_{i0}\right) Y_{i0} - f_i(y_i(t_{ik})) = 0 \tag{4.25}$$

where $W_{ik} = \begin{bmatrix} T_{i1}(t_{i1}) & \cdots & T_{iM}(t_{i1}) \\ \vdots & \ddots & \vdots \\ T_{i1}(t_{iM}) & \cdots & T_{iM}(t_{iM}) \end{bmatrix}$, $W_{i0} = \begin{bmatrix} T_{i1}(t_{i0}) \\ \vdots \\ T_{iM}(t_{i0}) \end{bmatrix}$, $Y_{ik} = \left[y_i(t_{i1}), \cdots, y_i(t_{iM})\right]^T$ and $Y_{i0} = \left[y_i(t_{i0}), \cdots, y_i(t_{0M})\right]^T \in \mathbb{R}^M$. We solve now the nonlinear Eq. (4.25) for $y_i(t_{i1}), \cdots, y_i(t_{iM})$, by using the Newton-Raphson methods as described below.

Newton-Raphson Method

The Newton-Raphson method is one of numerical methods to solve a general nonlinear algebraic equation system:

$$F(Y) = 0 \tag{4.26}$$

where $F : \mathbb{R}^{n_Y} \to \mathbb{R}^{n_Y}$. Newton-Rapshon method uses a derivative based method and tries to improve an initial guess for a solution vector iteratively using a linearization procedure. We use an index l to denote a Newton-Raphson iteration. If there exists a present initial guess, $Y^l = Y^0$, for the solution of Eq. (4.26), the function F is written for Y as a Taylor series expansion about Y^l as follows:

$$F(Y^l) + \frac{\partial F(Y^l)}{\partial Y}(Y^{l+1} - Y^l) \cong 0 \tag{4.27}$$

where $\frac{\partial F(Y^l)}{\partial Y} \in \mathbb{R}^{n_Y \times n_Y}$ is a Jacobian and can be evaluated analytically by using first patrials for each entry. The Newton-Rhaphson method will converge and quit the iterations if the following two criterions are satisfied

$$\begin{aligned} |Y^l - Y^{l-1}| &\leq \varepsilon_1 \\ |F^l(Y^l)| &\leq \varepsilon_2 \end{aligned} \tag{4.28}$$

where ε_1 and ε_2 are Newton-Raphson method tolerances be determined by the user. Systems of nonlinear equations may have local solutions and it may fail, thus the Newton-Raphson method may converge to different solutions depending on an initial guess. In that case, it is suggested to use physically reasonable initial guess. A Newton-Raphson Algorithm 3 is used to solve a system of nonlinear equations.

Note that the inverse of the Jacobian matrix information is needed in each iteration to compute the difference $\Delta Y = [\frac{\partial F(Y)}{\partial Y}]^{-1} F(Y)$, which is almost impossible since it is time-consuming for large-scale systems. Thus, either direct direct factorization-based techniques or iterative linear algebra techniques ar used to solve such large systems of linear equations [55, 150, 173].

Algorithm 3 Newton-Raphson algorithm

Input: Nonlinear function $F(Y)$, Jacobian $[\frac{\partial F(Y)}{\partial Y}]$, initial guess Y^0, tolerances (ε_1 and ε_2), maximum number of iterations .
Output: Approximate solution Y.
Step 1: Set $l = 1$.
Step 2: If l is less that the number of iterations, do Steps 3-6.
Step 3: Set $Y = Y^0 - [\frac{\partial F(Y)}{\partial Y}]^{-1} F(Y)$
Step 4: If $|Y - Y^0| < \varepsilon_1$ and $|F(Y)| < \varepsilon_2$
Output Y.
Stop
Step 5: Set $l = l + 1$.
Step 6: Set $Y^0 = Y$.
Else :
Stop.

Theorem 4.6 (Truncation Error of CFE using Legendre Roots):
If the roots of the shifted Legendre polynomials are considered as collocation points, t_{ik}, $i = 0, 1..., N-1$, $k = 1, ..., M$ in Eq. (4.21), then the truncation error of the approximated solution is $O(\xi^{2M})$, where ξ is the step size between two collocation points and M is the total number of the collocation points which is equivalent to the order of the Legendre polynomial.

Proof. The proof of Theorem 4.6 can be found in [53].

To investigate the result of Theorem 4.6 for solving the IVP (4.5), the step size ξ must be less than one. This can be guaranteed by transforming the time interval from $[t_0, t_f]$ into $[0, 1]$ if $t_f - t_0 > 1$. Algorithm 4, *collocation on finite elements* (CFE), is considered to solve the IVP (4.5).

When we use a piecewise constant parametrization for the control vector $u(t)$ and since the parameter vector a is time invariant, there is no need to define M collocation points for the control and the parameter vectors in each finite element. Furthermore, the algebraic state $z(t)$ can be dealt without reformulation into differential state as in Eq. (4.4) since all states (algebraic and differential) are approximated with the Lagrange polynomials when the collocation method is used. Thus we can now decouple the state variables ($x(t)$ and $z(t)$), the control variables $u_i(t)$ and parameters a from each other, i.e. we return to IVP:

$$\begin{aligned} \dot{x}_i(t) &= f(x_i(t), z_i(t), v_i, a, t), && \forall t \in [t_i, t_{i+1}), \\ 0 &= g(x_i(t), z_i(t), v_i, a, t), && \\ x_i(t_i) &= h_i^x && i = 0, 1, ..., N-1. \end{aligned}$$

Algorithm 4 Collocation on finite elements

Input:Time interval $[t_0, t_f]$, ordinary differential equation $\dot{y}(t) = f(y,t)$, Jacobian matrix $[\partial\dot{y}/\partial y]$, initial condition $y(t) = y_0$, total number of elements N, total number of the collocation points M, tolerances of Newton-Raphson (ε_1 and ε_2), maximum number of the Newton-Raphson iterations.

Output: Approximate solution $P(t) \cong y(t)$.

Step 1: ***If*** $t_f - t_0 > 1$

Step 2: Transform the time interval from $[t_0, t_f]$ into $[0, 1]$.

Step 3: Use the time segmentation method (Definition 4.7) according the roots of shifted Legendre polynomials to construct the sequence $\{t_{ik}\}$ where $i = 0, 1..., N$ and $k = 0, 1, ..., M$.

Step 4: Initialize $Y_{00} = y_0$, $P(t) = 0$.

Step 5: ***For*** $i = 0 : N - 1$

Step 6: Define collocation points (t_{ik}, y_{ik}).

Step 7: Construct Eq. (4.25).

Step 8: Use the Jacobian matrix $[\frac{\partial \dot{y}}{\partial y}]$ to find the matrix $[\frac{\partial F_i}{\partial Y_{ik}}]$.

Step 9: Solve $F_i(Y_{ik}) = 0$ using Algorithm 3.

Step 10: Put $P(t) = P(t) + \sum_{j=0}^{M}(\prod_{\substack{k=0 \\ k\neq j}}^{M} \frac{t-t_{ik}}{t_{ij}-t_{ik}}) \cdot y_i(t_{ij})$.

Step 11: Put $Y_{(i+1)0} = Y_{iM}$

Step 12: Return the transformation $[t_0, t_f] \leftarrow [0, 1]$

Step 12: Output $y(t) \cong P(t)$.

Stop.

Therefore the matrix Eq. (4.25) can be rewritten as:

$$F_i^x(X_{ik}, Z_{ik}) = \dot{W}_{ik}(t)X_{ik} + \text{diag}\left(\dot{W}_{i0}\right) X_{i0} - f_i(x_i(t_{ik}), z_i(t_{ik}), v_i, a) = 0 \tag{4.29a}$$

$$F_i^z(X_{ik}, Z_{ik}) = g(X_{ik}, X_{i0}, Z_{ik}, Z_{i0}, v_i, a) = 0 \tag{4.29b}$$

where $W_{ik} = \begin{bmatrix} T_{i1}(t_{i1}) & \cdots & T_{iM}(t_{i1}) \\ \vdots & \ddots & \vdots \\ T_{i1}(t_{iM}) & \cdots & T_{iM}(t_{iM}) \end{bmatrix}$, $W_{i0} = \begin{bmatrix} T_{i1}(t_{i0}) \\ \vdots \\ T_{iM}(t_{i0}) \end{bmatrix}$, $X_{ik} = \left[x_i(t_{i1}), \cdots, x_i(t_{iM})\right]^T$, $X_{i0} = \left[x_i(t_{i0}), \cdots, x_i(t_{i0})\right]^T = \left[h_i^x, \cdots, h_i^x\right]^T \in \mathbb{R}^M$, $Z_{ik} = \left[z_i(t_{i1}), \cdots, z_i(t_{iM})\right]^T$, $Z_{i0} = \left[z_i(t_{i0}), \cdots, z_i(t_{i0})\right]^T = \left[h_i^z, \cdots, h_i^z\right]^T \in \mathbb{R}^M$,
$F_i^x : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}^M$ and $F_i^z : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}^M$. Therefore, $F_i(X_{ik}, Z_{ik}) = [F_i^{xT}(X_{ik}, Z_{ik}) \quad F_i^{zT}(X_{ik}, Z_{ik})]^T = 0$, where $F_i : \mathbb{R}^{2M} \to \mathbb{R}^{2M}$. Then the differential and algebraic states will be

$$\begin{aligned} x_i(t) =& W_{ik}(t)X_{ik} + W_{i0}I_M X_{i0}, \\ z_i(t) =& W_{ik}(t)Z_{ik} + W_{i0}I_M Z_{i0}, \end{aligned}$$

respectively. Accordingly, Algorithm 5 is needed to solve the discretized nonlinear equations in the resulted NLP from the multiple shooting approach when the piecewise constant parametrization is used for the control and parameter vectors.

Algorithm 5 Collocation on finite elements using a piece-wise constant parametrization for controls and the parameters

Input:Time interval $[t_0, t_f]$, DAEs , algebraic state equation $g(x, z, u, a, t) = 0$, Jacobian matrices $[\frac{\partial \dot{x}}{\partial x}]$, $[\frac{\partial \dot{x}}{\partial z}]$, $[\frac{\partial g}{\partial x}]$ and $[\frac{\partial g}{\partial z}]$, initial condition $x(t) = x_0$, piecewise constant parametrization for u and a, number of elements N, number of the collocation points M, tolerances of Newton-Raphson (ε_1 and ε_2) and maximum number of the Newton-Raphson iterations.

Output: Approximate solutions $P_x(t) \cong x(t)$ and $P_z(t) \cong z(t)$.

Step 1: ***If*** $t_f - t_0 > 1$

Step 2: Transform the time interval from $[t_0, t_f]$ into $[0, 1]$.

Step 3: Use the time segmentation method (Definition 4.7) according the roots of shifted Legendre polynomials to construct the sequence $\{t_{ik}\}$ where $i = 0, 1..., N$ and $k = 0, 1, ..., M$.

Step 4: Initialize $X_{00} = x_0$, $P_x(t) = 0$ and $P_z(t) = 0$.

Step 5: ***For*** $i = 0 : N - 1$

Step 6: Define collocation points (t_{ik}, x_{ik}) and (t_{ik}, z_{ik}).

Step 7: Construct the Eq. (4.29).

Step 8: Use the Jacobian matrices $[\frac{\partial \dot{x}}{\partial x}]$, $[\frac{\partial \dot{x}}{\partial z}]$, $[\frac{\partial g}{\partial x}]$ and $[\frac{\partial g}{\partial z}]$ to find the Jacobian matrices $[\frac{\partial F_i}{\partial X_{ik}}]$, $[\frac{\partial F_i}{\partial Z_{ik}}]$, $[\frac{\partial F_i}{\partial X_{ik}}]$ and $[\frac{\partial F_i}{\partial Z_{ik}}]$.

Step 9: Solve $F_i(X_{ik}, Z_{ik}) = 0$ using Algorithm 3.

Step 10: Put $P_x(t) = P_x(t) + \sum_{j=0}^{M} (\prod_{\substack{k=0 \\ k \neq j}}^{M} \frac{t - t_{ik}}{t_{ij} - t_{ik}}) \cdot x_i(t_{ij})$ and

$$P_z(t) = P_z(t) + \sum_{j=0}^{M} (\prod_{\substack{k=0 \\ k \neq j}}^{M} \frac{t - t_{ik}}{t_{ij} - t_{ik}}) \cdot z_i(t_{ij}).$$

Step 11: Put $X_{(i+1)0} = X_{iM}$, $Z_{(i+1)0} = Z_{iM}$.

Step 12: Return the transformation $[t_0, t_f] \leftarrow [0, 1]$

Step 13: Output: $x(t) \cong P_x(t)$ and $z(t) \cong P_z(t)$.

Stop.

Since $x \in \mathbb{R}^{n_x}$ and $z \in \mathbb{R}^{n_z}$, the dimensions of the variables are $X_{ik} \in \mathbb{R}^{Mn_x}$, $Z_{ik} \in \mathbb{R}^{Mn_z}$ and $Z_{i0} \in \mathbb{R}^{M}$ and the function $F_i(X_{ik}, Z_{ik})$ is defined as $F_i : \mathbb{R}^{Mn_x + Mn_z} \rightarrow \mathbb{R}^{Mn_x + Mn_z}$.

4.5 Error Analysis of ODE Solver

An accurate solution of ODE for $y_i(t)$, $i = 0, 1, , N - 1$, will provide an accurate solution of the trajectory $y(t)$ and accurate sensitivities $\frac{\partial x(t_{i+1})}{\partial x_i}$, $\frac{\partial x(t_{i+1})}{\partial z_i}$,$\frac{\partial x(t_{i+1})}{\partial u_i}$ and $\frac{\partial x(t_{i+1})}{\partial a}$, respectively. In this section, the accuracy of the ODE solution with two different methods will be analyzed. It can be concluded that a more accurate ODE solution will be obtained by using the CFE method.

For the error analysis the interval Δt_i is selected, as shown in Fig. 4.4. If $y(t)$, $t \in [t_i, t_{i+1}]$ is solved using constant controls $u(t_i)$ and parameters a_i, the trajectory $x^*(t)$ will approximate $x(t)$ with an error bound (tolerance) $e(t)$.

Fig. 4.5 shows the integration using the explicit RK method in one time interval. This method is used for the integration of the ODEs in the existing multiple shooting codes

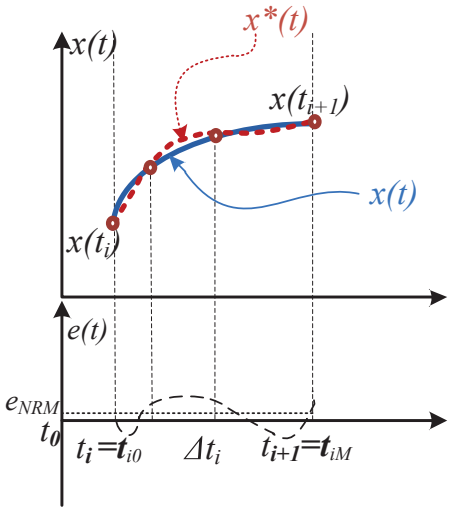

Figure 4.4: Error bound using piecewise constant control.

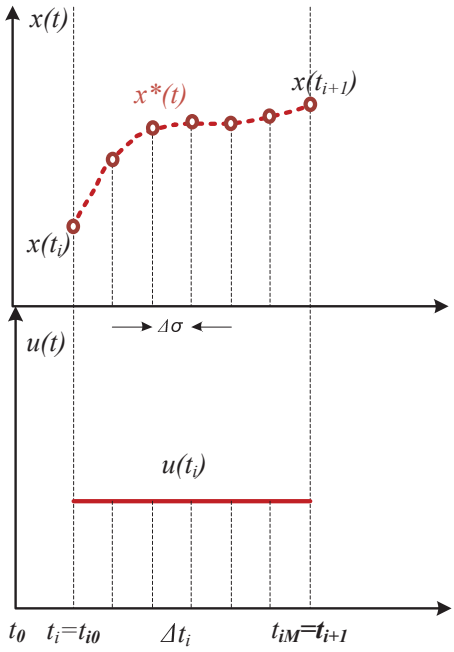

Figure 4.5: State solution within one time interval in MUSCOD.

like MUSCOD. Each subinterval Δt_i is divided into M equal subintervals where $t_i = t_{i0} < t_{i1} < ... < t_{iM} = t_{i+1}$ and $\Delta\sigma = t_{i(j+1)} - t_{ij}, \quad j = 0, 1..., M, \quad i = 0, 1, ..., N-1$. When $\Delta\sigma$ is chosen too small, i.e. even the value of M and/or N are large, the local truncation error $e(t)$ becomes small, but the computation will be expensive [103].

The solution formulation obtained by RK methods has local truncation error which

was addressed in its classical version by Stetter [165] and Zadunaisky [185].The idea of this formulation becomes the basis for the acceleration technique known as *iterated defect correction* [80, 98]. The classical error estimate due to Stetter works satisfactorily at the mesh points (i.e., when the solution values at the grid points are disregarded) but fails to be asymptotically correct for finer grids including grid points.

The local truncation error at the points $(t_{i(k+1)}, y(t_{i(k+1)}))$ of the R-*th* order RK solver as shown in Definition 4.6 is given by $O(\xi^{M+1})$ [11, 35, 155, 159]. However, if a fast solution is needed, i.e. $\Delta\sigma$ is large or a small number of sub-subintervals M, we have a significant truncation error at the points $(t_{ik}, y(t_{ik}))$, $k = 0, 1, ..., M$, $t_{ik} \in [t_i, t_{i+1}]$. This error will be increased at points $(t, y^*(t))$, $t \in [\delta_k, \delta_{k+1}]$ due to the RK method itself. In addition, the error in the sensitivities $\frac{\partial x(t_{i+1})}{\partial x_i}$, $\frac{\partial x(t_{i+1})}{\partial z_i}$, $\frac{\partial x(t_{i+1})}{\partial u_i}$ and $\frac{\partial x(t_{i+1})}{\partial a}$ will also increase. For this reason, a small $\Delta\sigma$ is always chosen in MUSCOD [103] to guarantee high accuracy integration.

The original idea of the *defect correction* was presented by Zadunaisky [185] is that "*when the numerical integration of the given system $\dot{y} = f(y, t)$ generates the approximate values then the integration interpolates sets of $M + 1$ successive componentwise by polynomials of degree M. The asymptotic orders of the iteration steps turned out to be $O(\xi^{2M}), O(\xi^{3M}), ...,$ where M denotes the order of the method, the global error of which is estimated in the single iteration steps*" [1, 10].

Furthermore, when the CFE method is used to approximate the trajectory vector $y(t)$, the subinterval Δt_i will have M collocation points as shown in Fig. 4.3, and the system of algebraic equations Eq. (4.25) will be solved. The solution of ODE has an error due to solving the nonlinear algebraic system, ε_{NRM}, at the points $(\delta_k, y^*(\delta_k))$ as shown in Fig 4.4[1].

Using a larger number of M leads to a more accurate trajectory $y(t)$ and makes the size of the nonlinear algebraic equation system larger, thus the solution of this resulted system will take more computation time. In addition, the location of the collocation points will affect the error function $e(t)$ [1, 35, 50]. Several schemes were presented in the literature in order to compromise between the error, the number and placement of collocation points. Cuthrell and Bigler [50, 52] imposed an error minimization strategy to obtain accurate subinterval solution by locating the collocation points on the subinterval over which the error will be minimized and leads to accurate solution of the ODE. Russell and Christian [149] presented a strategy for the development of the subinterval placement to choose a 'good' distribution of the subintervals. Ascher *et al* [7] used iterative extrapolation to limit the approximation error. DeBoor [54] presented a method to select the subintervals in outer loop and solve the collocation equation, then repositioning the subintervals until obtaining a better approximation. A far reaching result was proved by deBoor and Swartz [53]. They considered high degree polynomials and used the zeros of the Legendre polynomials [77, 141] and proved that the method of CFE is equivalent to the fully-implicit RK method under certain circumstances. First the collocation is done at the Gaussian roots and second the elements of Butcher's block array are specified in certain way for the RK method and the error in the function and in its derivative at the end of each elements is $O(\xi^{2M})$. In addition, the method converges since the error can be made as small as desired by adding more elements. For more information you may refer to [1, 51].

[1]The maximum tolerance of the Newton method of all case studies in this work will be controlled within the square root of the machine precision [11].

Based on above results, if the IVP (4.5) is solved by dividing the interval $[t_i, t_{i+1}]$ into M subintervals by using RK and CFE methods, the accuracy in each case will be $O(\xi^{\rho+1})$ and $O(\xi^{2M})$, respectively. Accordingly, an additional accuracy will be obtained by CFE if $2M > \rho + 1$ and the same step size ($\xi < 1$) is selected.

Example 4.1 (A Simple Demonstrative IVP):
Here, a demonstrative example that shows the high accuracy of CFE and compares the integrator error with RK is considered:

$$\begin{aligned} &\dot{z}(z,t) = z, \quad t \in [0 \;\; 1] \\ &z(0) = 1 \end{aligned} \tag{4.30}$$

The analytical solution for this IVP is $z(t) = \exp(t)$. *We use CFE and the RK method (order 4) to compute the numerical solution of this problem. For comparison, first, we select the mesh grid* $M = 3$ *for* $t \in [0 \;\; 1]$. *The maximum local truncation error at the mesh points using CFE and RK4 are* 1×10^{-9} *and* 1.23×10^{-3}, *respectively, while the maximum global truncation error using both methods are* 4.2×10^{-3} *and* 3.204×10^{-2}, *respectively. The CPU time*[2] *of this example using CFE was* 19μ*seconds. The problem was solved by the RK method with the same accuracy and the CPU time was* 28μ*second.*

[2]The simulation is done using "Pentium 4", 3 GHz and 1G Byte RAM.

5 A Combined Multiple Shooting-Collocation Method

5.1 Introduction

In this Chapter we will introduce the main algorithm of this thesis which is a combination of multiple shooting and CFE. In Section 3.1 the optimal control problem Eq. (2.2) is converted into a finite dimensional NLP problem (3.9) using the direct multiple shooting approach. To solve the resulted IVPs in this NLP problem, we use collocation on finite elements as addressed in Section 4.4. In addition, we compute the gradients (sensitives) information for the NLP solution as shown in Fig. 3.4 by taking a Taylor expansion of the functions of the state variables. In Section 5.2 an elementary theory to drive the main algorithm will be presented.

5.2 ODE Solution and Sensitives

To solve the resulted NLP problem described in Section 3.2 we need to compute all of the information needed for each SQP iteration as described in Fig 3.4. To avoid the computation of integrals we redefine the Lagrange term of the objective function as an additional state equation. That means the objective function will be the summation of the Mayer term and the terminal value of the additional state. Therefore the nonlinear optimal control problem can be rewritten as:

$$\min_{u(t),x(t),z(t),a} J = E(x(t_f), z(t_f), a, t_f) + x^{(\mathrm{Lag})}(t_f). \tag{5.1a}$$

subject to

$$\dot{x}^{(\mathrm{Mod})}(t) = f(x(t), u(t), z(t), a, t), \qquad t \in [t_0, t_f], \tag{5.1b}$$
$$\dot{x}^{(\mathrm{Lag})}(t) = L(x(t), z(t), u(t), a, t), \qquad t \in [t_0, t_f], \tag{5.1c}$$
$$g(x(t), u(t), z(t), a, t) = 0, \qquad t \in [t_0, t_f], \tag{5.1d}$$
$$s(x(t), z(t), u(t), a, t) \geq 0, \qquad t \in [t_0, t_f], \tag{5.1e}$$
$$x^{(\mathrm{Mod})}(t_0) = x_0^{(\mathrm{Mod})}, \tag{5.1f}$$
$$x^{(\mathrm{Lag})}(t_0) = x_0^{(\mathrm{Lag})} = 0, \tag{5.1g}$$
$$x_0^T = [{x^{(\mathrm{Mod})}}_0^T \;\; {x^{(\mathrm{Lag})}}_0^T], \tag{5.1h}$$
$$r_e(x(t_f), z(t_f), a, t_f) = 0, \tag{5.1i}$$
$$r_i(x(t_f), z(t_f), a, t_f) \geq 0, \tag{5.1j}$$

$$x_{\min} \leq x(t) \leq x_{\max}, \tag{5.1k}$$
$$z_{\min} \leq z(t) \leq z_{\max}, \tag{5.1l}$$
$$u_{\min} \leq u(t) \leq u_{\max}, \tag{5.1m}$$
$$a_{\min} \leq \quad a \quad \leq a_{\max}. \tag{5.1n}$$

where $x^{(\mathrm{Mod})}(t) \in \mathbb{R}^{n_x}$, $x^{(\mathrm{Lag})}(t) \in \mathbb{R}$, are the state vector and the derivative of the Lagrangian term of the original objective function, respectively, $x^{(\mathrm{Mod})}(t_0) \in \mathbb{R}^{n_x}$ is an initial state vector, $x^{(\mathrm{Lag})}(t_0) \in \mathbb{R}$ is the initial value of the Lagrange term and normally is equal to zero, $x(t) = [{x^{(\mathrm{Mod})}}^T(t) \;\; {x^{(\mathrm{Lag})}}^T(t)]^T \in \mathbb{R}^{n_x+1}$, $z(t) \in \mathbb{R}^{n_z}$, $u(t) \in \mathbb{R}^{n_u}$ and $a \in \mathbb{R}^{n_a}$. The optimal control problem (5.1) is discretized using the multiple shooting approach stated in Chapter 3, therefore we yield the NLP problem of form (3.9). In addition, in the equality constraints $h^x_{i+1} - x_i(h^x_i, h^z_i, v_i, a, t_{i+1}) = 0$ and $g(h^x_i, h^z_i, v_i, a, t_i) = 0$ the information h^x_i and h^z_i is needed which will be provided by solving the IVPs (4.4). Now in the IVP (4.4) the dimension of the vector $y_i(t) \in \mathbb{R}^{(n_x+n_u+n_z+n_a+1)}$.

5.2.1 Sensitivity Calculations

Required by the NLP solver, we evaluate the sensitivities $\frac{\partial B(w)}{\partial w}$ by calculating the Jacobian $\frac{\partial y_i(t_{i+1})}{\partial y_i(t_i)} = \frac{\partial y_i(t_{iM})}{\partial y_i(t_{i0})}$ by the first order Taylor-expansion of Eq. (4.24)

$$\dot{W}_{ik}\frac{\partial Y_{ik}}{\partial Y_{i0}} + \mathrm{diag}\left(\dot{W}_{i0}\right) - \frac{\partial f_i(y_i(t_{ik}))}{\partial Y_{i0}} = 0 \tag{5.2}$$

where $\frac{\partial Y_{ik}}{\partial Y_{i0}}$ and $\frac{\partial f_i(y_i(t_{ik}))}{\partial Y_{i0}}$ are Jacobian matrices with dimension $M \times M$. In addition, the last point of an element is used as the initial point to the next element, i.e. $t_{iM} = t_{(i+1)0}$. Defining $\frac{\partial Y_{ik}}{\partial Y_{i0}} = \Psi_{ik}$, we have

$$\dot{W}_{ik}\Psi_{ik} + \mathrm{diag}\left(\dot{W}_{i0}\right) - \frac{\partial f_i(y_i(t_{ik}))}{\partial Y_{ik}}\Psi_{ik} = 0 \tag{5.3}$$

or equivalently

$$\Psi_{ik} = -[\dot{W}_{ik} - \frac{\partial f_i(y_i(t_{ik}))}{\partial Y_{ik}}]^{-1}\mathrm{diag}\left(\dot{W}_{i0}\right) \tag{5.4}$$

In fact, Eq. (5.3) is a linear equation system and thus can be solved by a LU factorization using the forward and backward substitution [86]. The dimension of the Jacobian Ψ_{ik} is $(Mn_y) \times (Mn_y)$. From the computed value Ψ_{iM} the Jacobians can be gained as

$$\frac{\partial y_i(t_{iM})}{\partial y_i(t_{i0})} = \begin{bmatrix} I_M & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \Psi_{0M} & & I_M & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \Psi_{1M} & & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \Psi_{2M} & & I_M & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \Psi_{(N-1)M} & I_M \end{bmatrix}, \tag{5.5}$$

where I_M is a unit matrix. For the piecewise constant form of the control and the parameter variables, there is no need to define M collocation points for the controls and parameters.

Thus we can write Eq. (4.24) by decoupling the state variables from controls, i.e. we define the vector $\gamma(t)$ instead of $y(t)$ and vector q which combines the control vector v_i and the parameter vector a, such as

$$\begin{pmatrix}\dot{x}_i(t)\\ \dot{z}_i(t)\end{pmatrix} = \begin{pmatrix}f(x_i(t), z_i(t), v_i, a, t)\\ \hat{g}(x_i(t), z_i(t), v_i, a, t)\end{pmatrix} \Longrightarrow \begin{pmatrix}\dot{\gamma}_i(t) = \hat{f}(\gamma_i(t), q_i, t)\\ \gamma_i(t_i) = [h_i^{xT}\ h_i^{zT}]^T\\ q_i = [v_i{}^T\ a^T]^T\end{pmatrix}$$

where $\gamma_i(t) \in \mathbb{R}^{n_\gamma}$, $q_i \in \mathbb{R}^{n_q}$, $n_\gamma = n_x + n_z + 1$ and $n_q = n_u + n_a$. Then we define Γ_{i0} and Γ_{ik} instead of Y_{i0} and Y_{ik} , respectively, so that

$$S(\Gamma_{i0}, \Gamma_{ik}, q_i) = \dot{W}_{ik}\Gamma_{ik} + \operatorname{diag}\left(\dot{W}_{i0}\right)\Gamma_{i0} - f_i(\gamma_i(t_{ik}), q_i) = 0 \tag{5.6}$$

where $\Gamma_{ik} = \begin{bmatrix}\gamma_i(t_{i1}) & \cdots & \gamma_i(t_{iM})\end{bmatrix}^T$ and $\Gamma_{i0} = \begin{bmatrix}\gamma_i(t_{i0}) & \cdots & \gamma_i(t_{i0})\end{bmatrix}^T$.
The first order Taylor-expansion of Eq. (5.6) leads to

$$\frac{\partial S}{\partial \Gamma_{i0}}\Delta\Gamma_{i0} + \frac{\partial S}{\partial \Gamma_{ik}}\Delta\Gamma_{ik} + \frac{\partial S}{\partial q_i}\Delta q_i = 0 \tag{5.7}$$

According Eq. (5.7), the sensitivities can be computed by

$$\frac{\Delta\Gamma_{ik}}{\Delta\Gamma_{i0}} \cong \frac{\partial\Gamma_{ik}}{\partial\Gamma_{i0}} = -\left(\frac{\partial S}{\partial \Gamma_{ik}}\right)^{-1}\frac{\partial S}{\partial \Gamma_{i0}} \tag{5.8a}$$

$$\frac{\Delta\Gamma_{ik}}{\Delta q_i} \cong \frac{\partial\Gamma_{ik}}{\partial q_i} = -\left(\frac{\partial S}{\partial \Gamma_{ik}}\right)^{-1}\frac{\partial S}{\partial q_i} \tag{5.8b}$$

where $\{S : \mathbb{R}^{n_\gamma} \times \mathbb{R}^{M(n_\gamma)} \times \mathbb{R}^{n_q} \to \mathbb{R}^{M(n_\gamma)}\}$. From the solutions of Eq. (5.8), the last column of the Jacobian matrices $\frac{\partial\Gamma_{ik}}{\partial\Gamma_{i0}} = \begin{bmatrix}\frac{\partial\gamma_{i1}}{\partial\gamma_{i0}} & \cdots & \frac{\partial\gamma_{iM}}{\partial\gamma_{i0}}\end{bmatrix}^T$ and $\frac{\partial\Gamma_{ik}}{\partial q_i} = \begin{bmatrix}\frac{\partial\gamma_{i1}}{\partial q_i} & \cdots & \frac{\partial\gamma_{iM}}{\partial q_i}\end{bmatrix}^T$ lead to the sensitivity $\frac{\partial B}{\partial w}$. The sensitivity $\frac{\partial C}{\partial w}$ can be obtained from the direct differentiation of the states with respect to w.

The sensitivities have to be computed from time interval to time interval, therefore we consider the dimensions of the state and control vectors where $\gamma_i = \begin{bmatrix}\gamma_i^1 & ... & \gamma_i^{n_\gamma}\end{bmatrix}^T = \begin{bmatrix}x_i^1 & ... & x_i^{n_x+1} & z_i^1 & ... & z_i^{n_z}\end{bmatrix}^T$, $S = \begin{bmatrix}S^1 & ... & S^{n_\gamma}\end{bmatrix}^T = [S^1\ ...\ S^{n_x+1}\ S^{n_x+2}\ ...\ S^{n_x+n_z+1}]^T$ and $q = [q^1\ ...\ q^{n_q}]^T = [v_i^1\ ...\ v_i^{n_u}\ \ a^1\ ...\ a^{n_a}]^T$ in each interval i, where $i = 0, 1, ..., N-1$. Hence, we have the Jacobians

$$\frac{\partial S}{\partial \Gamma_{i0}} = \begin{bmatrix}
\frac{\partial S^1}{\partial x_{i0}^1} & \cdots & \frac{\partial S^1}{\partial x_{i0}^{n_x+1}} & \frac{\partial S^1}{\partial z_{i0}^1} & \cdots & \frac{\partial S^1}{\partial z_{i0}^{n_z}}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
\frac{\partial S^{n_x+1}}{\partial x_{i0}^1} & \cdots & \frac{\partial S^{n_x+1}}{\partial x_{i0}^{n_x+1}} & \frac{\partial S^{n_x+1}}{\partial z_{i0}^1} & \cdots & \frac{\partial S^{n_x+1}}{\partial z_{i0}^{n_z}}\\
\frac{\partial S^{n_x+2}}{\partial x_{i0}^1} & \cdots & \frac{\partial S^{n_x+2}}{\partial x_{i0}^{n_x+1}} & \frac{\partial S^{n_x+2}}{\partial z_{i0}^1} & \cdots & \frac{\partial S^{n_x+2}}{\partial z_{i0}^{n_z}}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
\frac{\partial S^{n_x+n_z+1}}{\partial x_{i0}^1} & \cdots & \frac{\partial S^{n_x+n_z+1}}{\partial x_{i0}^{n_x+1}} & \frac{\partial S^{n_x+n_z+1}}{\partial z_{i0}^1} & \cdots & \frac{\partial S^{n_x+n_z+1}}{\partial z_{i0}^{n_z}}
\end{bmatrix}$$

$$= \begin{bmatrix}
\dot{W}_{i0} & \cdots & 0 & 0 & \cdots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
0 & \cdots & \dot{W}_{i0} & 0 & \cdots & 0\\
0 & \cdots & 0 & \dot{W}_{i0} & \cdots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
0 & \cdots & 0 & 0 & \cdots & \dot{W}_{i0}
\end{bmatrix},$$

$$\frac{\partial S}{\partial q_i} = \begin{bmatrix} \frac{\partial S^1}{\partial v_i^1} & \cdots & \frac{\partial S^1}{\partial v_i^{n_u}} & \frac{\partial S^1}{\partial a^1} & \cdots & \frac{\partial S^1}{\partial a^{n_a}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial S^{n_x+1}}{\partial v_i^1} & \cdots & \frac{\partial S^{n_x+1}}{\partial v_i^{n_u}} & \frac{\partial S^{n_x+1}}{\partial a^1} & \cdots & \frac{\partial S^{n_x+1}}{\partial a^{n_a}} \\ \frac{\partial S^{n_x+2}}{\partial v_i^1} & \cdots & \frac{\partial S^{n_x+2}}{\partial v_i^{n_u}} & \frac{\partial S^{n_x+2}}{\partial a^1} & \cdots & \frac{\partial S^{n_x+2}}{\partial a^{n_a}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial S^{n_x+n_z+1}}{\partial v_i^1} & \cdots & \frac{\partial S^{n_x+n_z+1}}{\partial v_i^{n_u}} & \frac{\partial S^{n_x+n_z+1}}{\partial a^1} & \cdots & \frac{\partial S^{n_x+n_z+1}}{\partial a^{n_z}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial f^1}{\partial v_i^1} & \cdots & \frac{\partial f^1}{\partial v_i^{n_u}} & \frac{\partial f^1}{\partial a^1} & \cdots & \frac{\partial f^1}{\partial a^{n_a}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f^{n_x+1}}{\partial v_i^1} & \cdots & \frac{\partial f^{n_x+1}}{\partial v_i^{n_u}} & \frac{\partial f^{n_x+1}}{\partial a^1} & \cdots & \frac{\partial f^{n_x+1}}{\partial a^{n_a}} \\ \frac{\partial \hat{g}^1}{\partial v_i^1} & \cdots & \frac{\partial \hat{g}^1}{\partial v_i^{n_u}} & \frac{\partial \hat{g}^1}{\partial a^1} & \cdots & \frac{\partial \hat{g}^1}{\partial a^{n_a}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \hat{g}^{n_z}}{\partial v_i^1} & \cdots & \frac{\partial \hat{g}^{n_z}}{\partial v_i^{n_u}} & \frac{\partial \hat{g}^{n_z}}{\partial a^1} & \cdots & \frac{\partial \hat{g}^{n_z}}{\partial a^{n_a}} \end{bmatrix}$$

$$\frac{\partial S}{\partial \Gamma_{ik}} = \begin{bmatrix} \frac{\partial S^1}{\partial x_{ik}^1} & \cdots & \frac{\partial S^1}{\partial x_{ik}^{n_x+1}} & \frac{\partial S^1}{\partial z_{ik}^1} & \cdots & \frac{\partial S^1}{\partial z_{ik}^{n_z}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial S^{n_x+1}}{\partial x_{ik}^1} & \cdots & \frac{\partial S^{n_x+1}}{\partial x_{ik}^{n_x+1}} & \frac{\partial S^{n_x+1}}{\partial z_{ik}^1} & \cdots & \frac{\partial S^{n_x+1}}{\partial z_{ik}^{n_z}} \\ \frac{\partial S^{n_x+2}}{\partial x_{ik}^1} & \cdots & \frac{\partial S^{n_x+2}}{\partial x_{ik}^{n_x+1}} & \frac{\partial S^{n_x+2}}{\partial z_{ik}^1} & \cdots & \frac{\partial S^{n_x+2}}{\partial z_{ik}^{n_z}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial S^{n_x+n_z+1}}{\partial x_{ik}^1} & \cdots & \frac{\partial S^{n_x+n_z+1}}{\partial x_{ik}^{n_x+1}} & \frac{\partial S^{n_x+n_z+1}}{\partial z_{ik}^1} & \cdots & \frac{\partial S^{n_x+n_z+1}}{\partial z_{ik}^{n_z}} \end{bmatrix}$$

$$= \begin{bmatrix} \dot{W}_{ik} - \frac{\partial f^1}{\partial x_{ik}^1} & -\frac{\partial f^1}{\partial x_{ik}^2} & \cdots & -\frac{\partial f^1}{\partial x_{ik}^{n_x}} & -\frac{\partial f^1}{\partial x_{ik}^{n_x+1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{\partial f^{n_x+1}}{\partial x_{ik}^1} & -\frac{\partial f^{n_x+1}}{\partial x_{ik}^2} & \cdots & -\frac{\partial f^{n_x+1}}{\partial x_{ik}^{n_x}} & \dot{W}_{ik} - \frac{\partial f^{n_x+1}}{\partial x_{ik}^{n_x+1}} \\ -\frac{\partial \hat{g}^1}{\partial x_{ik}^1} & -\frac{\partial \hat{g}^1}{\partial x_{ik}^2} & \cdots & -\frac{\partial \hat{g}^1}{\partial x_{ik}^{n_x}} & -\frac{\partial \hat{g}^1}{\partial x_{ik}^{n_x+1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{\partial \hat{g}^{n_z}}{\partial x_{ik}^1} & -\frac{\partial \hat{g}^{n_z}}{\partial x_{ik}^2} & \cdots & -\frac{\partial \hat{g}^{n_z}}{\partial x_{ik}^{n_x}} & -\frac{\partial \hat{g}^{n_z}}{\partial x_{ik}^{n_x+1}} \end{bmatrix}$$

$$\begin{bmatrix} -\frac{\partial f^1}{\partial z_{ik}^1} & -\frac{\partial f^1}{\partial z_{ik}^2} & \cdots & -\frac{\partial f^1}{\partial z_{ik}^{n_z-1}} & -\frac{\partial f^1}{\partial z_{ik}^{n_z}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{\partial f^{n_x+1}}{\partial z_{ik}^1} & -\frac{\partial f^{n_x+1}}{\partial z_{ik}^2} & \cdots & -\frac{\partial f^{n_x+1}}{\partial z_{ik}^{n_z-1}} & -\frac{\partial f^{n_x+1}}{\partial z_{ik}^{n_z}} \\ \dot{W}_{ik} - \frac{\partial \hat{g}^1}{\partial z_{ik}^1} & -\frac{\partial \hat{g}^1}{\partial x_{ik}^2} & \cdots & -\frac{\partial \hat{g}^1}{\partial n_{ik}^{n_z-1}} & -\frac{\partial \hat{g}^1}{\partial z_{ik}^{n_z}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{\partial \hat{g}^{n_z}}{\partial z_{ik}^1} & -\frac{\partial \hat{g}^{n_z}}{\partial z_{ik}^2} & \cdots & -\frac{\partial \hat{g}^{n_z}}{\partial z_{ik}^{n_z-1}} & \dot{W}_{ik} - \frac{\partial \hat{g}^{n_z}}{\partial x_{ik}^{n_z}} \end{bmatrix},$$

where $f^{n_x+1} = \dot{x}^{\text{Lag}} = L(x(t), z(t), u(t), a, t)$, $\frac{\partial f^{(.)}}{\partial v_i^{(.)}} = \frac{\partial f^{(.)}}{\partial u^{(.)}}|_{u^{(.)}=v_i^{(.)}}$, $\frac{\partial f^{(.)}}{\partial x_{ik}^{(.)}} = \frac{\partial f^{(.)}}{\partial x^{(.)}}|_{x^{(.)}=x_{ik}^{(.)}}$, $\frac{\partial f^{(.)}}{\partial z_{ik}^{(.)}} = \frac{\partial f^{(.)}}{\partial z^{(.)}}|_{z^{(.)}=z_{ik}^{(.)}}$, $\frac{\partial \hat{g}^{(.)}}{\partial x_{ik}^{(.)}} = \frac{\partial \hat{g}^{(.)}}{\partial x^{(.)}}|_{x^{(.)}=x_{ik}^{(.)}}$ and $\frac{\partial \hat{g}^{(.)}}{\partial z_{ik}^{(.)}} = \frac{\partial \hat{g}^{(.)}}{\partial z^{(.)}}|_{z^{(.)}=z_{ik}^{(.)}}$, in addition the vectors $x = [x^1 \ \cdots \ x^{n_x} \ x^{n_x+1}]^T$, where $x^{n_x+1} = x^{\text{Lag}}$, $z = [z^1 \ \ldots \ z^{n_z}]^T$, $u = [u^1 \ \cdots \ u^{n_u}]^T$ and $a = [a^1 \ \cdots \ a^{n_a}]^T$. Note that the Jacobian matrices $\frac{\partial S}{\partial \Gamma_{i0}}$, $\frac{\partial S}{\partial q_i}$ and $\frac{\partial S}{\partial \Gamma_{ik}}$ have the

dimensions $(M \times (n_x + n_z + 1) \times (n_x + n_z + 1))$, $(M \times (n_x + n_z + 1)) \times (n_u + n_a))$ and $(M \times (n_x + n_z + 1) \times M \times (n_x + n_z + 1))$, respectively. Therefore the sensitivity matrices $\frac{\partial \Gamma_{ik}}{\partial \Gamma_{i0}}$ and $\frac{\partial \Gamma_{ik}}{\partial q}$ have the dimensions $(M \times (n_x + n_z + 1) \times (n_x + n_z + 1))$ and $(M \times (n_x + n_z + 1) \times (n_u + n_a))$, respectively, such that

$$\frac{\partial \Gamma_{ik}}{\partial \Gamma_{i0}} = \begin{bmatrix} \frac{\partial x_{ik}^1}{\partial x_{i0}^1} & \cdots & \frac{\partial x_{ik}^1}{\partial x_{i0}^{n_x+1}} & \frac{\partial x_{ik}^1}{\partial z_{i0}^1} & \cdots & \frac{\partial x_{ik}^1}{\partial z_{i0}^{n_z}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_{ik}^{n_x+1}}{\partial x_{i0}^1} & \cdots & \frac{\partial x_{ik}^{n_x+1}}{\partial x_{i0}^{n_x+1}} & \frac{\partial x_{ik}^{n_x+1}}{\partial z_{i0}^1} & \cdots & \frac{\partial x_{ik}^{n_x+1}}{\partial z_{i0}^{n_z}} \\ \frac{\partial z_{ik}^1}{\partial x_{i0}^1} & \cdots & \frac{\partial z_{ik}^1}{\partial x_{i0}^{n_x+1}} & \frac{\partial z_{ik}^1}{\partial z_{i0}^1} & \cdots & \frac{\partial z_{ik}^1}{\partial z_{i0}^{n_z}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_{ik}^{n_z}}{\partial x_{i0}^1} & \cdots & \frac{\partial z_{ik}^{n_z}}{\partial x_{i0}^{n_x+1}} & \frac{\partial z_{ik}^{n_z}}{\partial z_{i0}^1} & \cdots & \frac{\partial z_{ik}^{n_z}}{\partial z_{i0}^{n_z}} \end{bmatrix} \quad \text{and} \tag{5.9}$$

$$\frac{\partial S}{\partial q_i} = \begin{bmatrix} \frac{\partial x_{ik}^1}{\partial v_i^1} & \cdots & \frac{\partial x_{ik}^1}{\partial v_i^{n_u}} & \frac{\partial x_{ik}^1}{\partial a^1} & \cdots & \frac{\partial x_{ik}^1}{\partial a^{n_a}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_{ik}^{n_x+1}}{\partial v_i^1} & \cdots & \frac{\partial x_{ik}^{n_x+1}}{\partial v_i^{n_u}} & \frac{\partial x_{ik}^{n_x+1}}{\partial a^1} & \cdots & \frac{\partial x_{ik}^{n_x+1}}{\partial a^{n_a}} \\ \frac{\partial z_{ik}^1}{\partial v_i^1} & \cdots & \frac{\partial z_{ik}^1}{\partial v_i^{n_u}} & \frac{\partial z_{ik}^1}{\partial a^1} & \cdots & \frac{\partial z_{ik}^1}{\partial a^{n_a}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_{ik}^{n_z}}{\partial v_i^1} & \cdots & \frac{\partial z_{ik}^{n_z}}{\partial v_i^{n_u}} & \frac{\partial z_{ik}^{n_z}}{\partial a^1} & \cdots & \frac{\partial z_{ik}^{n_z}}{\partial a^{n_a}} \end{bmatrix}, \tag{5.10}$$

with $k = 1, ..., M$. Each entry of the last Jacobian matrices has a dimension $M \times 1$ which means that the last row of each entry represents the sensitivity of the terminal point of each element with respect to the initial point of the element.
From (5.1a) the gradient vector $\frac{\partial A}{\partial w}$ will be

$$\frac{\partial A}{\partial w} = \begin{bmatrix} 0 & \cdots & 0 & 0 & \frac{\partial E}{\partial h_N^x} & 1 & \frac{\partial E}{\partial h_N^z} & \frac{\partial E}{\partial a} \end{bmatrix}. \tag{5.11}$$

The sensitivity matrix $\frac{\partial B}{\partial w} =$

$$\begin{bmatrix}
0 & I_{n_x+1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\frac{\partial B_x}{\partial v_0} & \frac{\partial B_x}{\partial x_{00}} & \frac{\partial B_x}{\partial z_{00}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & (\frac{\partial B_x}{\partial a})_0 \\
0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & \cdots & 0 & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & \cdots & 0 & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I_{n_x+1} & 0 & 0 & 0 & 0 & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial B_x}{\partial v_{N-1}} & \frac{\partial B_x}{\partial x_{(N-1)0}} & \frac{\partial B_x}{\partial z_{(N-1)0}} & 0 & I_{n_x+1} & 0 & (\frac{\partial B_x}{\partial a})_{N-1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial B_r}{\partial x_N} & \frac{\partial B_r}{\partial z_N} & (\frac{\partial B_r}{\partial a})_N \\
0 & 0 & I_{n_z} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\frac{\partial B_z}{\partial v_0} & \frac{\partial B_z}{\partial x_{00}} & \frac{\partial B_z}{\partial z_{00}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & (\frac{\partial B_z}{\partial a})_0 \\
0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & \cdots & 0 & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 & \cdots & 0 & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_{n_z} & 0 & 0 & 0 & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial B_z}{\partial v_{N-1}} & \frac{\partial B_z}{\partial x_{(N-1)0}} & \frac{\partial B_z}{\partial z_{(N-1)0}} & 0 & 0 & I_{n_z} & (\frac{\partial B_z}{\partial a})_{N-1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial B_z}{\partial v_N} & \frac{\partial B_z}{\partial x_N} & \frac{\partial B_z}{\partial z_N} & (\frac{\partial B_z}{\partial a})_N
\end{bmatrix}. \tag{5.12}$$

The sensitivity matrix $\frac{\partial C}{\partial w}$ can be found by direct differentiation with respect to w,

$$\frac{\partial C}{\partial w} = \begin{bmatrix}
\frac{\partial s}{\partial v_0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{\partial s}{\partial h_0^x} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{\partial s}{\partial h_0^z} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\partial s}{\partial a}\right)_0 \\
0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & \vdots \\
0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & \left(\frac{\partial s}{\partial a}\right)_{N-1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \frac{\partial s}{\partial v_N} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial s}{\partial h_N^x} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial s}{\partial h_N^z} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\partial s}{\partial a}\right)_N \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial r_i}{\partial h_N^x} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial r_i}{\partial h_N^z} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial r_i}{\partial a}
\end{bmatrix}, \tag{5.13}$$

5.3 Solution Algorithm

Multiple shooting method depends mainly on the SQP iteration. Inside each SQP iterate gradient values of the objective function and Jacobian of the constraints as well as the approximated Hessian need to be computed. The main steps of the proposed approach

are shown in Fig. 5.1. In the algorithm implantation, piecewise constant parameters are used for the controls and we use three collocation points to solve the model equations and compute the sensitivities. Generally the number of the collocation points can be determined by the user. When a higher accuracy is needed, more collocation points should be used, but this will lead to more computation time. In addition, a higher order Lagrangian polynomial can cause oscillating state behaviors inside an element. Therefore, the three-point-collocation is unusually used for the discretization. We use the roots of Legendre polynomials as the collocation points in each element [77]. Then the locations of the three-point-collocation in an element will be $t_{i0} = t_i$, $t_{i1} = \alpha_1(t_{i+1} - t_i) + t_i$, $t_{i2} = \alpha_2(t_{i+1} - t_i) + t_i$ and $t_{i3} = \alpha_3(t_{i+1} - t_i) + t_i$ where α_1, α_2 and α_3 are the roots of Legendre polynomials [141]. Since we define the last point of an element as the initial point of the next element i.e. $t_{iM} = t_{(i+1)0}$, the locations of the collocation points will then be

$$\begin{aligned} t_{i0} &= t_i, \\ t_{i1} &= \hat{\alpha}_1(t_{i+1} - t_i) + t_i, \\ t_{i2} &= \hat{\alpha}_2(t_{i+1} - t_i) + t_i, \\ t_{i3} &= t_{i+1}. \end{aligned} \tag{5.14}$$

where $\hat{\alpha}_1 = \alpha_1/\alpha_3$ and $\hat{\alpha}_2 = \alpha_2/\alpha_3$. On the other hand, the typical form of the path inequality constraints are normally minimum and maximum bounds on states and controls, thus the sensitivity matrix $\frac{\partial C}{\partial w}$ will be

$$\frac{\partial C}{\partial w} = \begin{bmatrix} I_{M(n_u+n_x+n_z)} & 0 \\ 0 & I_{n_a} \\ -I_{M(n_u+n_x+n_z)} & 0 \\ 0 & -I_{n_a} \\ 0 \quad \frac{\partial r_i}{\partial h_N^x} \quad 0 & 0 \\ 0 \quad 0 \quad \frac{\partial r_i}{\partial h_N^z} & 0 \\ 0 \quad 0 \quad 0 & \frac{\partial r_i}{\partial a} \end{bmatrix}. \tag{5.15}$$

In this case, we do not need to compute the Jacobian since we can define the bounds at the problem initialization. However, we use Algorithm 6[1] to implement the solution of the NMPC problem even we do not have a path inequality constraints. In the algorithm implementation, we first transform the algebraic states into differential states.

This approach has several advantages which can be summarized as follows:

1. The optimal control problem will be transformed into NLP problem using multiple shooting, and an efficient NLP solver, IPOPT, is used for the solution. In addition, the collocation on finite elements method is used for the integration of the model equations and the computation of the gradient required.

[1] In the realized framework, the the numerical algorithm group (NAG) library Mark 8 [88] is used to solve the linear and nonlinear systems and the interior point optimizer (IPOPT) [180] is used to solve the NLP problem.

Algorithm 6 Optimization using multiple shooting and collocation

Input: time horizon $[t_0, t_f]$, number of intervals N, number of differential states n_x, number of algebraic states n_z, number of controls n_u, number of parameters n_a, lower and upper bounds of all variables, differential and algebraic state equation, path constraints, initial guess, optimizer tolerance and Hessian approximation.

Output: optimal state trajectories, optimal control trajectories, optimized performance index and CPU time.

Step 1: NLP problem initialization.
- Time horizon.
- Intervals $[t_i \;\; t_{i+1}]$.
- Upper and lower bounds for states, controls and constraints.
- Given initial state.
- Initial guess w_0.

Step 2: Define the continuity constraint $B(w)$ Eq. (3.9b).

Step 3: Define the path constraint $C(w)$ Eq. (3.9c).

Step 4: Initialize the three collocation points for each interval Eq. (5.14).

Step 5: Compute the constraint equations and their sensitivities.
- Solve the equation (5.6) using the Newton-Raphson method.
- Solve equation (5.8) using LU factorization.

Step 6: Compute objective function (3.9a) and its sensitivity.

Step 7: Solve NLP iteration.
- If KKT is not satisfied go to *Step* 4.

Stop.

Due this combination, the proposed approach has a higher computation efficiency since the collocation on finite elements method requires a smaller amount of computation expense compared with the typical DAE solvers. This small computation expense is justified by the fact that the collocation on finite elements method approximates each state variable by an interpolating function in each time interval with a number of collocation points. This process results in a system of nonlinear equation and therefore an efficient root-finding technique, e.g. Newton-Raphson, will be used to compute the collocation points. Furthermore, the collocation on finite elements method needs only to solve a system of linear equations, which is obtained by taking the first order Taylor expansion of the system of nonlinear equations, to compute the necessary sensitivities for the NLP problem, and thus a fast and well-suited LU factorization method can be used for this solution. On the other hand, typical DAE solvers use the integration of multisteps to compute the state variables in each time interval. This integration normally costs additional time if the number of steps is increased. In addition, the variable sensitivities will be also computed using a multisteps method if the typical DAE solver is used. This method uses the derivatives of the integrated steps and the chain-rule to these derivatives, which also increase the time expense.

At the same time, more accuracy of the system states can be obtained using collocation on finite elements since the truncation error of the state variable will be $O(\xi^{2M})$ if the roots of Lagender polynomials are used to locate the positions of the collocation points in each element, where ξ is the time length between two successive

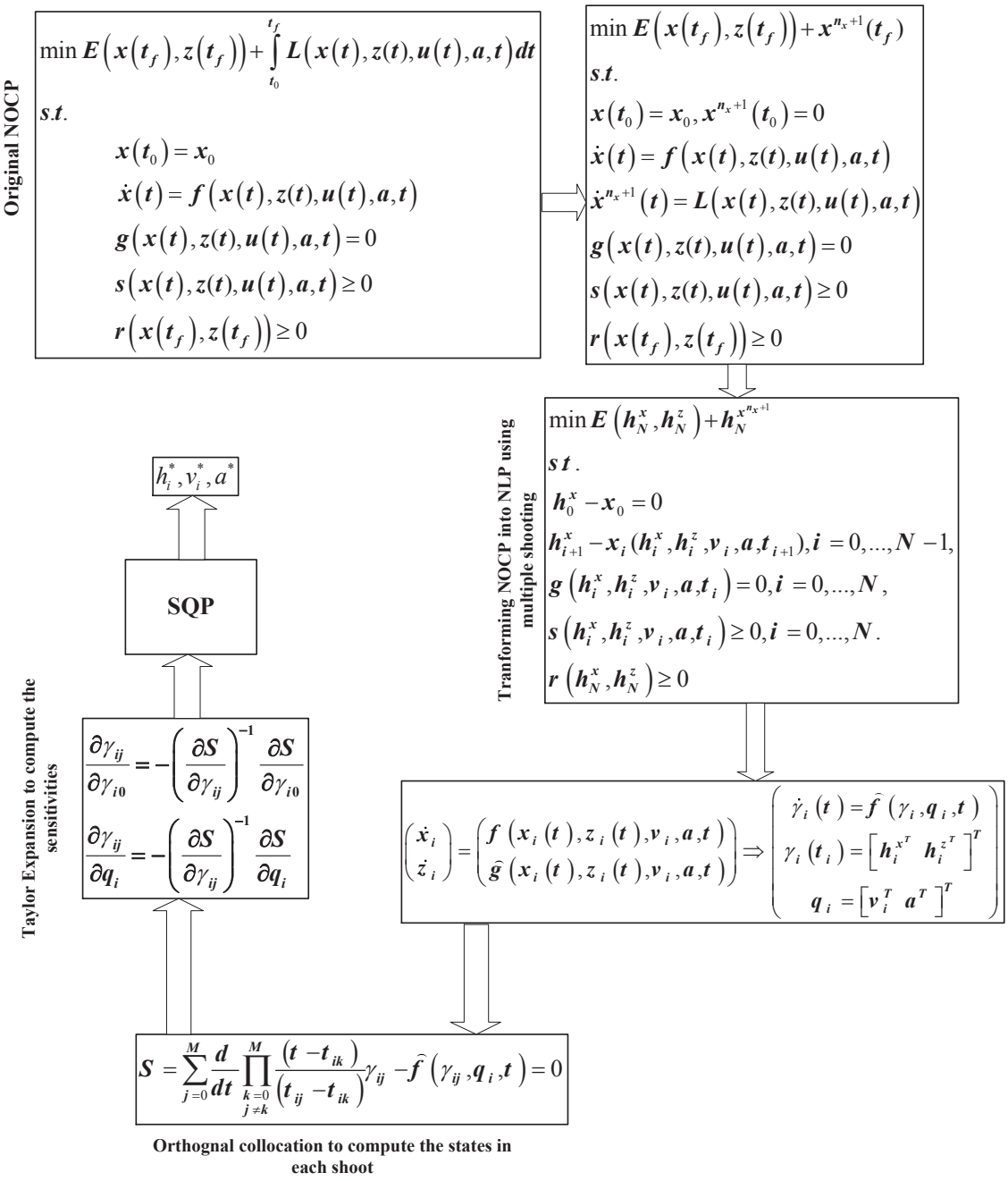

Figure 5.1: Structure of proposed approach.

collocation points and M is the number of the collocation points. In addition, more truncation error will be yielded in the state variables if a typical DAE solver is used. This error is characterized by a Taylor series and given by $O(\xi^{\rho})$ where ξ is the step size and ρ is the order of DAE solver.

This accurate solutions of the state variables also provide accurate sensitives which lead to more accuracy in the solution of the optimal control problem.

2. In comparison to the collocation method, by using multiple shooting the size of the resulted NLP problem is much smaller. Since the size of the NLP problem in multiple shooting is only composed of the parameterized states and controls at the time nodes, while if the collocation method is used the state and control variables at the collocation points in each finite elements are also considered as the variables of the NLP problem. In addition, since the sensitivity of the problem in each time interval is independent using multiple shooting the sensitivity matrix is more sparse if the collocation method is used.

3. Since in multiple shooting the integrations on each multiple shooting intervals are completely independent, this makes a parallel computation possible. Therefore, the increase of the number of time intervals will not lead to more computation expense if a parallel computation is implemented.

6 Stability Analysis

In this Chapter we present the principle of the receding horizon control (RHC) and discuss important schemes that have been used to analyze the stability of NMPC systems. In addition, we propose a new approach to ensure the stability of a wide class of NMPC systems.

6.1 Receding Horizon Control

To introduce RHC we need first to review the principle of an optimal feedback control. We assume that the optimal control problem (2.2) at a given initial condition, $x(t_0) = x_0$, has an optimal control $u^*(x_0, t)$ with corresponding differential and algebraic states $x^*(x_0, t)$ and $z^*(x_0, t)$, respectively, where $t \in [t_0, t_f]$, as well as an optimal parameter vector a^*. In addition, at a certain time point, e.g., $t_1 \in [t_0, t_f]$ we have a corresponding state $x_1 = x^*(x_0, t_1)$, so that the optimal control with the new updated initial condition x_1 instead of x_0 is $u^*(x_1, t)$ and an optimal feedback control function is $u^f = u^*(x_0, t_0)$. Using this feedback control function, a closed-loop DAE system can be redefined

$$\begin{aligned} &\dot{x}(t) = f(x^p(t), u^f(t), z^p(t), a, t), && t \in [t_0, t_f], \\ &g(x^p(t), u^f(t), z^p(t), a, t) = 0, && t \in [t_0, t_f]. \end{aligned}$$

where $u^f(t)$ is an optimal feedback control and $x^p(t)$ and $z^p(t)$ are closed-loop state vectors. Within the receding horizon the state values $(x_0, x_1, ...)$ will be measured continuously at the instants $t_0, t_1, ...$ where $t_1 \in [t_0, t_f]$, $t_2 \in [t_1, t_f + \Delta t_i]$... and $\Delta t = t_{i+1} - t_i$, then the optimal control prblem will be solved to compute optimal controls $u^f(x_0, t_0)$, $u^f(x_1, t_1), ...$.

Nonlinear optimal control problems with finite time horizons can be solved by the proposed approach presented in Chapter 5. When an infinite time horizon in the optimal control problem is considered, i.e. $t_f \to \infty$, then the optimality conditions can hold for the trajectories $x^*_\infty(x_0, .), z^*_\infty(x_0, .)$ and $u^*_\infty(x_0, .)$ and the feedback control law can be defined accordingly as $u^f_\infty(x_0) = u^*_\infty(x_0, t_0)$.

To ensure a nominal stability of the corresponding closed-loop system for steady-state tracking problems we define an objective functional

$$V_\infty(x_0) = \int_0^\infty L\left(x(t), z(t), u(t), a, t\right) dt \tag{6.1}$$

that satisfies $L(x, z, u, a) > 0, \quad \forall \quad (x, z, u, a) \neq (x_S, z_S, u_S, a)$ and $L(x_S, z_S, u_S, a) = 0$ at the steady-state $(x_S,\ z_S,\ u_S)$. As the optimal cost functional (6.1) remains finite, it serves as a Lyapunov function for the closed-loop system. That means

$$\begin{aligned} &V_\infty(x_S) = 0 \text{ and} \\ &V_\infty(x_0) > 0 \end{aligned} \tag{6.2}$$

where $x_0 \neq x_S$, so that

$$L(.) < 0, \qquad \forall x^p(t) \neq x_S \tag{6.3}$$

which means that the steady-state value of the closed-loop trajectory $x(t)$ will be accumulated to be equal the finite value x_S [164].

Generally, it is difficult to handel this problem for nonlinear and constrained systems. Therefore, a so called *moving horizon* or *receding horizon* control principle is usually used to treat the infinite horizon problems. Fig. 6.1 shows how the receding horizon control performs, where a finite prediction horizon (T_P), say $[t_0, t_f]$, is chosen to carry out the optimization task and a constant control horizon (T_C), say $[t_0, t_c]$, will be chosen such that the length of the control horizon is smaller than or equal to the length of the prediction horizon. However, if the control horizon is sufficiently large, the computed optimal state trajectory vectors $x^*(t)$ and $z^*(t)$ and the control trajectory $u(t)$ are expected to be similar to the corresponding values in the infinite horizon case.

We assume that the system states are measurable, then the basic concept of receding horizon can be summarized as: at the current instant, say t_i, the optimal control is obtained on a fixed finite horizon, e.g. $[t_i, t_i+N\Delta t]$ where $N\Delta t$ is the length of the prediction horizon. This optimal control will be obtained by solving a predefined optimal control problem based on the current measurements of the states, x_i, in the optimal control problem. The optimal control vector will be applied to the system over a control horizon, e.g. $[t_i, t_i + P\Delta t]$ where $P < N$ and $P\Delta t$ is the length of the control horizon . The states will be measured again at the end of the control horizon. These state measurements will be considered again as initial conditions of the optimal control problem and the procedure is then repeated at the next prediction horizon, say $[t_i + P\Delta t, t_i + P\Delta t + N\Delta t]$. The term "receding horizon" is introduced, since the horizon recedes as time proceeds as shown in Fig 6.1.

To implement the RHC using the proposed optimization approach presented in Chapter 5 we apply Algorithm 7.

Algorithm 7 Receding horizon control

Input: optimization (prediction) horizon ($T_P = t_f - t_0$), control horizon ($T_C = t_c - t_0$), number of subintervals N, number of differential states n_x, number of algebraic states n_z, number of controls n_u, number of parameters n_a, lower and upper bounds of all variables, model equations, initial condition vector x_0, path constraints, initial guess, optimizer tolerance and hessian approximation.

Output: optimal state trajectories, optimal control trajectories, optimized performance index and CPU time for every optimal control problem solution.

Step 0: Set $k = 0$.

Step 1: Solve the optimal control problem by applying Algorithm 6.
The result of this step is an optimal control vector $u^*(t),\ t \in [t_0,\ t_0 + T_P]$.

Step 2: Apply the optimal control vector $(u^*(t),\ t \in [t_0, t_0 + T_C)$ to the system.

Step 3: Measure the system states x_{k+1} at $t = t_0 + T_C$.

Step 4: Set $x_0 = x_{k+1}$

Step 5: Set $k = k + 1,\ t_0 = t_0 + T_C$ and $t_f = t_0 + T_P$ then go to Step 1.

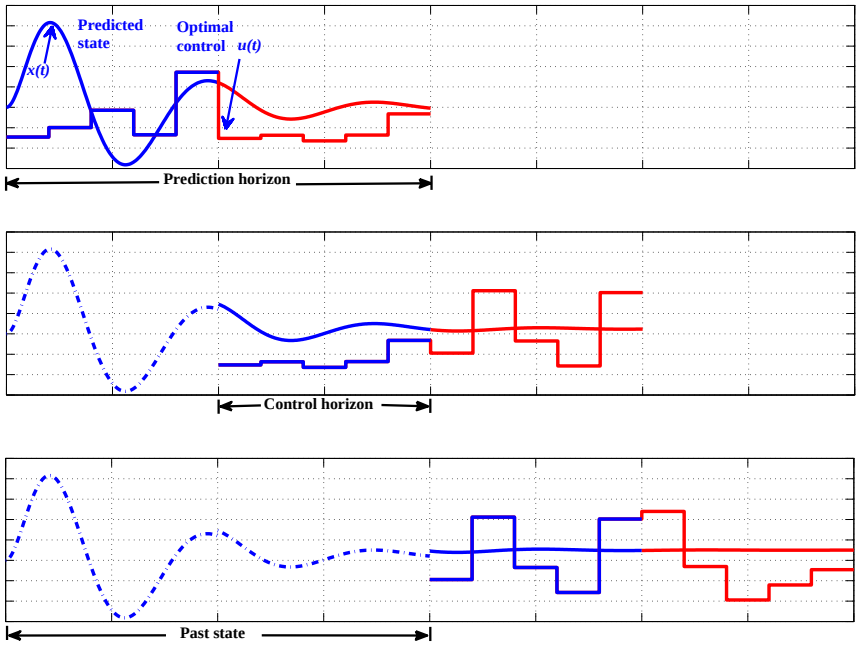

Figure 6.1: Principle of receding horizon control.

6.2 Lyapunov Stability Theory

Lyapunov stability theory is a necessary tool to analyze the stability of NMPC systems with constraints. In this section we review previous studies which will be used in the sequel to analyze the stability properties of NMPC systems. First we consider the following nonlinear differential equation

$$\dot{y} = f(y,t) \qquad y(t_0) = y_0 \qquad y \in \mathbb{R}^{n_y} \tag{6.4}$$

With a direct method of Lyapunov we can determine the stability of the system without explicitly integrating the differential equation (6.4). The idea of the method is to analyze a 'measure of energy' in a system, then a rate of change of the energy of the system can be studied to ensure the stability. To make this precise, we need the following definitions and notations.

Definition 6.1 (Scalar and Euclidean Norms):
For a given state vector $y \in \mathbb{R}^{n_y}$, $|y|$ and $\|y\|^2$ represent its scalar norm and Euclidean norm, respectively, in addition, $\|y\|_Q^2 = y^T Q y$ is a weighted Euclidean norm of $y \in \mathbb{R}^{n_y}$ where Q is a positive definite matrix ($Q > 0$) [164, 176].

Definition 6.2 (Equilibrium Point):
A point $\eta_y \in \mathbb{R}^{n_y}$ is an equilibrium point of differential Eq. (6.4) if $f(\eta_y, t) = 0$ and η_y is the point at which the plant operates at a steady-state [96, 164].

Definition 6.3 (Local Lyapunov Stability):
*The equilibrium point η_y is **locally Lyapunov stable** if all solutions which start near η_y remain near η_y for all time.*

Thus, for each $\varepsilon > 0$ and each $t_0 \in \mathbb{R}_+$, there exists $\delta = \delta(\varepsilon)$ such that if $\|y_0\| < \delta(\varepsilon)$ then $\|y(t)\| < \varepsilon, \quad \forall t \geq t_0$ [130, 164, 176].

Definition 6.4 (Asymptotical Stability):
*The equilibrium point η_y is **uniformly asymptotically stable** if η_y is locally stable and solutions starting near η_y tend towards η_y as $t \to \infty$.*

That means for each $t_0 \in \mathbb{R}_+$ there exists $\delta > 0$ such that if $\|y_0\| < \delta$, then $\lim_{t\to\infty} y(t) = \eta_y$ [130, 164, 176].

Definition 6.5 (Exponential Stability):
*The equilibrium point $\eta_x \in \mathbb{R}^n$ of the system is **exponentially stable** if it is asymptotically stable and if there exist constants α, β and $\delta > 0$ such that*

$$\|y(t)\| \leq \alpha \|y_0\| e^{-\beta t} \qquad \forall t \geq t_0, \forall y_0 \text{ with } \|y_0\| < \delta. \tag{6.5}$$

*The largest constant β utilized in Eq. (6.5) is called **the rate of convergence** [96, 130, 176].*

Definition 6.6 (Locally Positive Definite Function):
A continuous function $V(y,t) : \{\mathbb{R}^{n_y} \times \mathbb{R}_+ \to \mathbb{R}\}$ is a locally positive definite function (with respect to the origin) if for some $\epsilon > 0$ and some continuous, strictly increasing function $\mathfrak{a}(p) : \mathbb{R}_+ \to \mathbb{R}$,

$$V(0,t) = 0 \text{ and } V(y,t) \geq \mathfrak{a}(\|y\|) \qquad \forall y \in \mathbb{R}^{n_y}, \quad \|y\| < \epsilon, \tag{6.6}$$

where ϵ is the radius of the ball around the origin [130].

Definition 6.7 (Positive Definite Function):
A continuous function $V(y,t) : \{\mathbb{R}^{n_y} \times \mathbb{R}_+ \to \mathbb{R}\}$ is a positive definite function if it is a locally positive definite function and, furthermore, $\lim_{p\to\infty} \mathfrak{a}(p) = \infty$ [96].

Definition 6.8 (Decrescent Function):
A continuous function $V(y,t) : \{\mathbb{R}^{n_y} \times \mathbb{R}_+ \to \mathbb{R}\}$ is decrescent for $\epsilon > 0$ and some continuous, strictly increasing function $\mathfrak{b} : \{\mathbb{R}_+ \to \mathbb{R}\}$ if [130]

$$V(y,t) \geq \mathfrak{b}(||y||) \qquad \forall y \in \mathbb{R}^{n_y}, \qquad \|y\| < \epsilon, \quad \forall t \geq t_0. \tag{6.7}$$

Using these definitions, the basic theorem of Lyapunov can be derived to determine stability for a system by studying an appropriate energy function.

Theorem 6.1 (Basic Theorem of Lyapunov [96]):
When $V(y,t)$ is a non-negative function with $\dot{V}(y,t)$ along the trajectories of the system where

$$\dot{V}(y,t) = \dot{V}(y,t)|_{\dot{y}=f(y,t)} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial y} f$$

- *If $V(y,t)$ is positive definite and $\dot{V}(y,t) \leq 0$ is locally in y and for all t, then the origin of the system is locally stable.*
- *If $V(y,t)$ is positive definite and decrescent, and $\dot{V}(y,t) \leq 0$ locally in y and for all t, then the origin of the system is uniformly locally stable.*
- *If $V(y,t)$ is positive definite and $-\dot{V}(y,t)$ is locally positive definite, then the origin of the system is uniformly locally asymptotically stable.*
- *If $V(y,t)$ is positive definite and $-\dot{V}(y,t)$ is positive definite, then the origin of the system is globally uniformly asymptotically stable.*

Proof. The proof of Theorem 6.1 can be found in [176].

The search for a Lyapunov function that establishes the stability of an equilibrium point could be arduous, thus Theorem 6.1 gives only a sufficient condition. To apply the Lyapunov stability theory to NMPC of a constrained system, we define an optimal control problem with a Lyapunov performance index. A typical example of a suitable Lyapunov objective functional is a quadratic objective functional

$$J = \int_{t_k}^{t_k+T_P} (\|x\|^2_{Q_x} + \|z\|^2_{Q_z} + \|u\|^2_{R_u} + \|a\|^2_{R_a})dt. \tag{6.8}$$

This optimal control problem minimizes the objective functional J, Eq. (6.8), subject to the system Eqs. (2.2b) to (2.2k). We will analyze the stability of this problem in Section 6.3.

6.3 Schemes to Ensure the Stability of the NMPC

A stability analysis of NMPC systems has long been a challenge due to the nonlinear problem formulation. There are many approaches to ensure the nominal stability for the steady-state. Some of the most straightforward ways to achieve stability can be summarized as follows:

- Using stability constraints (besides the control and state constraints) by introducing, e.g.,
  1. terminal equality constraints [95],
  2. terminal cost functions into the open-loop optimal control problem [143],
  3. terminal constraint set [127] or
  4. terminal cost or constraint set [47, 161].
- Introducing a Lyapunov function for an infinite prediction horizon.
- Approximating the nonlinear system by a linear system at the stationary point [40, 41].
- Introducing contractive constraints with contractive parameters to the optimal control problem.

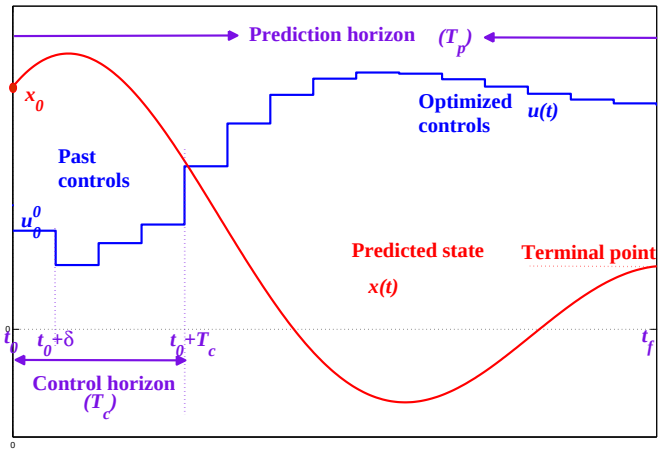

Figure 6.2: Principle of terminal equality constraints.

Terminal Equality Constraints

It can be shown that a stability for a NMPC system can not be guaranteed, if a finite prediction horizon is employed [16, 22, 37]. But if a terminal constraint is introduced, an asymptotic stability can be achieved [12, 81, 125, 187]. In this method, an open-loop nonlinear optimal control problem (2.2) will be defined such that the Mayer term (terminal cost) $E(.)$ and the terminal constraints (2.2f) satisfy $E(.) \equiv 0$ and the vector function$\{r_e : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_a} \rightarrow \mathbb{R}^{n_x}\}$, respectively.

The first algorithm in the literature was presented by Keerthi an Gilbert [95] and Mayne and Michalska [123]. Fig. 6.2 shows schematically how the terminal equality constraint enforces a state to satisfy a terminal condition in the optimal control problem. If the prediction horizon is short the terminal constraints will force an excessive control effort and the computational load will be increased as the prediction horizon is increased [117]. Moreover, in the case of continuous feedback control the instable system can be stabilized using the NMPC with terminal equality constraints [125].

Although using the terminal equality constraints to guarantee stability is an intuitive approach, it increases significantly the on-line computation necessary to solve the open-loop optimization problem, and often causes feasibility problems. For more details see [56–58, 116, 124].

Terminal Cost Function

In this method, the terminal cost (Mayer term) is nontrivial value and the terminal equality constraints (2.2d) do not exist. The idea of this method is that the Mayer term is defined to force the states to behave such that the terminal cost is bounded. A typical terminal cost is a quadratic terminal penalty. More details can be found [143].

Terminal Constraint Set

This method is similar to the terminal equality constraint method, but $\mathbb{R}^{n_e}$ is a subset of $\mathbb{R}^{n_x}$. In this method the terminal constraint set will steer the states to specific terminals

in a finite time. The terminal constraint set may have a stabilizing controller. Therefore, this method of NPMC is referred to as dual-control mode [127, 168].

Terminal Cost and Terminal Constraint Set

In this method, a terminal constraint (set) does exist and a terminal cost (Mayer term) is nontrivial value [47, 124]. The main advantage of this scheme is that the Mayer term should be equivalent to the objective function value of the infinite horizon J_∞. Thus

- the Lagrangian term in the case of finite horizon will be equivalent to the Lagrangian term in the case of infinite horizon ($\int_{t_0}^{t_f} L(.) \equiv \int_{t_0}^{\infty} L(.)$),
- an online optimization would be unnecessary and
- stability and robustness advantages of the infinite problems would be automatically satisfied to the finite horizon problems.

It is difficult, however, to apply this scheme to the finite optimal control problem due to nonlinearity of the states. But it is possible to choose a Mayer term such that it is exactly or approximately equivalent to the infinite horizon value around the origin. On the other hand, as stated before, choosing a terminal constraint set adds additional advantages to the robustness and stability properties of the NMPC system.

Lyapunov-Based Model Predictive Control

A common approach to stability analysis in the literature is a Lyapunov stability theory. It is indicated in, e.g., [2, 113, 117, 126, 153, 187] that an asymptotic stability can be obtained by using a Lyapunov function if an infinite prediction horizon is defined. In this section, we review the main concept of this approach. First, we consider the nonlinear system model Eq. (6.4) by decoupling the state variables from the controls, i.e. the vector $\gamma(t)$ is defined instead of $y(t)$ to address the differential and algebraic states (x and z) and vector $q(t)$ which combines the control vector $u(t)$ and the parameter vector a, such as

$$\begin{aligned} &\dot{\gamma}(t) = \hat{f}(\gamma(t), q(t), t), \qquad t \geq t_k, \\ &\gamma(t_k) = \gamma_k = [x_k^T \quad z_k^T]^T, \\ &q = [u^T \quad a^T]^T, \end{aligned} \tag{6.9}$$

where $\gamma(t) \in \mathbb{R}^{n_\gamma}$ and $q \in \mathbb{R}^{n_q}$, $n_\gamma = n_x + n_z$, $n_q = n_u + n_a$ and the index $k = 0, 1, ...,$ denotes the index of the control horizon. Now we define a finite optimal control problem

$$\min_{q(t),\gamma(t)} J = E(\gamma(t_f), t_f) + \int_{t_k}^{t_k+T_P} \left(\|\gamma\|^2_{Q_\gamma} + \|q\|^2_{R_q} \right) dt. \tag{6.10}$$

subject to the system Eq. (6.9) as well as some terminal and path constraints on the system states and controls.

By applying the control vector $q(t) = q^0_{t_k,t_{k+1}}, \quad t \in [t_k, t_{k+1})$, where $q^0_{t_k,t_{k+1}}$ is the optimal vector in the first control horizon $t_{k+1} - t_k = T_C$, we can implicitly define a sampled state feedback control law

$$q(t) = q^f(t, \gamma(t_k)) \quad t \in [t_{k+1}, t_k) \tag{6.11}$$

According, e.g., to [117] the terminal constraint set and the terminal cost function, which are introduced in the optimal control problem, must be properly chosen in order to guarantee the stability of the NMPC in the closed-loop system.

Theorem 6.2 (Lyapunov-Based Model Predictive Control [117]):
If there exist an auxiliary control law $q^f(t, \gamma(t_k))$, a terminal constraint set and a terminal penalty function $(E(.))$ such that, letting $\gamma^p(\gamma(t_k), t)$ be a solution of the closed-loop system $\dot{\gamma}(t) = \hat{f}(\gamma(t), q^f, t),\ t \geq t_k,$ from the initial time t_k and the initial state $\gamma(t_k)$, in addition the following conditions hold:

- $\mathbb{R}^{n_e} \subset \mathbb{R}^{n_\gamma}, \quad \mathbb{R}^{n_e}$ *is closed and* $0 \in \mathbb{R}^{n_e}$.
- $q^f(\gamma(t_k), t) \in \mathbb{R}^{n_q}, \quad \forall \gamma \in \mathbb{R}^{n_\gamma}$.
- $\mathbb{R}^{n_e}$ *is positively invariant for (6.10).*
- *The Mayer term* $E(.) : \mathbb{R}^{n_\gamma} \to \mathbb{R}$ *such that* $\forall \gamma \in \mathbb{R}^{n_\gamma}$

$$E(\gamma^p(\gamma(t_k)), t_{k+1}, t_k) - E(\gamma(t_k)) \leq \\ - \int_{t_k}^{t_{k+1}} \left(\|\gamma^p(\gamma(t_k), \tau, t_k)\|^2_{Q_\gamma} + \|q^f(\gamma(t_k), \tau, t_k)\|^2_{R_q} \right) d\tau \tag{6.12}$$

then the origin is an asymptotically stable equilibrium point for a closed-loop system formed by (6.9) *and* (6.11) *with maximum output.*

Proof. The proof of Theorem 6.2 can be found in [117].

It can be seen that Theorem 6.2 can be applied to a NMPC formulation with terminal equality constraint and therefore leads to an asymptotical stable closed-loop system. Chen and Allgwer [42, 43] considered nonlinear systems with infinite prediction horizon. They first approximated the nonlinear system by a linear system at the stationary point and a quadratic terminal cost is used to penalize the objective function. Here the satisfaction of Theorem 6.2 can be easily checked and the asymptotica stability can then be ensured by linear state feedback design [44, 74]. Rawlings and Muske [143] introduced a Lyapunov stable MPC. They considered infinite and quasi-infinite NMPC and assumed constant weights on control and state variables. On the other hand, Zheng [186] showed that constrained NMPC can be stabilized by using time-varying weights. Santos and Biegler [153, 154] developed a strategy to build stability bounds on model mismatch for a class of NMPC algorithms.

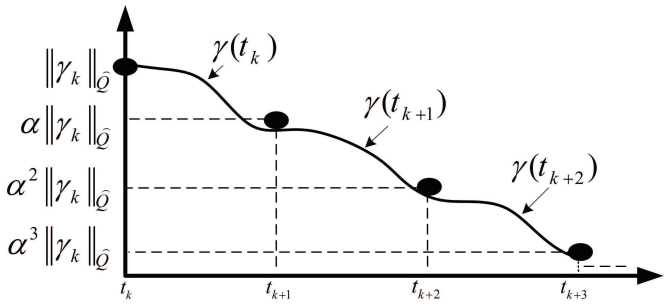

Figure 6.3: Exponential decay of the state trajectory.

Contractive Approach to Ensure the MPC Stability

Kothare and Morari [100] proposed a contractive algorithm to stabilize the NMPC system by introducing additional state constraints. This method is called contractive MPC (CNTMPC) and is also a Lyapunov-based approach. The contractive constraints render the system exponentially stable in the state feedback case and uniformly asymptotically stable in the case of output feedback [99, 100]. CNTMPC, which should be parameterized by the user, is based on the contraction property of the state feedback control system [23, 118, 119, 184]. With CNTMPC, contractive state inequality constraints will be introduced to impose system states at the end of each sample time to be contracted in norm with respect to the states at the beginning of the sample time, i.e.

$$\|\gamma_{k+1}\|^2_{\hat{Q}_\gamma} = \|\gamma_k(t_{k+1})\|^2_{\hat{Q}_\gamma} \leq \alpha\|\gamma_k\|^2_{\hat{Q}_\gamma}, \qquad \alpha \in [0,1). \tag{6.13}$$

The inequality constraint (6.13) is called a contractive constraint which is steered mainly by the contractive parameter α. This parameter is chosen such that the optimization problem feasible. The employment of these constraints guarantees a global exponential stability of the closed-loop system [100].

The behavior of the closed-loop system generated by CNTMPC by choosing a suitable α to make the optimal control problem feasible is illustrated in Fig. 6.3. This figure displays the exponential decay of the state trajectories with state contraction occurring at every prediction horizon.

6.4 A New Approach to Ensure the NMPC Stability

In this section, we propose a new approach to ensure the stability of NMPC systems by introducing auxiliary state variables and corresponding linear state equations. We enforce the system states to be contracted with respect to the auxiliary state variables by adding inequality constraints. Thus the stability properties of system states will conform to those of the auxiliary states, i.e. the system states will be stable, if the auxiliary states are stable. The eigenvalues of the linear state equations introduced will be determined such that they stabilize the auxiliary state variables and at the same time make the optimal control problem feasible. This is achieved by considering the eigenvalues as optimization variables in the optimal control problem. Therefore, the solution of the optimal control problem guarantees the feasibility, stability and optimality of the NMPC system.

First we consider an open-loop optimal control problem of the form

$$\min_{x,u} J(x(t),u(t),t) = E(x(t_f)) + \int_{t_0}^{t_f} L(x(t),u(t),t)dt, \tag{6.14a}$$

subject to

$$x(t_0) = x_0. \tag{6.14b}$$
$$\dot{x}(t) = f(x(t),u(t),t), \qquad \forall t \in [t_0,t_f], \tag{6.14c}$$
$$s(x(t),u(t),t) \geq 0, \qquad \forall t \in [t_0,t_f], \tag{6.14d}$$
$$u_{\min} \leq u(t) \leq u_{\max}, \qquad \forall t \in [t_0,t_f], \tag{6.14e}$$
$$x_{\min} \leq x(t) \leq x_{\max}, \qquad \forall t \in [t_0,t_f], \tag{6.14f}$$

where $x(t) \in \mathbb{R}^{n_x}$, and $u(t) \in \mathbb{R}^{n_u}$ are the differential algebraic state vector and the control vector, respectively, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $s : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_s}$. The following definitions and assumptions are needed for the derivation of the proposed approach.

Definition 6.9 (Feasibility of the Optimal Control Problem):
The optimal control problem (6.14) is feasible if there exist trajectory vectors $x^*(t) \in \mathbb{R}^{n_x}$ and $u^*(t) \in \mathbb{R}^{n_u}$ such that [103]

$$x^*(t_0) = x_0.$$
$$\dot{x}^*(t) = f(x^*(t),u^*(t),t), \qquad \forall t \in [t_0,t_f],$$
$$s(x^*(t),u^*(t),t) \geq 0, \qquad \forall t \in [t_0,t_f],$$
$$u_{\min} \leq u^*(t) \leq u_{\max}, \qquad \forall t \in [t_0,t_f]$$
$$x_{\min} \leq x^*(t) \leq x_{\max}, \qquad \forall t \in [t_0,t_f].$$

Definition 6.10 (Optimality of the Optimal Control Problem):
A trajectory vector $u^*(t) \in \mathbb{R}^{n_u}$ is a local optimal control with a corresponding state trajectory vector $x^*(t) \in \mathbb{R}^{n_x}$ of the optimal control problem (6.14) if [103]

- *the optimal control problem (6.14) is feasible on $u^*(t)$ and $x^*(t)$ and*
- *the trajectory vectors $x^*(t)$ and $u^*(t)$ satisfy $J(x^*(t),u^*(t),t) \leq J(x(t),u(t),t)$ for all feasible neighborhoods of x^* and u^*.*

Definition 6.11 (Stabilizability):
The system Eq. (6.18c) *is a* ***stabilizable*** *system if a bounded input vector yields a bounded output [133].*

Assumption 6.1:
The system Eq. (6.18c) *has a shifted equilibrium point at $\eta = (\eta_x, \eta_u) = (0,0)$.*

Assumption 6.2:
The bounds of the system state vector of Eq. (6.18c) *are finite, i.e. $x_{\min} \neq -\infty$ and $x_{\max} \neq \infty$.*

Assumption 6.3:
The linearization of the system dynamics (6.18c) *around the origin is stabilizable (satisfies Definition 6.11 for* $u_{\min} \leq u(t) \leq u_{\max}$*), i.e.,* $\{(\partial f/\partial x)(0,0), (\partial f/\partial u)(0,0)\}$ *is a stabilizable pair.*

Assumption 6.4:
There exists a time-varying vector $z(t) \in \mathbb{R}^{n_x}$, $t \geq t_0$, *such that the following conditions hold:*

- $z(t) \in [0, x_{\max}]$ *and* $-z(t) \in [x_{\min}, 0]$,
- $z(t)$ *is an asymptotically decreasing function,*
- $\lim_{t\to\infty} z(t) = 0$ *and*
- $-z(t) \leq x(t) \leq z(t)$.

Assumption 6.5:
The optimal control problem (6.14) is feasible according to Definition 6.9 and at the same time Assumption 6.4 holds .

Assumption 6.6:
There exists a time-varying trajectory vector $z(t) \in \mathbb{R}^{n_x}$, $t \geq t_0$, *which satisfies Assumption 6.4 and* $z(t)$ *is an exponentially decreasing function such that*

$$\dot{z}(t) = A_z z(t),\ z(t_0) = Z_0 \in \mathbb{R}^{n_x}, \qquad \forall t \in [t_0, t_f], \tag{6.15}$$

where a time-invariant matrix $A_z \in \mathbb{R}^{n_x \times n_x}$ *is a diagonal matrix,* $A_z = \mathrm{diag}(a_z)$, *where* $a_z < 0$.

Assumption 6.7:
The optimal control problem (6.14) is feasible according to Definition 6.9 and at the same time Assumption 6.6 holds.

Theorem 6.3 (Constrained Asymptotical Stability):
Suppose that Assumptions 6.1-6.4 are satisfied then the state $x(t)$ *is asymptotically stable.*

Proof. Since Assumption 6.4 is satisfied on the system dynamics (6.14c) we have a time-varying trajectory vector $z(t)$ such that

$$\lim_{t\to\infty} z(t) = 0.$$

and $z(t)$ is asymptotically stable, i.e. there exists $\delta_z \geq 0$ (from Definition 6.4) and

$$\|z_0\| < \delta_z,$$

then from the inequality $-z(t) \leq x(t) \leq z(t)$ there exists x_0 such that

$$\|x_0\| \leq \|z_0\| < \delta_z,$$

and also

$$\lim_{t\to\infty} x(t) = 0.$$

which means $x(t)$ is asymptotically stable. □

Theorem 6.4 (Constrained Exponential Stability):
Suppose that Assumptions 6.1-6.3 and 6.6 are satisfied then the state $x(t)$ is exponentially stable.

Proof. Since Assumption 6.6 is satisfied on the system dynamics (6.14c) we have a time-varying trajectory vector $z(t)$ such that

$$z(t) = e^{A_z t} Z_0,$$

i.e.there exist α_z, β_z and $\delta_z > 0$ such that

$$\|z(t)\| \leq \alpha_z \|z_0\| e^{-\beta_z t},\ \forall t \geq t_0,\ \ \forall |z_0\| < \delta_z.$$

Since $-z(t) \leq x(t) \leq z(t)$, there is $\|x_0\|$ such that

$$\|x_0\| \leq \|z_0\| < \delta_z,$$

and

$$\|x(t)\| \leq \|z(t)\| \leq \alpha_z \|z_0\| e^{-\beta_z t},$$

Let $\alpha_c = \frac{\|z_0\|}{\|x_0\|} \geq 1$,then

$$\|x(t)\| \leq \|z(t)\| \leq \alpha_c \cdot \alpha_z \|x_0\| e^{-\beta_z t},$$

and

$$\alpha_c \cdot \alpha_z \geq 0.$$

This means $x(t)$ is exponentially stable. □

6.4.1 A Stable NMPC System

Now we augment the number of state variables by introducing an auxiliary linear state vector $z(t)$ Eq. (6.15) and add it into the optimal control problem (6.14) with holding Assumptions 6.1 to 6.2 and 6.6 on the model equations (6.14c). Thus the open-loop optimal control problem will be

$$\min_{x,z,u} J(x(t), u(t), t) = E(x(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t), t) dt. \tag{6.16a}$$

subject to

$$x(t_0) = x_0, \tag{6.16b}$$
$$\dot{x}(t) = f(x(t), u(t), t), \qquad \forall t \in [t_0, t_f], \tag{6.16c}$$
$$s(x(t), u(t), t) \geq 0, \qquad \forall t \in [t_0, t_f], \tag{6.16d}$$
$$\dot{z}(t) = A_z z(t),\ z(t_0) = Z_0 \in \mathbb{R}^{n_x}, \quad \forall t \in [t_0, t_f], \tag{6.16e}$$
$$u_{\min} \leq u(t) \leq u_{\max}, \qquad \forall t \in [t_0, t_f], \tag{6.16f}$$
$$-z(t) \leq x(t) \leq z(t), \qquad \forall t \in [t_0, t_f]. \tag{6.16g}$$

It should be noted that the initial value of the auxiliary state should be defined. Since $-z(t) \leq x(t) \leq z(t)$, $z(t) \in [0, x_{\max}]$, $-z(t) \in [x_{\min}, 0]$, we define the constant vector $Z_0 \in \mathbb{R}^{n_x}$ to represent the maximum and minimum values of the state vector $x(t)$ such that it is the maximum of the absolute values of the upper and lower state bounds in Eq. (6.14f) as shown in Fig. 6.4.

$$Z_0 = \max\{|x_{\max}|, \quad |x_{\min}|\} \tag{6.17}$$

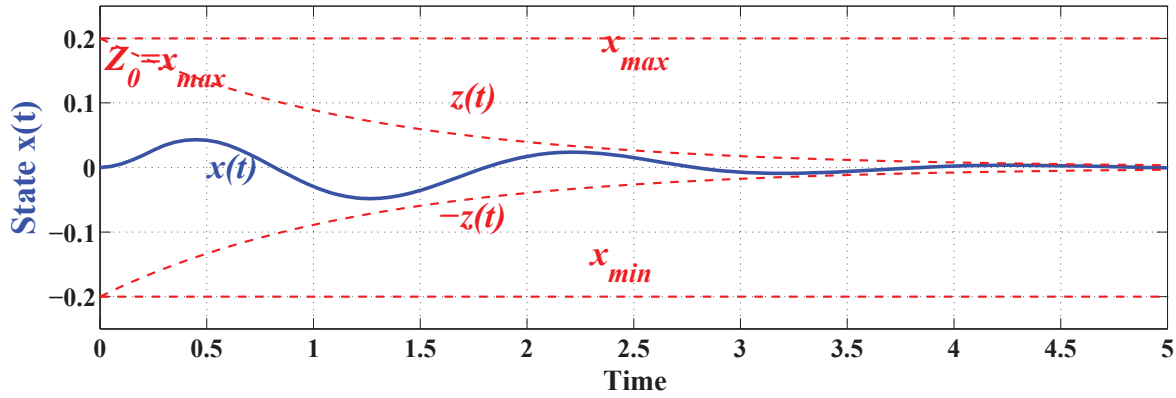

Figure 6.4: State $x(t)$ (blue-solid), auxiliary states $\pm z(t)$ (red-dashed), $-0.2 \leq x(t) \leq 0.2 \Rightarrow -z(t) \leq x(t) \leq z(t)$, $Z_0 = x_{\max}$.

On the other hand, the negative eigenvector a_z, which characterizes the auxiliary state vector $z(t)$, should be also defined. The model state is bounded at $t = t_0$ (i.e. $x(t_0) \in [-Z_0, Z_0]$) and will be more restricted as t increases as shown in Fig. 6.4. By applying Theorem 6.4, the system state vector will also be *exponentially stable*. If large negative eigenvalues are defined, the time constants of the auxiliary state vector $z(t)$ will be small and the model states will be restricted fast, therefore the optimal control problem can be infeasible. Furthermore, when Assumption 6.7 is satisfied by defining small negative eigenvector a_z, then the time constants of auxiliary states are large, and the model states will be slowly restricted by the auxiliary states, so the optimal control problem (6.16) can be feasible. In this case, if the time horizon is 'short', the effect of the state restriction cannot be noticed. The effect of the auxiliary state can only be obtained if a long time horizon is used. That means, the stability of model states and the auxiliary states can be ensured if negative eigenvalues of matrix A_z are defined such that: 1) large negative eigenvalues should not be used to maintain the feasibility and 2) maintaining the stability for optimal control problems with a 'short' time horizon, small negative eigenvalues of matrix A_z should not be defined.

To address the dilemma of defining suitable negative eigenvalues of matrix A_z in the optimal control problem (6.16) we let these eigenvalues be optimized in the new optimal control framework. Again, let Assumptions 6.1 to 6.3 be satisfied on the system model (6.14c), Assumption 6.5 is satisfied on the optimal control problem and the states augmentation is applied. Moreover, we consider the eigenvector a_z is an optimization variable in the optimal control problem. Therefore, the new open-loop optimal control problem setup

can be defined by holding Assumption 6.6

$$\min_{x,z,u,a_z} J(x(t), z(t), u(t), a_z, t) = E(x(t_f)) + \int_{t_0}^{t_f} (L(x(t), u(t), t) + \|z(t)\|^2_{\hat{Q}})dt, \tag{6.18a}$$

subject to

$$\begin{aligned}
&x(t_0) = x_0, && && (6.18b)\\
&\dot{x}(t) = f(x(t), u(t), t), && \forall t \in [t_0, t_f], && (6.18c)\\
&s(x(t), u(t), t) \geq 0, && \forall t \in [t_0, t_f], && (6.18d)\\
&u_{\min} \leq u(t) \leq u_{\max}, && \forall t \in [t_0, t_f], && (6.18e)\\
&\dot{z}(t) = A_z z(t),\ z(t_0) = Z_0,\ A_z = \mathrm{diag}(a_z), && \forall t \in [t_0, t_f], && (6.18f)\\
&-z(t) \leq x(t) \leq z(t), && \forall t \in [t_0, t_f], && (6.18g)\\
&a_z < 0, && && (6.18h)
\end{aligned}$$

where $x(t) \in \mathbb{R}^{n_x}$, $z(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$ and $a_z \in \mathbb{R}^{n_x}$.

Note that a quadratic penalty term with a weighting penalty matrix $\hat{Q}$ is added to the objective function to penalize the auxiliary state vector in the prediction horizon. The positive definite and symmetric weighting matrix $\hat{Q}$ can be selected or determined off-line. Based on Theorem 6.4, the exponential stability of the auxiliary state can be obtained near the operating point in the finite horizon. We solve this optimal control problem using the approach presented in Chapter 5.

Assumption 6.8:
The open-loop optimal control problem (6.18) has a local optimal control vector $u^*(t)$ with a corresponding optimal state vector and auxiliary state vector $x^*(t)$ and $z^*(t)$, respectively, that satisfy Definition 6.10. In addition, the state vector $x^*(t)$ is bounded by an optimal auxiliary state vector $z^*(t)$ which is characterized only by the optimal eigenvector a_z^*.

If the constraint (6.18g) is relaxed the optimal eigenvector a_z^* will be large negative value, since the penalty term of the objective should be minimized. On the other hand, if the penalty term of the objective function (6.18a) is relaxed the auxiliary state will be less restrictive and thus the optimal eigenvector a_z^* will be small negative value.

Theorem 6.5 (Open-Loop Exponentially Convergence):
Suppose that the system model(6.18c) has an equilibrium point at the origin with bounded and stabilizable states for $u \in [u_{\min}, u_{\max}]$ and Assumption 6.8 is satisfied on the open-loop optimal control problem (6.18) then the optimal state vector $x^*(t)$ is exponentially converges.

Proof. From Assumption 6.8, we have a point $(x^*, z^*, u^*, a_z^*) \in (\mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x})$ such that the optimal control problem (6.18) is feasible. Thus, it follows that

$$\begin{aligned}
&\dot{z}^*(t) = A_z z^*(t),\ z^*(t_0) = Z_0,\ A_z = \mathrm{diag}(a_z^*), && \forall t \in [t_0, t_f].\\
&-z^*(t) \leq x^*(t) \leq z^*(t) && \forall t \in [t_0, t_f].\\
&a_z^* < 0.
\end{aligned}$$

Since the eigenvector a_z^* is negative, then the trajectory vector $z^*(t)$ is exponentially stable and $\lim_{t\to\infty} z(t) = 0$. Therefore, from $-z^*(t) \leq x^*(t) \leq z^*(t)$, the system state $x(t)$ matches Theorem 6.4 which results that $x^*(t)$ is exponentially stable. □

6.4.2 Nonlinear MPC Algorithm to Ensure the Stability

According to the optimal control formulation (6.18), we propose a new RHC algorithm to guarantee the stabilization of nonlinear MPC systems.

Algorithm 8 A proposed RHC algorithm to ensure the stability

Inputs: Initial condition vector (x_0), prediction horizon (T_P). control horizon (T_C), sampling time Δt_i, system model Eq. (6.18c), path constraints Eq. (6.18d), objective function Eq. (6.18a), weighting matrix ($\hat{Q}$), control bounds (vectors $u_{\min}$ and $u_{\max}$), state bounds (vector $Z_0 \equiv \max\{|x_{\min}|,\ |x_{\max}|\}$).

Outputs: Optimal state and control trajectories, optimal value of the performance index and CPU time for each optimization cycle.

Step 0: Set $k = 0$.

Step 1: Solve the optimal control problem (6.18).
The result of this step is the control vector $u^*(t)$, $t \in [t_0, t_0 + T_P]$.

Step 2: Apply the control vector ($u^*(t),\ t \in [t_0, t_0 + T_C]$) to the system.

Step 3: Measure the states x_{k+1} at $t = t_0 + T_C$.

Step 4: Set $x_0 = x_{k+1}$

Step 5: Set $k = k + 1,\ t_0 = t_0 + T_C$ and $t_f = t_0 + T_P$ then go to Step 1.

In Algorithm 6.4 the optimization cycle k denotes solving the optimal control problem (6.18) in the prediction horizon $[t_k, t_k + T_P]$. Each optimization cycle $k = 0, 1, ...$ in the implementation of Algorithm 8, produces an optimal eigenvector a_{z_k} which steers the auxiliary state vector $z_k(t)$ that makes the optimal control problem feasible. That also means, the inequality (6.18g) must be satisfied in the time interval $[t_k, t_k + T_P]$, namely

$$-e^{A_{z_k}t} Z_0 \leq x_k(t) \leq e^{A_{z_k}t} Z_0, \qquad \forall t \in [t_k, t_k + T_P]. \tag{6.19}$$

From inequality (6.19) we note that the maximum and minimum state bounds in the first optimization cycle $k = 0$ will be Z_0 and $-Z_0$, respectively. In addition, in the following optimization cycles k, $k = 1, 2, ...$, the initial point of the auxiliary state vector $z_k(t_k)$ will be optimized, and according to the optimization the solution will be $e^{A_{z_k}t_k} Z_0$. Each optimization cycle produces the optimal eigenvector a_{z_k} which can be unequal to the optimal eigenvector $a_{z_{k+1}}$ in the next cycle. That means the maximum values of the auxiliary state vector of each optimization cycle $z_k(t_k)$, $k = 0, 1, ...$,

$$\begin{aligned}
z_0(t_0) &= e^{A_{z_0}t_0} Z_0 = Z_0, \\
z_1(t_1) &= e^{A_{z_1}(t_0+T_C)} Z_0 = e^{A_{z_1}T_C} Z_0, \\
&\vdots \\
z_k(t_k) &= e^{A_{z_k}(t_0+kT_C)} Z_0 = e^{A_{z_k}kT_C} Z_0,
\end{aligned}$$

in this way there can be $a^*_{z_k} \neq a^*_{z_{k+1}}$. It means that the auxiliary state profiles can be discontinues at the sampling points. To explain that more precisely, let the optimal control problem (6.18) be used to implement Algorithm 6.4.2 from the initial time point $t_0 = 0$. In the first optimization cycle the inequality (6.18g) will be set to

$$-e^{A_{z_0}t}Z_0 \leq x_0(t) \leq e^{A_{z_0}t}Z_0, \qquad \forall t \in [0, T_P],$$

which implies that the initial condition of the auxiliary state is Z_0. If the solution of the optimal control problem converged we will obtain an optimized vector $a^*_{z_0}$ such that the optimal control problem is feasible and stable around its equilibrium point. Now we increase k by 1 and solve the optimal control problem in the new horizon $[T_C, T_P + T_C]$. Then the inequality (6.18g) will be set to

$$-e^{A_{z_1}t}Z_0 \leq x_1(t) \leq e^{A_{z_1}t}Z_0, \qquad \forall t \in [T_C, T_P + T_C],$$

with a new initial condition of the auxiliary state $e^{A_{z_1}T_C}Z_0$. Then the vector a_{z_1} will be optimized again in the new moving horizon satisfying the feasibility and stability properties. Fig. 6.5 shows an example of auxiliary states in the optimization cycles $k = 0, 1, 2$ which are $5.0 \times 10^{-2}e^{-1.6t}$, $5.0 \times 10^{-2}e^{-2.0t}$ and $5.0 \times 10^{-2}e^{-2.4t}$, respectively. Thus the initial values of the auxiliary states in each cycle are $z_0(t_0) = 5.00 \times 10^{-2}$, $z_1(t_1) = 5.00 \times 10^{-2}e^{-1}$ and $z_3(t_3) = 5.00 \times 10^{-2}e^{-2.4}$, where $T_C = 0.5$.

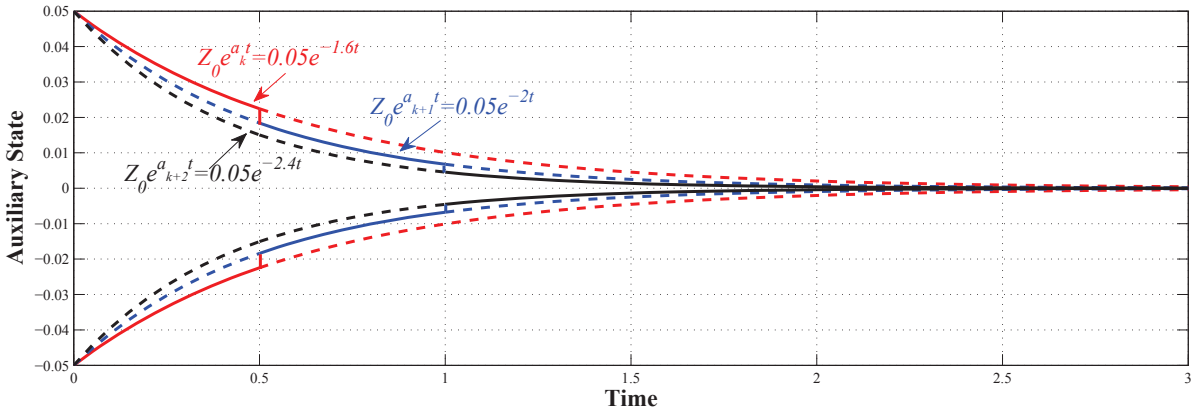

Figure 6.5: Auxiliary state in each optimization cycle.

In our formulation, although the number of the state variables in the setup of the optimal control problem is doubled, the size of the NLP problem produced from this problem will not significantly increase. This is because the auxiliary states introduced are linear, and therefore the additional size of the NLP problem comes only from the size of the eigenvector a_z. For example, if the proposed approach in Chapter 5 is used to solve the optimal control problem (6.18), the size of the NLP problem will be $n_w = (N + 1)(n_x + n_u) + n_x$[1]. On the other hand, in the setup of (6.18) we do not need to introduce additional terminal constraints for stability purposes, since the introduced auxiliary constraints will steer the model states to the equilibrium point. However, the new optimal control problem setup may cause an additional computational expense due to the restriction of the model states, see Eq. (6.19).

[1]Note that the size of the original NLP problem is $n_w = (N + 1)(n_x + n_u)$.

6.4.3 Stability Analysis of the Proposed Algorithm

In this Section, we prove that using the proposed NMPC algorithm an asymptotical stability of the closed-loop system can be guaranteed. We analyze the stability of the NMPC system when closed-loop system measurements are applied using Algorithm 8. If there are no disturbances, the feedback at every sampling time is not necessary since the prediction is exact [100]. When disturbances are considered, we assume that the plant is described by the following differential equations (instead of Eq. (6.18c)):

$$\dot{x}^p(t) = f(x^p(t), u(t), t) + d_k(t), \quad x^p(t_k) = x_k^p, \quad t \in [t_k, t_k + T_P]. \tag{6.20}$$

where x_k^p is the measured state vector at $t = t_k + T_C$, $k = 0, 1, ...$, and the disturbance $d_k(t)$ is bounded and asymptotically decaying with $|d_k(t)| \leq |D|$ and $|D| \leq |Z_0|$. We first analyze the effect of this disturbance in the open-loop case. Eq. (6.20) can be rewritten as:

$$\dot{x}_1^p(t) = f(x_1^p(t), u(t), t), \qquad t \in [t_k, t_k + T_P]. \tag{6.21a}$$
$$\dot{x}_2 = d_k(t), \tag{6.21b}$$
$$x^p(t) = x_1^p(t) + x_2(t). \tag{6.21c}$$

where $x_2(t) = \int_0^t d_k(t)dt$. If $\lim_{t\to\infty} x_2(t) = 0$ that means the disturbance that enters the system is equal to the disturbance that goes out from the system and the equilibrium point will not be changed. Otherwise, if $\lim_{t\to\infty} x_2(t) \neq 0$, i.e. the disturbance will case an additional amount added to the states and thus the equilibrium point will be changed. In this situation, a closed-loop control is needed to compensate the disturbance, so that the desired equilibrium point will be kept.

Applying Algorithm 8 to solve the NMPC system, the inequality constraints (6.18h) must be satisfied in each optimization cycle k for an optimal a_{z_k}, then the state vectors satisfy

$$-e^{A_{z_k}t} Z_0 \leq x_k(t) \leq e^{A_{z_k}t} Z_0, \tag{6.22}$$
$$-e^{A_{z_{k+1}}t} Z_0 \leq x_{k+1}(t) \leq e^{A_{z_{k+1}}t} Z_0, \tag{6.23}$$
$$\text{and } t \in [t_k, t_k + T_P]$$

where $x_k(t)$ and $x_{k+1}(t)$ are the solutions of Eqs. (6.24) and (6.25), respectively,

$$\dot{x}(t) = f(x(t), u(t), t), \quad x(t_k) \quad = x_k^p, \tag{6.24}$$
$$\dot{x}(t) = f(x(t), u(t), t), \quad x(t_{k+1}) = x_{k+1}^p, \tag{6.25}$$
$$k = 0, 1, ..., \text{ and } t \in [t_k, t_k + T_P].$$

Theorem 6.6 (Closed-Loop Asymptotical Convergence):
Suppose that Assumptions 6.1 to 6.3 are satisfied on the system model (6.18c). *If Algorithm 8 is applied to control the system dynamics then the closed-loop asymptotical stability will be yielded.*

Proof. Algorithm 8 implies that each optimization cycle we have an optimal point $(x_k(t), z_k(t), u_k(t), a_{z_k})$ where $t \in [t_k, t_{k+1}]$. Thus, it follows that $\forall t \in [t_k, t_{k+1}]$, $x(t)$ is exponentially decaying, since

$$\begin{aligned}
&\dot{z}_k(t) = A_{z_k} z_k(t),\ z_k(t_k) = e^{A_{z_k} t_k} Z_0,\ A_{z_k} = \mathrm{diag}(a_{z_k}), && \forall t \in [t_k, t_{k+1}],\\
&-z_k(t) \leq x_k(t) \leq z_k(t) && \forall t \in [t_k, t_{k+1}],\\
&a_{z_k} < 0.
\end{aligned}$$

and $x_k(t)$ matches Theorem 6.4 $\forall t \in [t_k, t_{k+1}]$.

The difference between dynamics of the plant and the model used in Algorithm 8 for $t \in [t_k, t_{k+1}]$ is given by

$$\dot{x}_k^p(t) - \dot{x}_k(t) = f(x_k^p(t), u_k(t), t) - f(x_k(t), u_k(t), t) + d_k(t) \tag{6.26}$$

Since the states of this model are updated at every t_k, we can integrate Eq. (6.26) for $t \in [t_k, t_{k+1}]$ and obtain

$$x_k^p(t) - x_k(t) = \int_{t_k}^{t} \left(f(x_k^p(\tau), u_k(\tau), \tau) - f(x_k(\tau), u_k(\tau), \tau)\right) d\tau + \int_{t_k}^{t} d_k(\tau) d\tau. \tag{6.27}$$

Since f is a Lipschitz continuous function (see Definition 4.2) then

$$\|f(x_k^p(t), u_k(t), t) - f(x_k(t), u_k(t), t)\| \leq L \|x_k^p - x_k\|, \tag{6.28}$$

where $L > 0$ is a Lipschitz constant. Therefore, using Eq. (6.27) and Eq. (6.28), we have

$$\|x_k^p(t) - x_k(t)\| \leq L \int_{t_k}^{t} \|x_k^p(\tau) - x_k(\tau)\| d\tau + \int_{t_k}^{t} \|d_k(\tau)\| d\tau. \tag{6.29}$$

Since $\|d\| \leq D$, we yield

$$\|x_k^p(t) - x_k(t)\| \leq L \int_{t_k}^{t} \|x_k^p(\tau) - x_k(\tau)\| d\tau + D(t - t_k). \tag{6.30}$$

To solve the inequality (6.30) we use a Bellman-Gronwall (BG) [59] lemma.

Lemma 6.1 (Bellman-Gronwall Lemma[59]):
Let $r : \mathbb{R}_+ \to \mathbb{R}$ locally integrable on $\mathbb{R}_+$ and $r \geq 0$; if

$$s(t) \leq c + \int_{t_k}^{t} r(\tau) s(\tau) d\tau \qquad \forall t \geq t_k,$$

then

$$s(t) \leq c \cdot e^{\left(\int_{t_k}^{t} r(\tau) d\tau\right)} \qquad \forall t \geq t_k.$$

Using Lemma 6.1, Eq. (6.30) can be rewritten for $k+1$ as

$$\|x^p_{k+1}(t) - x_{k+1}(t)\| \leq (t - t_k)De^{((t-t_{k+1})L)}. \tag{6.31}$$

But $\|x_{k+1}(t)\| \leq \|z_{k+1}(t)\|$. Thus, by applying triangular inequality, Eq. (6.31) will be

$$\|x^p_{k+1}(t)\| \leq \|z_{k+1}(t)\| + (t - t_k)De^{((t-t_k)L)}, \qquad \forall t \in [t_{k+1}, t_{k+2}],$$

or

$$\|x^p_{k+1}(t)\| \leq \|e^{A_{z_{k+1}}t}Z_0\| + (t - t_k)De^{((t-t_k)L)}, \qquad \forall t \in [t_{k+1}, t_{k+2}]. \tag{6.32}$$

Thus, by taking the limit $k \to \infty$

$$\lim_{k\to\infty} \|x^p_{k+1}(t)\| \leq \lim_{k\to\infty} \|e^{A_{z_{k+1}}t}Z_0\| + \lim_{k\to\infty} (t - t_k)De^{((t-t_k)L)}, \qquad \forall t \in [t_{k+1}, t_{k+2}]. \tag{6.33}$$

Since $\lim\limits_{k\to\infty} D = 0$ and $\lim\limits_{k\to\infty} \|e^{A_{z_{k+1}}t}Z_0\| = 0$, then

$$\lim_{k\to\infty} \|x^p_{k+1}(t)\| = 0, \tag{6.34}$$

that means the closed-loop system is asymptotically stable. □

Fig. 6.6 shows an example that illustrates the behavior of the closed-loop generated by the proposed NMPC for an asymptotically stable system. We note that the auxiliary state (red-dashed) enforces the system state (blue-solid) to decay exponentially in the first optimization cycle since the auxiliary state in the first optimization cycle satisfy $e^{A_{z_k}t}Z_0$. Although the initial conditions of the auxiliary state in two successive optimization cycles are equal, e.g. first and second ($k = 0, 1$), the time constants of the auxiliary states are not equal $-a_{z_0}^{-1} \neq -a_{z_1}^{-1}$, and the system state is also enforced to be enveloped by the auxiliary state. When many optimization cycles are done, say ($k > 100$) in this example, the time constants of the auxiliary state in two successive optimization cycles satisfy $-a_{z_k}^{-1} \cong -a_{z_{k+1}}^{-1}$. That means, when the MPC algorithm begins, the state decaying will be heavy, but when many optimization cycles are done the system decaying will decrease as k increases.

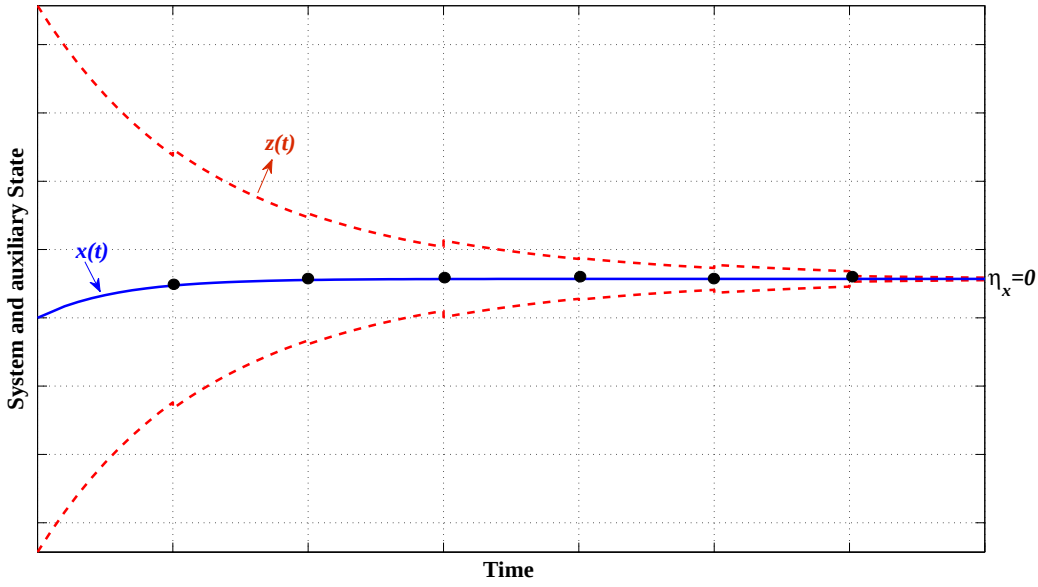

Figure 6.6: Asymptotical stability of a state trajectory.

If the system model, however, has an equilibrium point at $\eta = (\eta_x, \eta_u) \neq (0, 0)$, the auxiliary states will steer the system states to the shifted equilibrium point η. Thus, the optimal control problem (6.18) can be rewritten as

$$\min_{x,z,u,a_z} J(x(t), z(t), u(t), a_z, t) = E(x(t_f)) + \int_{t_0}^{t_f} (L(x(t), u(t), t) + \|(Z_0 - \eta_x)e^{A_z t}\|^2_{\hat{Q}})dt, \tag{6.35a}$$

subject to

$$x(t_0) = x_0, \tag{6.35b}$$
$$\dot{x}(t) = f(x(t), u(t), t), \qquad \forall t \in [t_0, t_f], \tag{6.35c}$$
$$s(x(t), u(t), t) \geq 0, \qquad \forall t \in [t_0, t_f], \tag{6.35d}$$
$$u_{\min} \leq u(t) \leq u_{\max}, \qquad \forall t \in [t_0, t_f], \tag{6.35e}$$
$$\dot{z}(t) = A_z z(t),\ z(t_0) = Z_0 - \eta_x,\ A_z = \mathrm{diag}(a_z), \qquad \forall t \in [t_0, t_f], \tag{6.35f}$$
$$\eta_x - e^{A_z t}(Z_0 - \eta_x) \leq x(t) \leq \eta_x + e^{A_z t}(Z_0 - \eta_x), \qquad \forall t \in [t_0, t_f], \tag{6.35g}$$
$$a_z < 0. \tag{6.35h}$$

6.5 Quasi-Infinite Horizon Method

I this Section, we use a quasi-infinite horizon method to determine the weighting matrix $\hat{Q}$ [40, 42]. First, let us select a sub-class of the open-loop optimal control problem with a

quadratic performance index and the setup of problem (6.18):

$$\min_{x,z,u,a_z} J(x(t),z(t),u(t),a_z,t) = \int_{t_0}^{t_f} (\|x(t)\|_Q^2 + \|u(t)\|_R^2 + \|z(t)\|_{\hat{Q}}^2)dt. \tag{6.36a}$$

subject to

$$\begin{aligned}
&x(t_0) = x_0, && && (6.36b)\\
&\dot{x}(t) = f(x(t),u(t),t), && \forall t \in [t_0,t_f], && (6.36c)\\
&s(x(t),u(t),t) \geq 0, && \forall t \in [t_0,t_f], && (6.36d)\\
&u_{\min} \leq u(t) \leq u_{\max}, && \forall t \in [t_0,t_f], && (6.36e)\\
&\dot{z}(t) = A_z z(t),\ z(t_0) = Z_0,\ A_z = \operatorname{diag}(a_z), && \forall t \in [t_0,t_f], && (6.36f)\\
&-z(t) \leq x(t) \leq z(t), && \forall t \in [t_0,t_f], && (6.36g)\\
&a_z < 0, && && (6.36h)
\end{aligned}$$

where $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$ are positive definite, symmetric matrices and design parameters of the problem (6.36) that can be chosen freely to determine the desired control performance.

To investigate the basic idea behind the quasi-infinite horizon method, we consider the following performance index

$$J^{\infty}(x,z,u,a_z,t) = \int_{t_0}^{\infty} (\|x(t)\|_Q^2 + \|u(t)\|_R^2 + \|z(t)\|_{\hat{Q}}^2)dt. \tag{6.37}$$

This performance index can be split up into two parts:

$$\begin{aligned}
J^{\infty}(x,z,u,a_z,t) = &\int_{t_0}^{t_f} (\|x(t)\|_Q^2 + \|u(t)\|_R^2 + \|z(t)\|_{\hat{Q}}^2)dt \ + \\
&\int_{t_f}^{\infty} (\|x(t)\|_Q^2 + \|u(t)\|_R^2 + \|z(t)\|_{\hat{Q}}^2)dt = J_1 + J_2
\end{aligned} \tag{6.38}$$

Now we assume that the prediction horizon, T_P, is chosen such that the system states and auxiliary states can reach a neighborhood of the equilibrium point $(\eta_x, \eta_y) = (0,0)$. Since the system states are always enveloped by the auxiliary states, both, system and auxiliary states, must exist in the neighborhood of the origin.

Rewriting the second part of the performance index (6.38) we yield

$$J_2 = \int_{t_f}^{\infty} (\|x(t)\|_Q^2 + \|u(t)\|_R^2))dt + \int_{t_f}^{\infty} \|z(t)\|_{\hat{Q}}^2 dt. \tag{6.39}$$

Our goal is to select a sufficient positive definite, possibly symmetric, weighting matrix $\hat{Q}$ such that the open-loop optimal control problem (6.36) is feasible.

Now, substitute $u = -Kx$ in the linearized system (6.36c) around the origin

$$\dot{x} = A_x x + B_u u \tag{6.40}$$

we get

$$\dot{x} = (A_x - B_u K)x = A_K x \tag{6.41}$$

with a linear state feedback law $u = -Kx$, that asymptotically stabilizes the linearized system in the terminal region. A linear-quadratic (LQ) state-feedback regulator for the state-space system (6.40), can be set-up, where K is given by

$$K = R^{-1}B^T P$$

and P is found by solving an algebraic Riccati equation.

$$A_x^T P + P A_x - P B_u R^{-1} B_u^T P + Q = 0$$

and Jacobian matrices $A_x = \frac{\partial f}{\partial x}\Big|_{(0,0)}$ and $B_u = \frac{\partial f}{\partial u}\Big|_{(0,0)}$.

Accordingly, Eq. (6.39) will be

$$\begin{aligned} J_2 &= \int_{t_f}^{\infty} (x^T Q x + (-Kx)^T R(-Kx))dt + \int_{t_f}^{\infty} z^T \hat{Q} z dt \\ &= \int_{t_f}^{\infty} x^T (Q + K^T R K)x dt + \int_{t_f}^{\infty} z^T \hat{Q} z dt \\ &= \int_{t_f}^{\infty} x^T (Q_K)x dt + \int_{t_f}^{\infty} z^T \hat{Q} z dt \end{aligned}$$

Since the inequality

$$-z(t) \leq x(t) \leq z(t)$$

holds, $\hat{Q}$ and Q_K are positive-definite, then

$$\int_{t_f}^{\infty} \|x(t)\|_{Q_K}^2 dt \leq \int_{t_f}^{\infty} \|z(t)\|_{\hat{Q}}^2 dt \tag{6.42}$$

which implies that the energy carried by vector z must be more than or equal the energy carried by vector x. Here we use the Lyapunov matrix equation to find a sufficient solution for $\hat{Q}$.

Theorem 6.7 (Lyapunov Matrix Equation [176]):
If there exist $Q_K > 0$ and $\hat{Q} > 0$ satisfy

$$A_K^T Q_K + Q_K A_K + \hat{Q} = 0 \tag{6.43}$$

then the linear system $\dot{x} = A_K x$ is globally asymptotically stable. The quadratic function $V(x) = x^T \hat{Q} x$ is a Lyapunov function that can be used to verify stability.

Proof. The proof of Theorem 6.7 can be found in [176].

Note that this solution is sufficient solution but not necessary unique to hold the feasibility assumption of problem (6.36) since we use the linearization of the system around the equilibrium point. In this case, the energy carried by the auxiliary state should be more than or equal to the energy carried by the system states.

However, we use the following steps to determine the weighting matrix $\hat{Q}$

- Linearize the system model (6.36c) around the origin using Eq. (6.40).
- Find the linear state feedback regulator K

  $$K = R^{-1}B^T P$$

  where P is found by solving the following algebraic Riccati equation:

  $$A_x^T P + P A_x - P B_u R^{-1} B_u^T P + Q = 0$$

- Find A_K using Eq. (6.40).
- Find $Q_K = Q + K^T R K$.
- Solve the Lyapunov equation for $\hat{Q}$, namely

  $$A_K^T Q_K + Q_K A_K + \hat{Q} = 0.$$

  Note that the weighting matrix $\hat{Q}$ can be non-diagonal matrix. However, for simplicity, we consider a diagonal weighting matrix by the following two steps:
  - Find the eigenvector of $\hat{Q}$.
  - Find $\hat{Q} = \mathrm{diag}(\lambda_{\hat{Q}})$.

6.6 Features at the Equilibrium Point

To analyze the features of the proposed NMPC at the equilibrium point, we assume that the linearized system model (6.36c) around the origin

$$\dot{x} = A_x x + B_u u,$$

where $A_x = (\partial f/\partial x)|_{(0,0)}$ and $B_u = (\partial f/\partial u)|_{(0,0)}$, and dynamics A_x has negative eigenvalues, that means, all the poles of the linearized system are stable, then it can be concluded that the system is stable at the equilibrium point, too. However, if A_x has some positive eigenvalues, that means the system has some poles on the right-hand side of the $s-$plane. The system states are augmented with an auxiliary state $z(t)$ and the open-loop optimal control problem (6.18) is solved. In addition, the prediction horizon is chosen such that the system and auxiliary states can reach a neighborhood of the equilibrium point. Then the optimal eigenvector a_z and the optimal auxiliary state z will replace the instable poles of the linearized system model and act as a compensator.

To prove this issue, let us analyze the system states at the terminal point, the solution of the open-loop optimal control problem (6.36) implies the feasibility for all $t \in [t_0, t_f]$

and t_f is chosen such that the system states and auxiliary states reach a neighborhood of the equilibrium point. Then the infinite optimal control problem can be defined in the neighborhood of the equilibrium point with respect to the linearized system model

$$\min_{x,z,u} J(x(t), z(t), u(t), t) = \int_{t_f}^{\infty} (\|x(t)\|_Q^2 + \|u(t)\|_R^2 + \|z(t)\|_{\hat{Q}}^2) dt. \tag{6.44a}$$

subject to

$$x(t_f) = x_f. \tag{6.44b}$$
$$\dot{x}(t) = A_x x(t) + B_u u(t), \qquad \forall t \geq t_f. \tag{6.44c}$$
$$\dot{z}(t) = A_z z(t),\ z(t_f) = Z_f,\ A_z = \mathrm{diag}(a_z), \qquad \forall t \geq t_f. \tag{6.44d}$$

where a diagonal matrix A_z solves the finite-open-loop optimal control problem (6.44) that contains only negative eigenvalues, x_f and Z_f are terminal values of the system and auxiliary state, respectively, that lie in the terminal region of the neighborhood of the origin which can be defined according to the following definition.

Definition 6.12 (Terminal Region):
A terminal region Ω is a region in terminal time such that the system and auxiliary state reach a neighborhood of the equilibrium point, that means:

$$\begin{bmatrix} x^T(t) & z^T(t) \end{bmatrix}^T \in \Omega \qquad t \geq t_f$$
$$\begin{bmatrix} x^T(t) & z^T(t) \end{bmatrix}^T \approx \begin{bmatrix} \eta_x^T & \eta_z^T \end{bmatrix}^T = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$

Since the inequality constraint (6.18g) *must be satisfied in the terminal region Ω, one of the following properties must be also satisfied, as shown in Fig. 6.4,*

$$x(t) = z(t) - \delta = z^-(t) \qquad t \geq t_f \text{ or} \tag{6.45a}$$
$$x(t) = -z(t) + \delta = -z^+(t) \qquad t \geq t_f. \tag{6.45b}$$

where δ is a small value that represents the deference between the system x and auxiliary state z in the terminal region which tends to zero.

Let us now rewrite the optimal control problem (6.44) satisfying property (6.45a), i.e. $x(t) = z(t), \quad t \geq t_f$, then the optimal control problem

$$\min_{x,z,u} J(x(t), z(t), u(t), t) = \int_{t_f}^{\infty} (\|[x^T(t) \quad z^T(t)]^T\|_{Q_{new}}^2 + \|u(t)\|_R^2) dt. \tag{6.46a}$$

subject to

$$[x^T(t_f) \quad z^T(t_f)]^T = [x_f^T \quad Z_f^T]^T. \tag{6.46b}$$
$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = A_{new} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + B_{new} u(t), \qquad \forall t \geq t_f. \tag{6.46c}$$

where $A_{new} = \begin{bmatrix} 0 & A_x \\ 0 & A_z \end{bmatrix}$, $B_{new} = \begin{bmatrix} B_u \\ 0 \end{bmatrix}$ and $Q_{new} = \begin{bmatrix} Q & 0 \\ 0 & \hat{Q} \end{bmatrix}$.

The solution of the open-loop optimal control problem (6.46) is equivalent to the problem solution by finding a linear feedback K_{new} such that

$$u = -K_{new}[x^T(t) \;\; z^T(t)]^T = [-K_x \;\; -K_z][x^T(t) \;\; z^T(t)]^T, \tag{6.47}$$

where $K_{new} \in \mathbb{R}^{n_u \times 2n_x}$ and $K_x,\ K_z \in \mathbb{R}^{n_u \times n_x}$ are designed to obtain the closed-loop asymptotical stability of system (6.44c) such that

$$K_{new} = R^{-1}B^T P_{new},$$

where P_{new} is a unique positive-definite solution and found by solving the following algebraic Riccati equation.

$$A_{new}^T P_{new} + P_{new}A_{new} - P_{new}B_{new}R^{-1}B_{new}^T P_{new} + Q_{new} = 0.$$

Combining Eq. (6.47) with Eq. (6.46c), we yield

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} -B_u K_x & A_x - B_u K_z \\ 0 & A_z \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}. \tag{6.48}$$

On the other hand, if the property Eq. (6.45b) is satisfied, i.e. $x(t) = -z(t),\ \ t \geq t_f$, then the solution of the open-loop optimal control problem is also equivalent to the closed-loop optimal control problem by finding a linear control law $u = [-K_x \;\; -K_z][x^T(t) \;\; z^T(t)]^T,\ \ t \geq t_f$ and Eq. (6.48) will then be

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} -B_u K_x & -A_x - B_u K_z \\ 0 & A_z \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}. \tag{6.49}$$

The eigenvalues of the closed-loop system (6.48) and (6.49) are given by those of the state feedback regulator dynamics $-B_u K_x$ together with those of auxiliary state system dynamics A_z. In case both matrices are asymptotically stable, then so is the closed-loop (6.48) and (6.49).

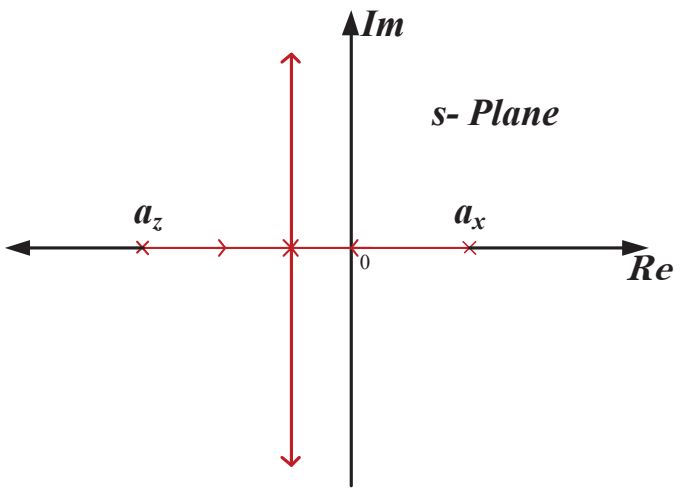

Figure 6.7: Root locus of one dimensional system dynamics.

Moreover, if the linearized system dynamics A_x has positive real parts of the eigenvalues, i.e. the system has some poles in the right-hand side of the $s-$plane, the feedback gain K_x will compensate the state dynamics to a closed-loop system with negative eigenvalues and brings the instable poles, in the open-loop case, to a stable poles as shown in Fig. 6.7.

7 Case Studies

As application examples of the proposed algorithm and the proposed stabilization approach we consider several NMPC problems in this Chapter. The following case studies are implemented using the framework of the numerical algorithm group (NAG) library Mark 8 [88] and the interior point optimizer (IPOPT) [178] and in C/C++ for the rest of the computation. In addition, all computations are done using a PC with an intel processor "Pentium 4, 3 GHz and 1G Byte RAM". In the implantation of the main Algorithm 6, we use three collocation points to discretize the ODE model equation and compute the constraints, Jacobian and performance index and their gradients. In addition, we define the end time point of a subinterval to be the beginning point of the next one.

Each case study in the following is solved, firstly, with an open-loop case to show the performance of the proposed algorithm. Then we apply each optimal control problem in the setup of NMPC. In addition, we examine the proposed stability approach within the NMPC system.

7.1 A Simple Instable System

To demonstrate the stability proposals in Chapter 6 we consider the following optimal control problem

$$\min_{x,u} J = \int_0^5 \left(\|x(t)\|_q^2 + \|u(t)\|_r^2 \right) dt. \tag{7.1a}$$

subject to

$$\dot{x}(t) = x(t) + u(t). \tag{7.1b}$$

where $x(0) = 1$, $u_{\min} = -1.5$, $u_{\max} = 1.5$, $q = 1$ and $r = 1$.

It is clear that exciting the system model Eq.(7.1b) with unit step control makes the state $x(t)$ instable, since the plant has an instable pole at $s = 1$. Therefore, the solution of the optimal control problem (7.1) leads to instable behavior of the state $x(t)$. Solving the problem using the setup of the optimal control problem (6.18) by adding an auxiliary state $\dot{z}(t) = A_z z(t), z(0) = 2$, an inequality constraint Eq. (7.3) and adding a new term, $(\int_0^5 \|z(t)\|_{\hat{Q}}^2 dt)$, to the objective functional with $\hat{Q} = 1$. The solution leads to the optimized value $a_z = -0.6127$, restricted and stabilized model state in the first prediction horizon as shown in Fig. 7.1.

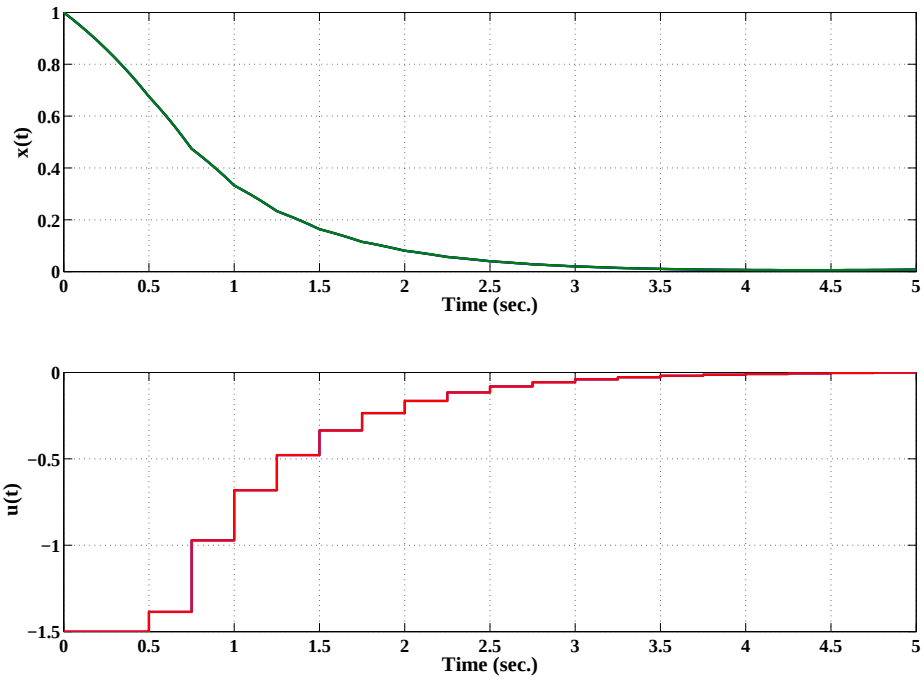

Figure 7.1: Solution of optimal control problem (7.1) with stable state behavior; state $x(t)$ and control $u(t)$.

Applying the features analysis in Section 6.6 to the solution of the optimal control problem with the auxiliary state $z(t)$, the closed loop augmented system (6.48) around the terminal point is

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \begin{bmatrix} -1 & -1.6201 \\ 0 & -0.6127 \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix}. \tag{7.2}$$

which means that the instable pole is compensated and the positive eigenvalue is shifted to make the dynamic asymptotically stable.

Now, we assume that an asymptotically decaying disturbance $d(t) = 0.2e^{-t}sin(2.5t)$ is added to the system model at each feedback measurement .

Repeating the optimization using MPC Algorithm 7 with control horizon $T_C = 2$ leads to the stabilized state as shown in Fig. 7.2. The optimal control problem solutions in the moving horizons after the first prediction horizon lead to the optimized values a_z= -0.61271, -1.46796, -7.41171, -500, -500..., respectively. In other words, the time constant of the auxiliary state is decreased as the time horizon is moved. Thus, the stabilization inequality constraint Eq.(7.3) is then more and more restrictive. Accordingly, the state asymptotical stability is guaranteed, namely

$$-2e^{a_z t} \leq x(t) \leq 2e^{a_z t}. \tag{7.3}$$

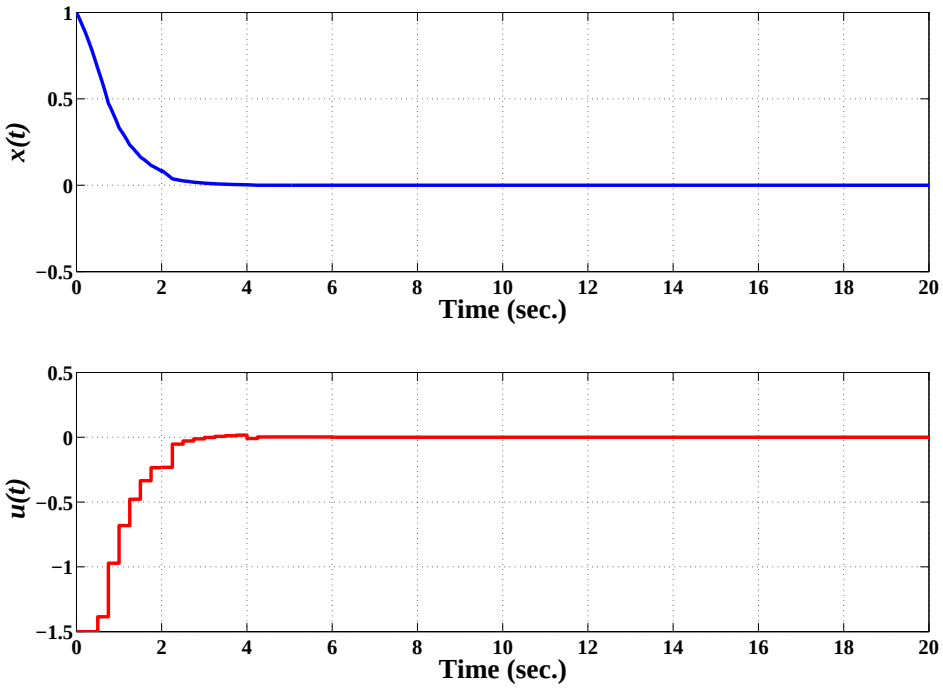

Figure 7.2: State $x(t)$ and control $u(t)$ of Section 7.1.

7.2 Batch Reactor

We consider a chemical reactor taken from [66] to maximize the yield of $x_2(t)$ after one hour operation by manipulating the reaction temperature $u(t)$, with the following problem formulation:

$$\max_{u,x_1,x_2} x_2(t_f) \tag{7.4a}$$

subject to

$$\dot{x}_1(t) = -(u(t) + (u^2(t))/2)x_1(t) \qquad t \in [0,1], \tag{7.4b}$$
$$\dot{x}_2(t) = u(t)x_1(t), \qquad t \in [0,1], \tag{7.4c}$$
$$x_1(0) = 1, x_2(0) = 0, \tag{7.4d}$$
$$0 \leq x_1(t), x_2(t) \leq 1, \tag{7.4e}$$
$$0 \leq u(t) \leq 5. \tag{7.4f}$$

We solve the open-loop control problem by dividing the prediction horizon into 20 subintervals. The solution took 250 ms and gave the of the performance index value with $x_2(t_f) = 0.57329$. Fig. 7.3(a) shows the optimal control trajectory and Fig. 7.3(b) the corresponding state profiles $x_1$and x_2 with the maximum error 1.269×10^{-5} and 1.1722×10^{-5}, respectively. These profiles (x_1 and x_2) and the optimal control trajectory are identical to the results by using both MSCOD II and the algorithm proposed. The CPU time for one subinterval simulation was 24 μseconds using CFE. Using the RK method with the same accuracy the CPU time was 42μseconds.

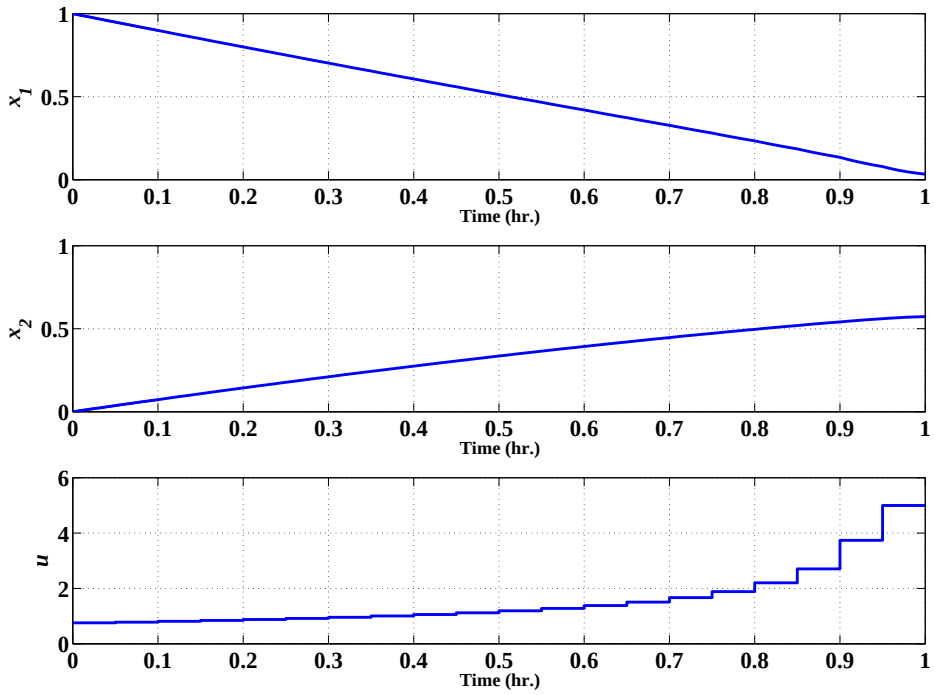

Figure 7.3: The optimal state and control trajectories $x_1(t)$, $x_2(t)$ and $u(t)$ of batch reactor.

If we solve this problem with different number of subintervals, e.g. 5, 10, 20, 40, 80 and 160 subintervals, we can note, as shown in Table 7.1, that the number of optimization variables n_w and the number of constrains n_B will be increased when the number of subintervals increases. The CPU time will increase exponentially. In addition, if we compare the CPU time taken by MUSCOD II with that of the proposed algorithm, it can be seen for a large number of subintervals (i.e. a high dimension of the NMPC) the proposed algorithm will be more efficient.

Table 7.1: Results of using different number of subintervals.

N	n_w	n_B	MUSCOD II		Algorithm 6	
			CPU-Time (ms)	J	CPU-Time (ms)	J
5	18	12	27	0.573117	188	0.568171
10	33	22	40	0.573080	290	0.572162
20	63	42	90	0.573527	350	0.573290
40	123	82	564	0.573544	480	0.573478
80	243	162	2177	0.573545	547	0.573528
160	483	322	12968	0.573545	735	0.573541

N: number of subintervals; n_w: total number of variables; n_B: total number of constraints; J: value of objective function.

7.3 Optimal Control of a Continuous Stirred Tank Reactor (CSTR)

We consider a CSTR as shown in Fig. 7.4. An exothermic, irreversible, first order reaction $A \rightarrow B$ occurs in the liquid phase and the temperature is regulated with external cooling. This example is taken from [90] or [136] with assumption that the level liquid is not constant.

The problem is formulated as follows:

$$\min_{x,u} \int_0^{t_f} \left((x_1 - x_1^s)^2 + 100(x_2 - x_2^s)^2 + 0.1(u_1 - u_1^s)^2 + 0.1(u_2 - u_2^s)^2\right) dt \tag{7.5a}$$

subject to

$$\dot{x}_1 = \frac{F_0 - u_1}{\pi r^2} \tag{7.5b}$$

$$\dot{x}_2 = \frac{F_0 c_0 - x_2}{\pi r^2 x_1} - k_0 x_2 \exp(\frac{-E}{R x_3}) \tag{7.5c}$$

$$\dot{x}_3 = \frac{F_0(T_0 - x_3)}{\pi r^2 x_1} + \frac{-\Delta H}{\rho C_p} k_0 x_2 \exp(\frac{-E}{R x_3}) + \frac{2U}{r \rho C_p}(u_2 - x_3) \tag{7.5d}$$

$$x_1(0) = 0.659, \quad x_2(0) = 0.877 \text{ and } x_3(0) = 324.5 \tag{7.5e}$$

$$0.5 \leq x_1 \leq 2.5, \qquad 0.8 \leq x_2 \leq 1.0 \tag{7.5f}$$

$$85 \leq u_1 \leq 115, \qquad 299 \leq u_2 \leq 301 \tag{7.5g}$$

where x_1 is the level of the tank, x_2 is the product concentration and x_3 is the reactor temperature. u_1 and u_2 are the outlet flow rate and coolant liquid temperature, respectively. In addition the inlet flow rate F_0 or the inlet concentration c_0 is considered as a disturbance to the CSTR. The desired steady-state operating points: x_1^s, x_2^s, u_1^s and u_2^s are $0.569m$, $0.884mol/L$, $100L/min$ and $299.5K$, respectively. The model parameters in nominal conditions are given in Table 7.2. It is assumed that, at the tenth minute a disturbance enters the plant at a level of $0.05mol/L$ on the inlet molar concentration c_0 where $t_f = 50\,\text{min}$.

Table 7.2: Parameters of the CSTR reactor.

F_0	$100\ L/\text{min}$	E/R	$8750\ K$
T_0	$305\ K$	U	$915.6\ Wm^{-2}K^{-1}$
c_0	$1.0\ mol/L$	ρ	$1\ kg/L$
r	$0.219\ m$	C_p	$0.239\ Jg^{-1}K^{-1}$
k_0	$7.2 \times 10^{10}\ \text{min}^{-1}$	ΔH	$-5 \times 10^4\ J/mol$

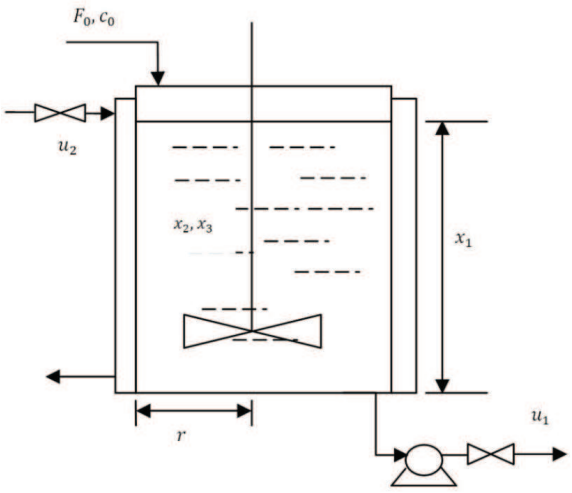

Figure 7.4: CSTR setup.

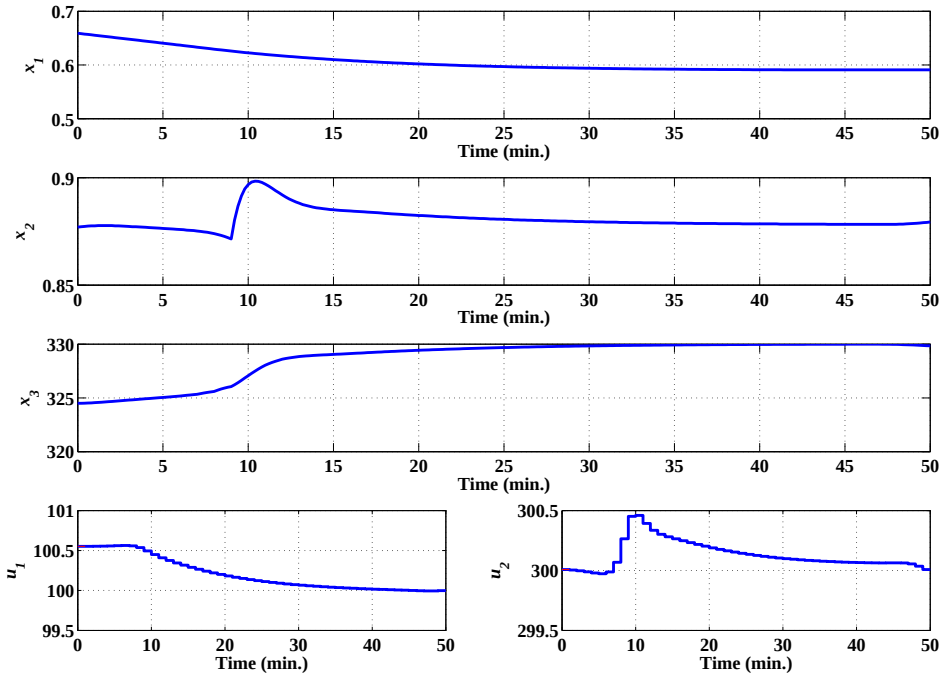

Figure 7.5: The optimal level of the tank $x_1(t)$, product concentration $x_2(t)$ and outlet temperature $x_3(t)$ of CSTR and the optimal control profiles, outlet flow rate $u_1(t)$ and coolant temperature $u_2(t)$ of CSTR.

To solve problem (7.5) using the proposed algorithm we divide the time horizon into 50 subintervals. The resulted NLP includes 306 variables with 204 constraints. We used the IPOPT 3.4.0 to solve the NLP and NAG mark 8 to solve the discretized model equations and compute the sensitivities. The CPU time for one subinterval simulation using CFE

and RK was 42 μseconds and 168 μseconds with a same accuracy, respectively. Fig. 7.5 shows the optimal control profiles of the states $x_1(t)$, $x_2(t)$, $x_3(t)$, and the optimal control profiles $u_1(t)$ and $u_2(t)$. The objective function value at the optimal solution is 0.9015886. Moreover, the solution converged in 35 iterations and took 984 mseconds. On the other hand, this problem was solved by [91] using a quasi-sequential approach to large scale dynamic optimization problem and it was converged in 16 iterations and 5.56 s of CPU time of SUN Ultra 10 Station with identical solutions.

We solve this optimal control problem by deviding the prediction horizon $[0, 40]$ into 80 subintervals. The optimal controls are shown in Fig. 7.6 where the optimal state profiles are shown in Figs. 7.8 to 7.10. Now we assume that an asymptotically decaying disturbance $d(t) = 0.1e^{-1.5t}\sin(5t)$ is added to the tank $x_1(t)$. Then the state profiles will change according to this disturbance as shown in the Figs. 7.8 to 7.10. We solve the optimal control problem (7.5) using the formulation of Eq. (6.18) by defining $\hat{Q}$=diag([1 1 1]), the equilibrium point $\eta_x = [0.536 \;\; 0.848 \;\; 330]^T$ and without adding any disturbance. In this case the NMPC formulation leads to the optimal controls that are shown in Fig. 7.7 that eigenvector $a_z = [-0.3138 \;\; -0.1503 \;\; -7.2844]^T$. In addition, the system states are forced to approach the equilibrium point η_x as shown in the Figs 7.8 to 7.10. Now we implement these controls with assuming that the disturbance $d(t)$ is entered to the tank. Therefore the systems states will not reach to the defined equilibrium point, since the disturbance term will cause an additional amount of the tank level, product concentration and outlet temperature, respectively. That means the equilibrium point will be changed due the disturbance, since $\lim_{t\to\infty} \int_0^t d(t)dt \neq 0$. See Figs. 7.8 to 7.10.

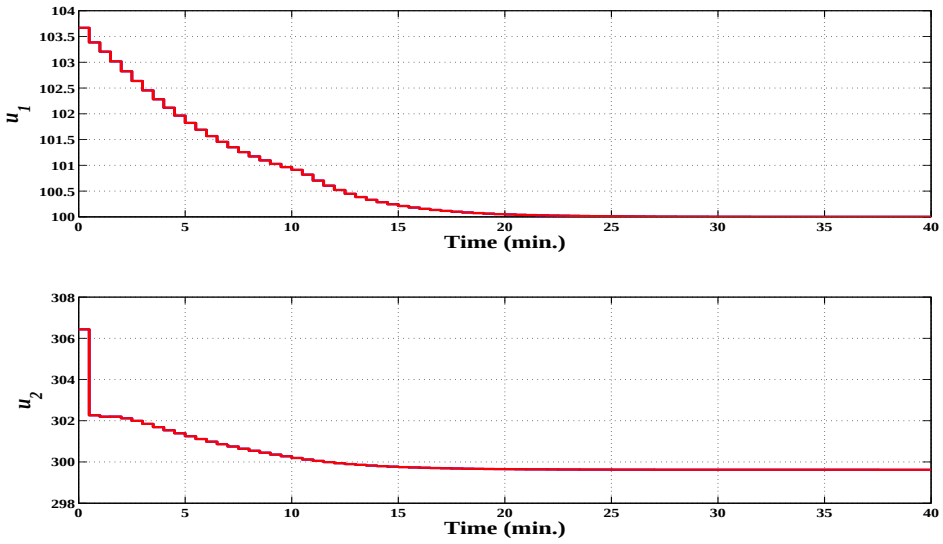

Figure 7.6: The open-loop optimal control profiles $u_1(t)$ and $u_2(t)$ of CSTR without adding the auxiliary constraints.

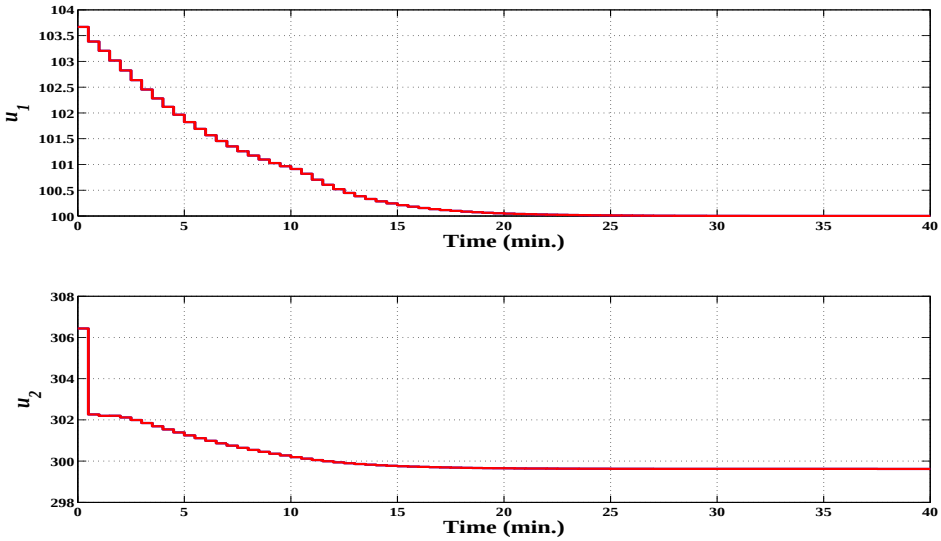

Figure 7.7: The optimal control profiles $u_1(t)$ and $u_2(t)$ of CSTR by adding the auxiliary constraints.

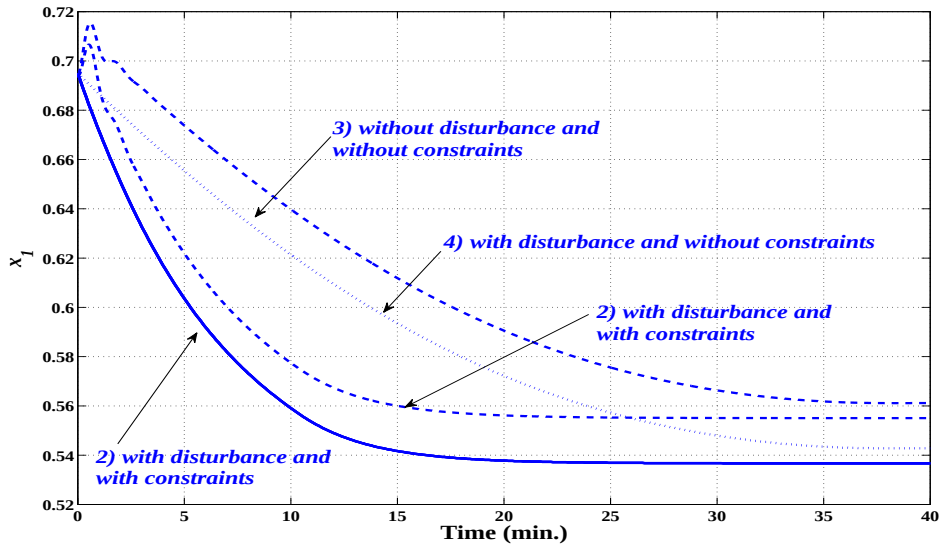

Figure 7.8: Optimal tank level in the open-loop case: 1)Without disturbance and with auxiliary constraints, 2)With disturbance and with constraints, 3)Without disturbance and without constraints and 4)With disturbance and with auxiliary constraints

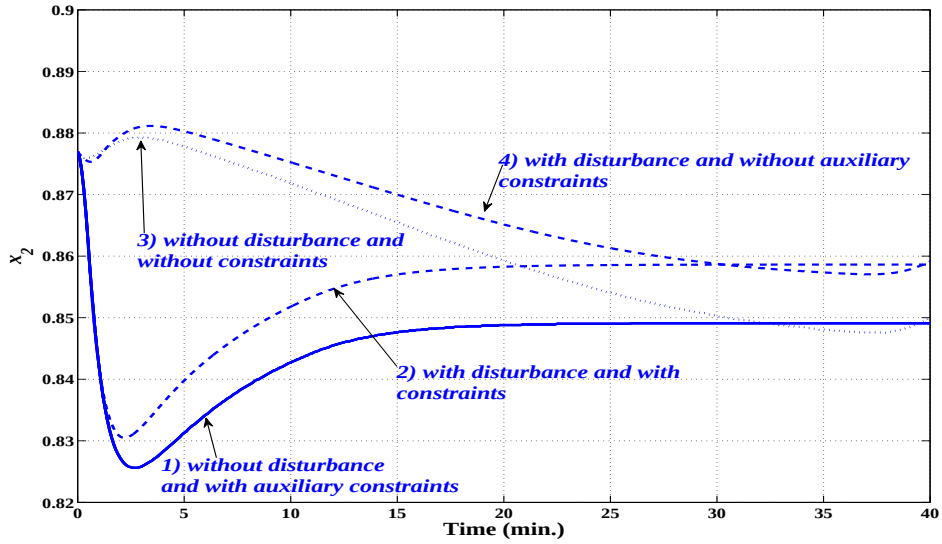

Figure 7.9: Optimal product concentration in the open-loop case: 1)Without disturbance and with auxiliary constraints, 2)With disturbance and with constraints, 3)Without disturbance and without constraints and 4)With disturbance and with auxiliary constraints

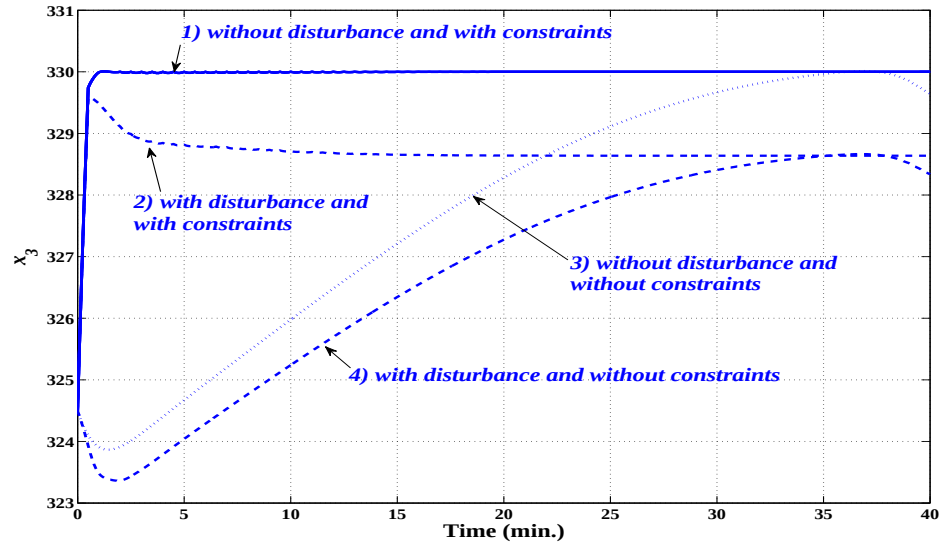

Figure 7.10: Optimal optimal outlet temperature in the open-loop case: 1)Without disturbance and with auxiliary constraints, 2)With disturbance and with constraints, 3)Without disturbance and without constraints and 4)With disturbance and with auxiliary constraints

To apply NMPC Algorithm 8 we choose the sampling time 0.5 min, $T_C = 0.5$ min and the prediction horizon $T_P = 40$ min, In addition the disturbance $d(t)$ enters to the tank. We solve this NMPC problem using a typical NMPC algorithm (i.e. without adding the auxiliary states) and using the proposed NMPC (i.e. by adding the auxiliary states

and constraints). The optimal control profiles for both cases, without and with auxiliary constraints, are shown in the Figs. 7.11 and 7.12, respectively, while the state profiles are shown in the Figs. 7.13 to 7.15. It can be seen that the auxiliary states and constraints force the system states to approach the operating point. In addition, with typical NMPC, the system states will approach the equilibrium point much more slowly than if the NMPC Algorithm 8 is used.

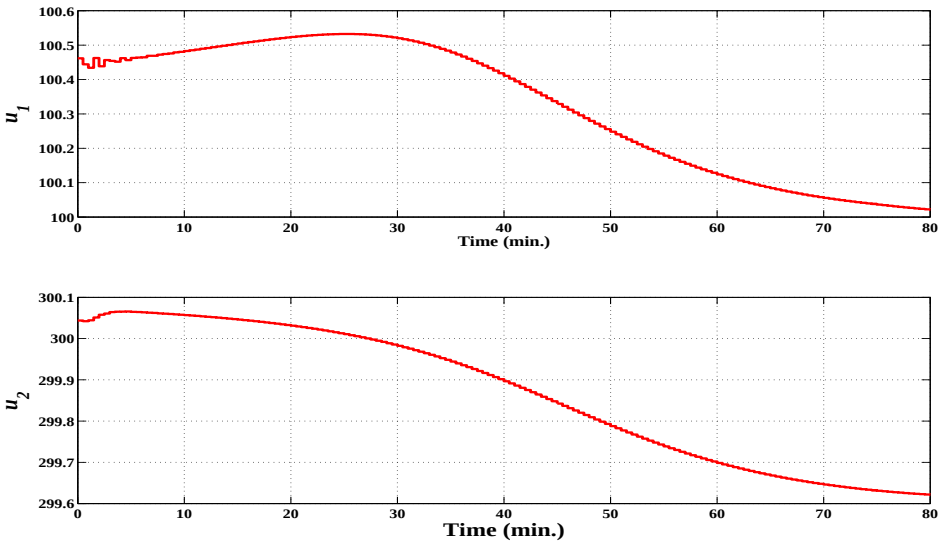

Figure 7.11: Closed-loop optimal control profiles without adding the auxiliary constraints.

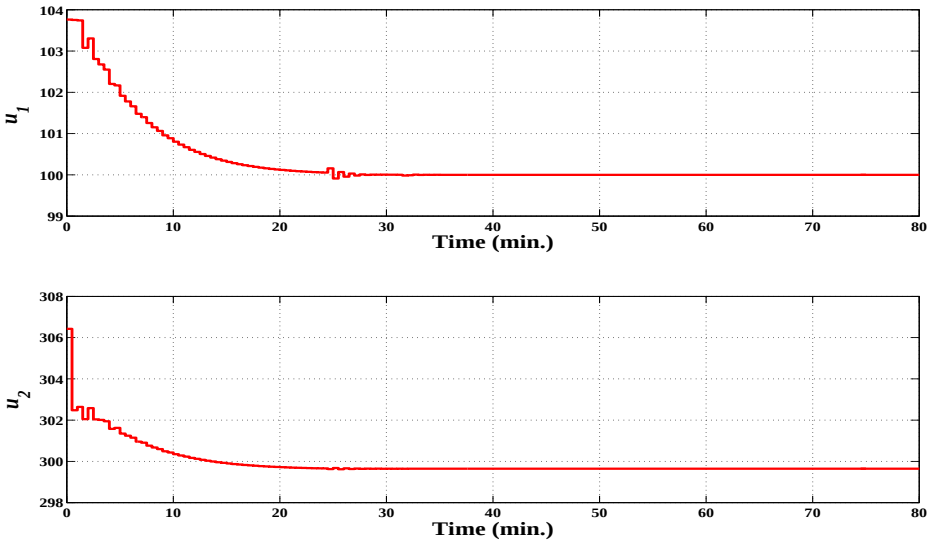

Figure 7.12: Closed-loop optimal control profiles by adding auxiliary constraints.

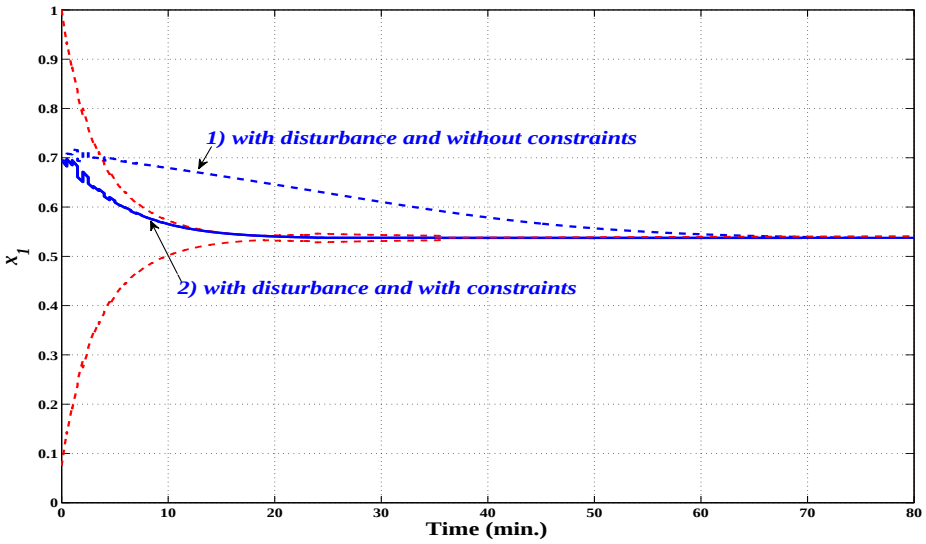

Figure 7.13: Optimal tank level in the closed-loop case: 1)Without adding the auxiliary constraints and 2)With adding the auxiliary constraints, auxiliary states $\pm z_1(t)$ (red-dashed).

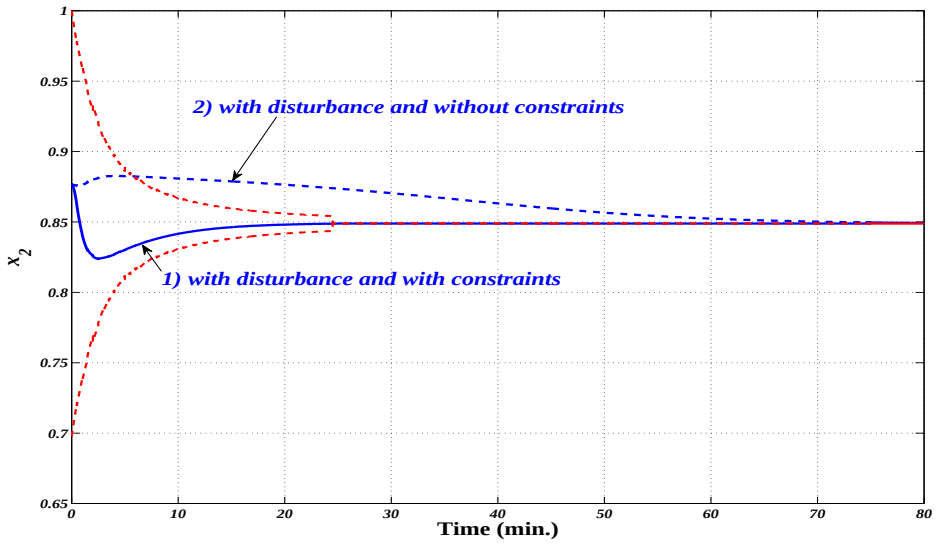

Figure 7.14: Optimal product concentration in the closed-loop case: 1)Without adding the auxiliary constraints and 2)With adding the auxiliary constraints, auxiliary states $\pm z_2(t)$ (red-dashed).

7.4 Satellite Control Problem

We consider a nonlinear optimal control problem of a rigid satellite initially undergoing a tumbling motion. The aim of the optimal control is to determine the torques that bring

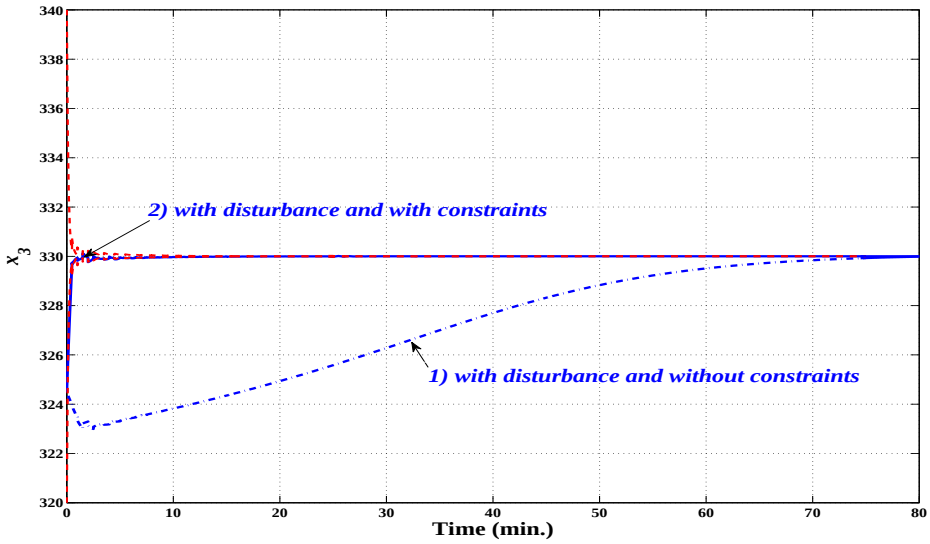

Figure 7.15: Optimal outlet temperature in the closed-loop: 1)Without adding the auxiliary constraints and 2)With adding the auxiliary constraints, auxiliary states $\pm z_3(t)$ (red-dashed).

the satellite to rest in the specified time $t_f = 100$ *sec.*, while minimizing the following performance index [70, 147]:

$$\begin{aligned}\min_{u,x}(x_1(100)-0.70106)^2+(x_2(100)-0.0923)^2+(x_3(100)-0.56098)^2\\ +(x_4(100)-0.43047)^2+x_5^2(100)+x_6^2(100)+x_7^2(100)\\ +\frac{1}{2}\int_0^{100}(u_1^2+u_2^2+u_3^2)dt\end{aligned} \tag{7.6a}$$

subject to

$$\dot{x}_1 = \frac{1}{2}(x_5x_4 - x_6x_3 + x_7x_2) \tag{7.6b}$$

$$\dot{x}_2 = \frac{1}{2}(x_5x_3 + x_6x_4 - x_7x_1) \tag{7.6c}$$

$$\dot{x}_3 = \frac{1}{2}(-x_5x_2 + x_6x_1 + x_7x_4) \tag{7.6d}$$

$$\dot{x}_4 = -\frac{1}{2}(x_5x_1 + x_6x_2 + x_7x_3) \tag{7.6e}$$

$$\dot{x}_5 = ((I_2 - I_3)x_6x_7 + \frac{T_{1S}}{I_1}u_1) \tag{7.6f}$$

$$\dot{x}_6 = ((I_3 - I_1)x_7x_5 + \frac{T_{2S}}{I_2}u_2) \tag{7.6g}$$

$$\dot{x}_7 = ((I_1 - I_2)x_5x_6 + \frac{T_{3S}}{I_3}u_3) \tag{7.6h}$$

where

$$
\begin{aligned}
&I_1 = 1.0 \times 10^6, \qquad I_2 = 833333, \qquad I_3 = 916667. \\
&T_{1S} = 550, \qquad T_{2S} = 50, \qquad T_{3S} = 550. \\
&x_1(0) = x_2(0) = x_3(0) = 0. \\
&x_4(0) = 1, \qquad x_5(0) = 0.01, \qquad x_6(0) = 0.005, \qquad x_7(0) = 0.001.
\end{aligned}
$$

Here u_1, u_2 and u_3 are the torques acting for the respective body-principle axes. The model equations (7.6b) to (7.6e) are called the kinematical equations associated with the orientation of the satellite. Model equations (7.6f) to (7.6h) are the dynamical equations associated with motion of the satellite. x_1 to x_4 are the Euler parameters, x_5 to x_7 are the angular rates of the satellite its principle axes and I_1, I_2 and I_3 are the principle moment of inertia, respectively.

We divide the time horizon into 50 subintervals, which leads to a NLP with 561 variables and 408 constraints. The computation took 531 m-seconds with the objective function value of 0.468287. For each subinterval it took 41 μseconds to simulate the system by CFE and 311 μseconds by RK. Figs. 7.16 and 7.17 show the state trajectories (from x_1 to x_7), and Fig. 7.18 shows the control profiles u_1, u_2 and $u3$, respectively.

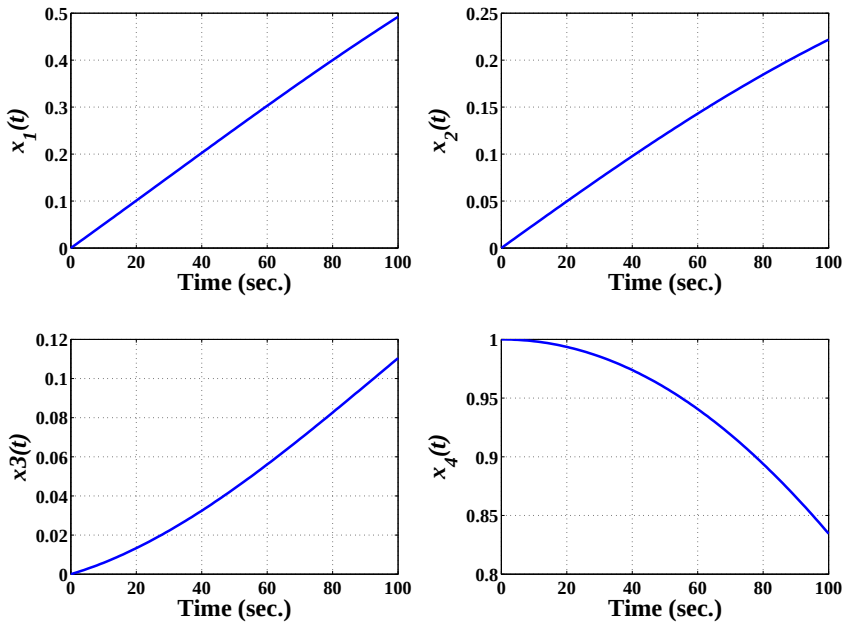

Figure 7.16: Satellite, state trajectories x_1, x_2, x_3 and x_4.

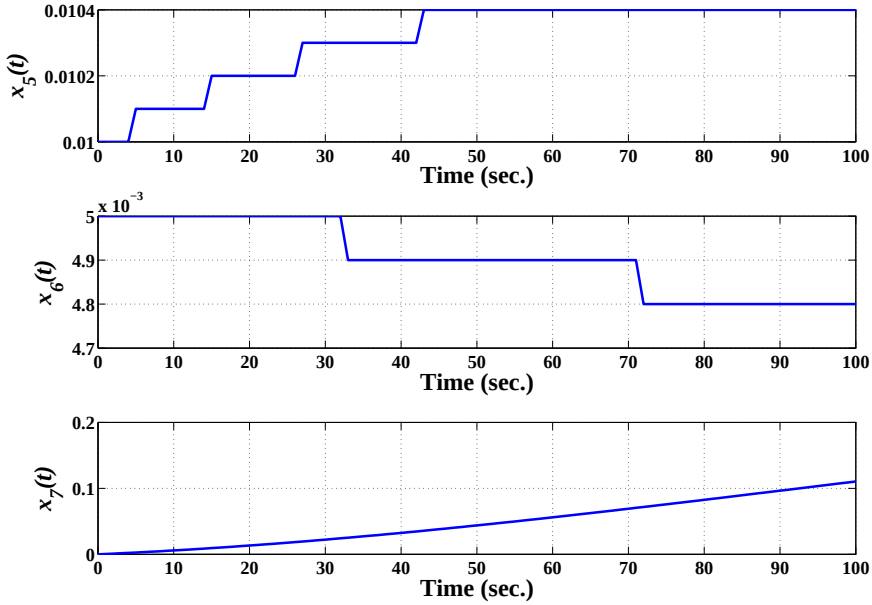

Figure 7.17: Satellite, state trajectories x_5, x_6 and x_7.

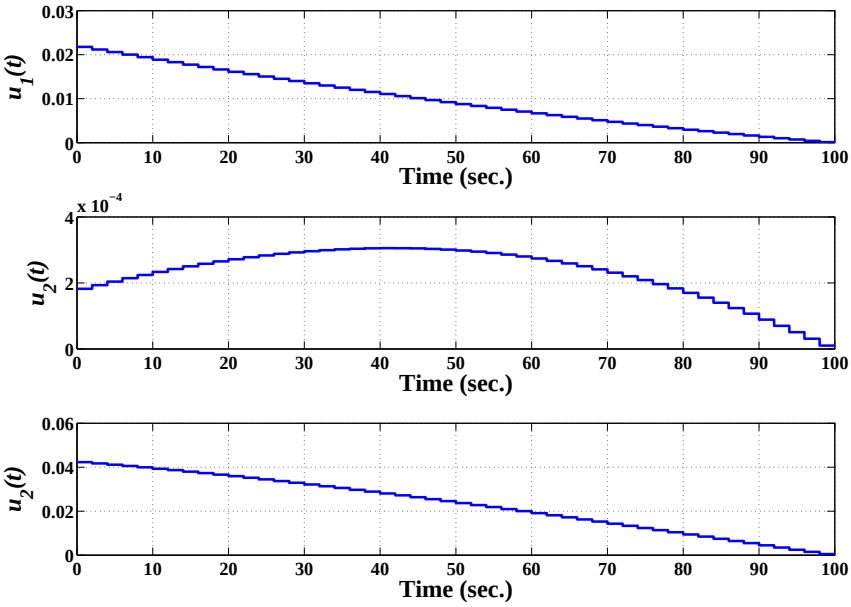

Figure 7.18: Satellite, control trajectories u_1, u_2, and u_3.

7.5 Nonholonomic System

This system is a car without a tail. The control of this model is a challenging problem, since the system is not controllable on the manifold of its equilibrium points [100]. Moreover,

this model violates the Brockett's necessary condition for smooth or even continuous stabilization [31]. The system dynamics can be represented by the following set of equations:

$$\dot{x}_1 = u_1 \cos(x_3). \tag{7.7a}$$
$$\dot{x}_2 = u_1 \sin(x_3). \tag{7.7b}$$
$$\dot{x}_3 = u_2. \tag{7.7c}$$

where $x_1(t)$ and $x_2(t)$ denote the Cartesian position of the center of the car, $x_3 \in (-\pi, \pi]$ represents the orientation of the car with respect of the x_1 axis. And u_1 and u_2 are the control inputs denoting linear and angular velocities, respectively. The coordinate system for the car is illustrated in Fig. 7.19.

The objective of the optimal control is to drive the system from any given initial condition to the origin with a satisfactory level of performance. The stable optimal control problem can be defined as

$$\min_{x,z,u,a_z} J = \frac{1}{2} \int_{t_0}^{t_f} \left(\|x(t)\|_Q^2 + \|u(t)\|_R^2 + \|z(t)\|_{\hat{Q}}^2 \right) dt. \tag{7.8}$$

subject to Eqs. (7.7a) to (7.7c) with initial condition $x(0) = [-0.3\ 0.8\ 1.2]$ and:

$$\dot{z}_i = A_z z, \quad A{=}\text{diag}(a_z). \tag{7.9}$$
$$z(0) = [1\ 1\ \pi]^T. \tag{7.10}$$
$$-z(t) \leq x(t) \leq z(t). \tag{7.11}$$

where $Q = \text{diag}([2\ 2\ 0])$, $\hat{Q} = \text{diag}([0.2\ 0.2\ 0.2])$, $R = \text{diag}([0\ 0])$, $t_0 = 0$, $t_f = 10$ sec and sampling time=0.5 sec. Figs. 7.20 to 7.22 show the open-loop control and control trajectories, respectively, according to the optimal solution using the proposed algorithm. The state trajectories approach the origin fast since the auxiliary state trajectories $\pm z_1(t)$, $\pm z_2(t)$ and $\pm z_3(t)$ approach the origin with time constants 1.521, 0.582 and 1.358 sec, respectively.

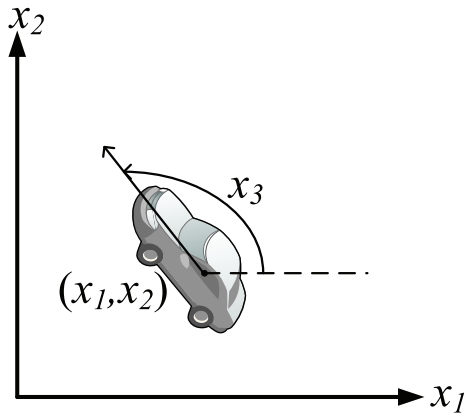

Figure 7.19: Coordinate system for the car.

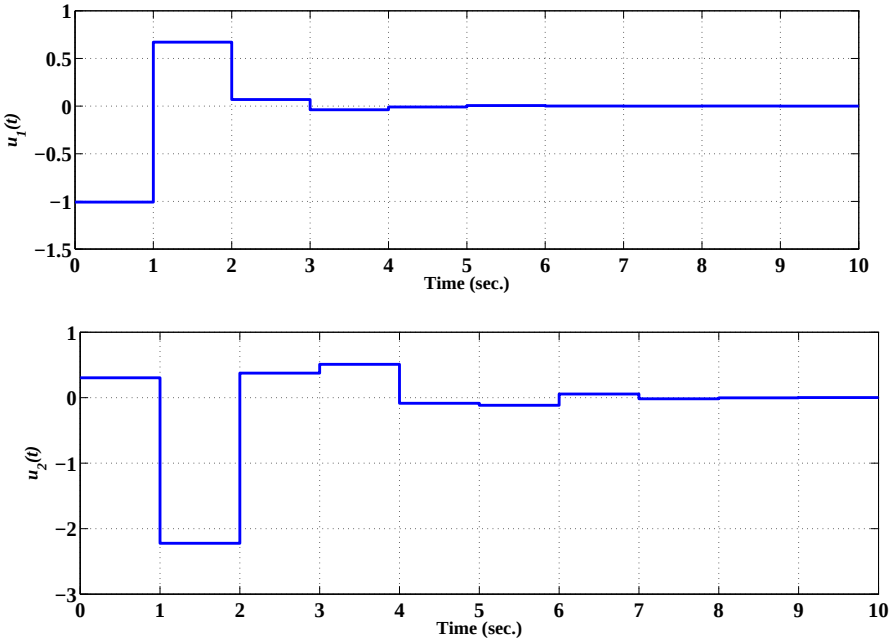

Figure 7.20: Open-loop optimal control solution of nanholonomic system.

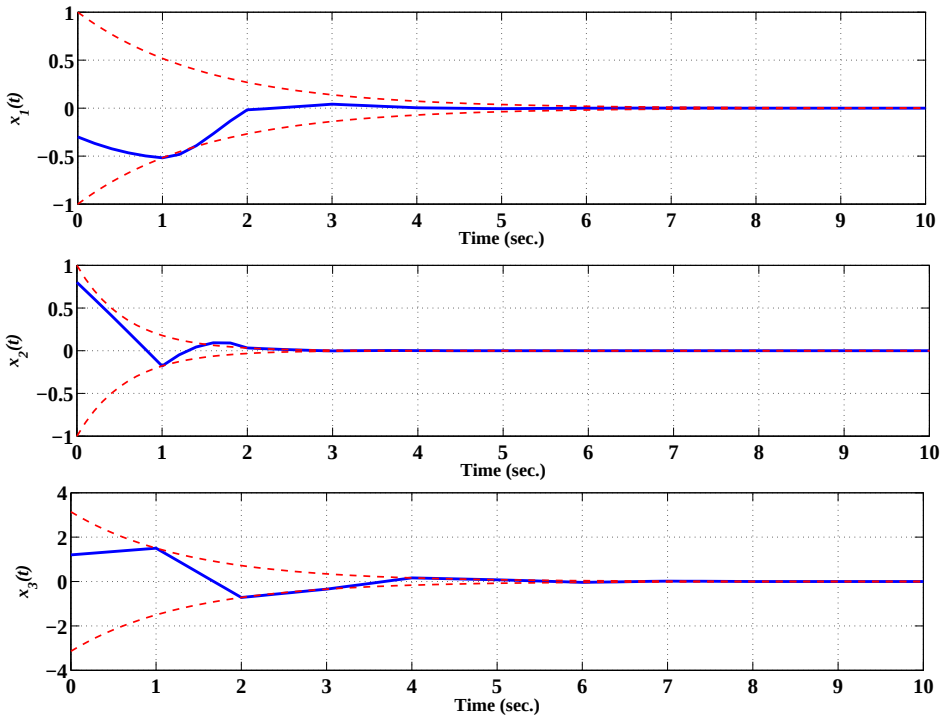

Figure 7.21: Open-loop optimal state profiles of nanholonomic system (blue-solid), auxiliary states $\pm z_i(t),\ i = 1, 2, 3$ (red-dashed).

For comparison purposes with result in [100], we apply the NMPC Algorithm 8 (closed system) to this system with $x(0) = [-0.5945 \quad 0.3299 \quad 0.8262]^T$, $u_{\min} = [-0.2 \quad -1]^T$, $u_{\max} = [0.2 \quad 1]^T$, Q=diag([2 16 0.2]), $\hat{Q}$=diag([0.2 1.8 0.02]), R=diag([0.02 0.02]), $t_0 = 0$, $t_f = 2$ sec, sampling time=0.1 sec, and $T_C = 0.6$ sec. The comparison to the CNTMPC [99, 100] shows that the controllers using both approaches with same bounds can stabilize the system states before 5 sec as shown in Figs. 7.22 to 7.23. In [100] the system states are stabilized by introducing a contractive inequality constraint in the optimal control problem. This inequality constraint imposes the system states at each sampling time to be contracted with respect to the states at the beginning of the predicted sample. With this strategy, an additional controller parameter is the so-called contractive parameter which must be carefully determined to make the optimal control problem feasible. However, in the proposed formulation, the auxiliary states in each sampling time will be optimized to satisfy the feasibility problem, i.e. a tuning is not required.In addition, using Theorem 6.6, the system states are asymptotically stable.

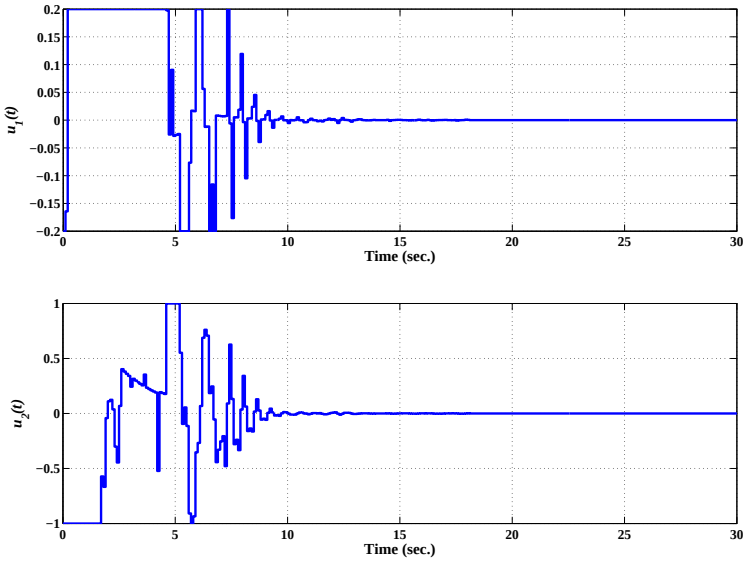

Figure 7.22: Closed-loop optimal control solution of nanholonomic system

7.6 Loading Bridge

7.6.1 Mechanical Setup

The mechanical setup of the loading bridge considered is shown in Fig. 7.24. It consists of a cart which can be moved along a metal guiding bar by means of a transmission belt. A winch drive is mounted on top of the cart to change the length of a rope. The transmission belt as well as the winch are driven by a current-controlled DC motor supplying a torque proportional to a control signal to accelerate the cart as well as the winch drive.

The position of the cart, the length of the rope as well as the angle of the rope are

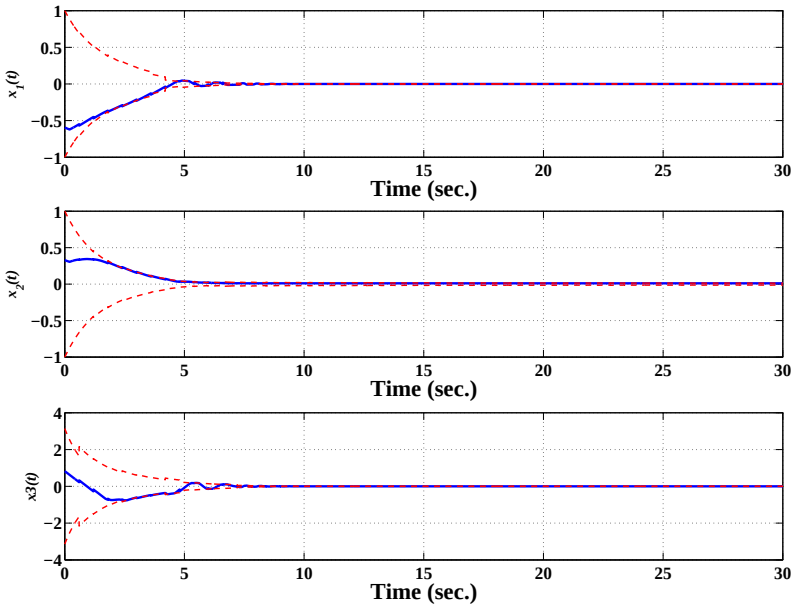

Figure 7.23: Closed-loop optimal state profiles of nanholonomic system (blue-solid), auxiliary states $\pm z_i(t)$. $i = 1, 2, 3$ (red-dashed).

measured by means of incremental encoders. The rope itself together with the load will be denoted as a pendulum as shown in Fig. 7.24.

7.6.2 Mathematical Model of a Loading Bridge

The complete system as shown in Fig. 7.24 is divided into the partial systems cart, winch, weight and center. The non-linear state space description of the of the loading bridge is given by [3]

$$\dot{x}_1 = x_2, \tag{7.12a}$$

$$\dot{x}_3 = x_4, \tag{7.12b}$$

$$\dot{x}_5 = x_6, \tag{7.12c}$$

$$\begin{aligned}\dot{x}_2 = &\frac{(u_1 - F_r x_2)(m_2 + \frac{\theta}{R_T^2}) + g m_2 \frac{\theta}{R_T^2} \sin(x_3)\cos(x_3)}{(m_1 + m_2 \sin^2(x_3))(m_2 + \frac{\theta}{R_T^2}) - m_2^2 \sin^2(x_3)} \\ &- \frac{(u_2 - F_{Tr} x_6) m_2 \sin(x_3) + x_4^2 x_5 m_2 \frac{\theta}{R_T^2} \sin(x_3)}{(m_1 + m_2 \sin^2(x_3))(m_2 + \frac{\theta}{R_T^2}) - m_2^2 \sin^2(x_3)},\end{aligned} \tag{7.12d}$$

$$\begin{aligned}\dot{x}_4 = &\frac{(u_2 - F_{Tr} x_6) m_2 \sin(x_3)\cos(x_3)}{x_5[(m_1 + m_2 \sin^2(x_3))(m_2 + \frac{\theta}{R_T^2}) - m_2^2 \sin^2(x_3)]} \\ &- \frac{g \sin(x_3)[m_1(m_2 + \frac{\theta}{R_T^2}) + m_2 \frac{\theta}{R_T^2}]}{x_5[(m_1 + m_2 \sin^2(x_3))(m_2 + \frac{\theta}{R_T^2}) - m_2^2 \sin^2(x_3)]}\end{aligned}$$

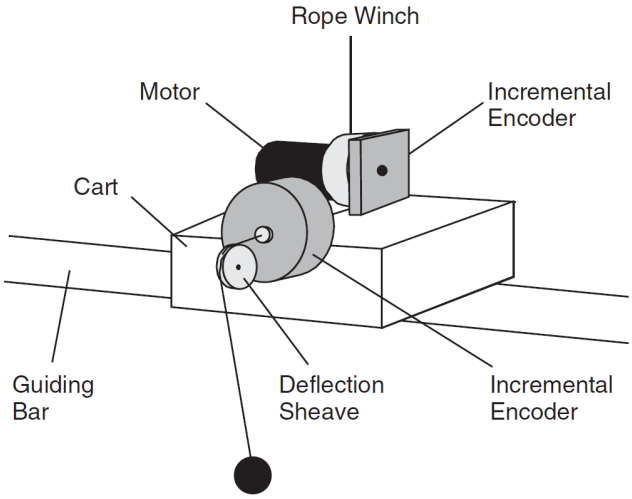

Figure 7.24: Schematics of the loading bridge.

$$
\begin{aligned}
&- \frac{x_4^2 x_5 m_2 \frac{\theta}{R_T^2} \sin(x_3)\cos(x_3)}{x_5[(m_1 + m_2\sin^2(x_3))(m_2 + \frac{\theta}{R_T^2}) - m_2^2\sin^2(x_3)]} \\
&- \frac{2x_4 x_6[(m_1 + m_2\sin^2(x_3))(m_2 + \frac{\theta}{R_T^2}) - m_2^2\sin^2(x_3)]}{x_5[(m_1 + m_2\sin^2(x_3))(m2 + \frac{\theta}{R_T^2}) - m_2^2\sin^2(x_3)]} \\
&- \frac{(u_1 - F_r x_1)(m_2 + \frac{\theta}{R_T^2})\cos(x_3)}{x_5[(m_1 + m_2\sin^2(x_3))(m_2 + \frac{\theta}{R_T^2}) - m_2^2\sin^2(x_3)]},
\end{aligned} \tag{7.12e}
$$

$$
\begin{aligned}
\dot{x}_6 &= \frac{(u_2 - F_{Tr} x_6)(m_1 + m_2\sin^2(x_3))}{(m_1 + m_2\sin^2(x_3))(m2 + \frac{\theta}{R_T^2}) - m_2\sin^2(x_3)} \\
&+ \frac{g m_1 m_2 \cos(x_3) + x_4^2 x_5 m_1 m_2}{(m_1 + m_2\sin^2(x_3))(m2 + \frac{\theta}{R_T^2}) - m_2\sin^2(x_3)} \\
&- \frac{(u_1 - F_r x_2) m_2 \sin(x_3)}{(m_2 + \frac{\theta}{R_T^2}) - m_2^2\sin^2(x_3)},
\end{aligned} \tag{7.12f}
$$

where the states x_1, x_2, x_3, x_4, x_5, and x_6 are the position of the cart , the velocity of the cart, the angle of the rope, the angular velocity of the rope, the length of the rope in m and the differentiation of the length of the rope, respectively. The controls u_1 and u_2 are the driving force of the cart and the winch, respectively. For more details on loading bridge and the model derivation see [3]. The parameters of the loading bridge model Eq. (7.12) are shown in Table 7.3.

7.6.3 Optimal Control Problem Formulation

The control task is to move the cart by means of the transmission belt to defined point at the metal guiding bar. In addition the movement should be carried out with a minimum

Table 7.3: Parameters of the loading bridge model.

P.	Description	Value
m_1	Cart mass	5.5 kg
m_2	Winch mass	0.2 kg
F_r	Friction constant on the cart	13 N
F_{tr}	Friction constant on the winch	2 N
θ	Moment of inertia of the winch	2.25×10^{-4} kg.m^2
g	Gravitational acceleration	9.81 m/s^2
R_T	Winch radius	3 cm

effort. Then the optimal control problem is formulated as

$$\min_{x,u} J = \frac{1}{2}\int_0^5 \left(\|x(t)\|_Q^2 + \|u(t)\|_R^2\right) dt. \tag{7.13a}$$

subject to the loading bridge dynamics (7.12) and

$$x(0) = [0.5\ 0\ 0\ 0.6\ 0]^T. \tag{7.13b}$$

$$u_{\max} = [22.5 \quad 3.75]^T,\ u_{\min} = [-22.5\ -3.75]^T. \tag{7.13c}$$

$$x_1(t_f) = 0.4, \qquad x_5(t_f) = 1 \tag{7.13d}$$

where $x(0)$ is the initial state vector, $u_{\min}$ and $u_{\max}$ are the minimum and maximum control bounds, respectively, and $x_1(t_f)$ and $x_5(t_f)$ are the terminal positions for the cart and the pendulum, respectively.

We solve the finite optimal control problem (7.13) by defining Q=diag([200 0.5 0.5 0.5 1 1]), R=diag([0.05 0.05]) and sampling time 0.25 sec. The optimal control problem will be converted into NLP problem which has 189 variables and 149 constraints. By solving this NLP problem the objective function value will be 1.5251. Fig. 7.25 shows the optimal controls with the corresponding state trajectories.

For comparison we solve the problem using the formulation setup of the optimal control problem (6.18) by adding an auxiliary state $\dot{z}_i = Az_i,\quad A_z$=diag$(a_z)\quad i = 1, 2..., 6$, $Z_0 = [0.6\ 1\ 0.5\ 1\ 1\ 1]^T$, $\hat{Q}$=dig([200 150 100 50 50 50]), the operating point η_x=[0.4 0 0 0 1 0]^T. Now the optimal control problem is formulated as

$$\min_{x,z,u,a_z} J = \frac{1}{2}\int_0^5 \left(\|x(t)\|_Q^2 + \|u(t)\|_R^2 + \|z(t)\|_{\hat{Q}}^2\right) dt. \tag{7.14a}$$

subject to the loading bridge system model (7.12), Eqs. (7.13b) to (7.13c) and

$$\dot{z} = A_z z,\quad A_z\text{=diag}(a_z). \tag{7.14b}$$

$$z(0) = [0.6\ \ 1\ \ 0.5\ \ 1\ \ 1\ \ 1]^T. \tag{7.14c}$$

$$\eta_x - z(t) \le x(t) \le \eta_x + z(t). \tag{7.14d}$$

where $z(t) \in \mathbb{R}^6$ are the auxiliary states, $A_z \in \mathbb{R}^{6\times6}$ is a time-invariant diagonal matrix and $\eta_x \in \mathbb{R}^6$ is the operating point around which the system should be reached.

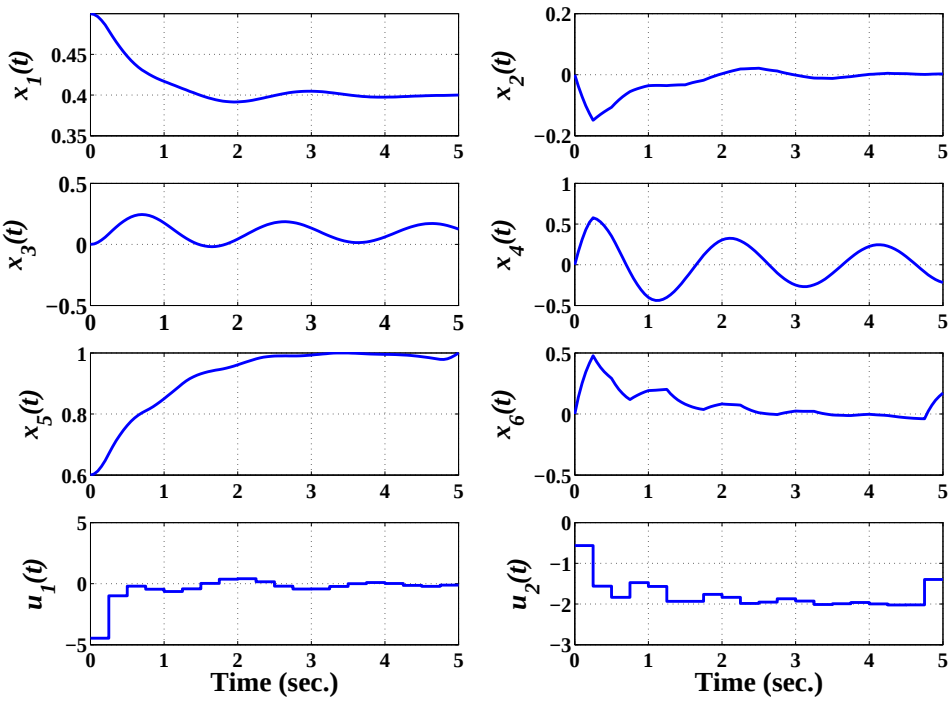

Figure 7.25: Corresponding optimal states of the loading bridge, $x_i(t)$, $i = 1, ..., 6$, optimal controls $u_1(t)$ and $u_2(t)$.

The optimal control solution leads to the objective function value of the main part (i.e. without the additional term) 1.8168. Fig. 7.26 shows that the system states are forced to approach the operating point η_x. If the solutions in this case are compared with the result shown in Fig. 7.27, the objective function value will be increased from 1.5251 to 1.8168. But the states $x_3(t)$, $x_4(t)$ and $x_6(t)$ do not approach the operating point in the defined time horizon without the stabilization constraints. This means the difference of the objective value 0.2918 is paid to guarantee the stability.

7.6.4 NMPC Formulation

We apply the NMPC Algorithm 8 to the loading bridge by choosing $T_C = 2$ sec. The NMPC is to solve optimal control problem (7.14) with the initial condition Eq. (7.13b) using the moving horizon technique. We assume that a disturbance d is added to system states at the beginning of each repeated optimization process. The NMPC solution leads to state and control trajectories as shown in Fig.7.28.

We apply the optimal control setup (7.14) to the NMPC system. Fig.7.29 shows the state and control trajectories in this case. It can be seen that the auxiliary states and constraints force the system states to approach the operating point in the first control horizon. Therefore, the asymptotical stability of system states around the operating point is obtained.

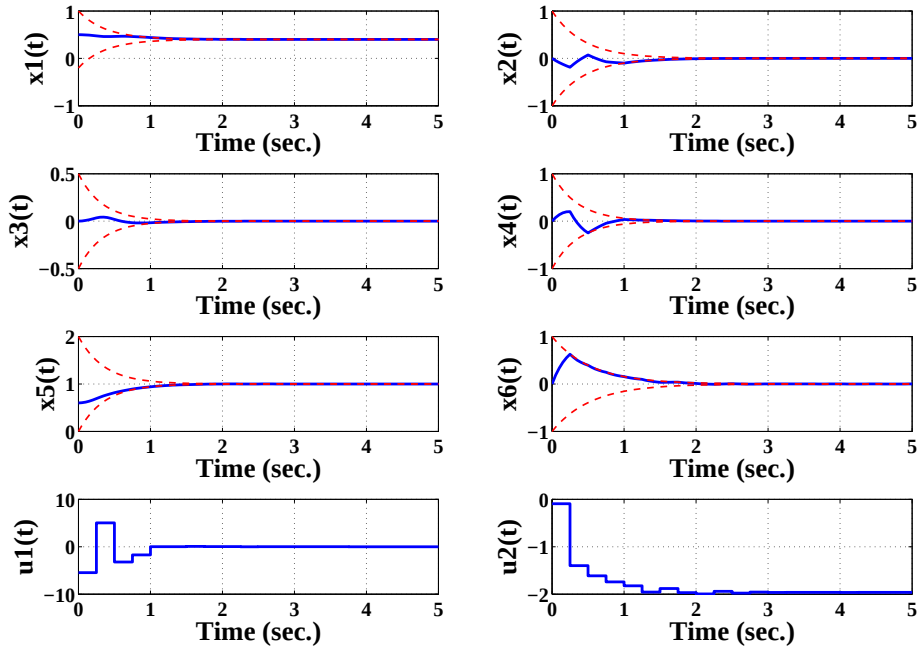

Figure 7.26: Optimal control problem solution (states and controls) of loading bridge, $x_i(t)$, $i = 1, ..., 6$ and $u_1(t)$ and $u_2(t)$ (blue-solid), auxiliary states $\pm z_i(t)$, $i = 1, ..., 6$ (red-dashed).

7.6.5 NMPC Realization

We realize the NMPC system of the loading bridge presented in Section 7.6.4 to the loading bridge laboratory unit [3]. We use the Matlab-Simulink environment to implement the optimal controls. The Simulink block diagram Fig. 7.30 shows the graphical user interface (GUI) that is built to control the loading bridge. Lookup-tables are used for the control of the cart and the pendulum. These controls are generated from a C-code which is prepared to solve the formulated optimal control problem. The control vector u is limited by a saturation block, for safety reasons, and applied to an analogue transducer which represents the motors of the cart and the pendulum. A digital encoder is used to measure the cart and pendulum positions and the necessary signals are used for the control.

We apply the open-loop optimal controls u_1 and u_2 as shown in Fig.7.31 using the proposed approach with the following weight matrices R =diag[0.45 0.001] and Q =diag[361 5 0.3 0.2 254 20]. In addition, it is desired the the cart and the pendulum move from (0.2, 0.8) to (0.8, 0.5), respectively. The results of cart and pendulum movements are shown in Fig. 7.31. Fig. 7.32 shows the weight movement on the cartesian plane. We apply the open-loop optimal controls u_1 and u_2 by solving the optimal control problem using the optimal control strategy (6.18). Fig. 7.33 shows cart and pendulum movements wile Fig. 7.34 shows the weight movement on the plan.

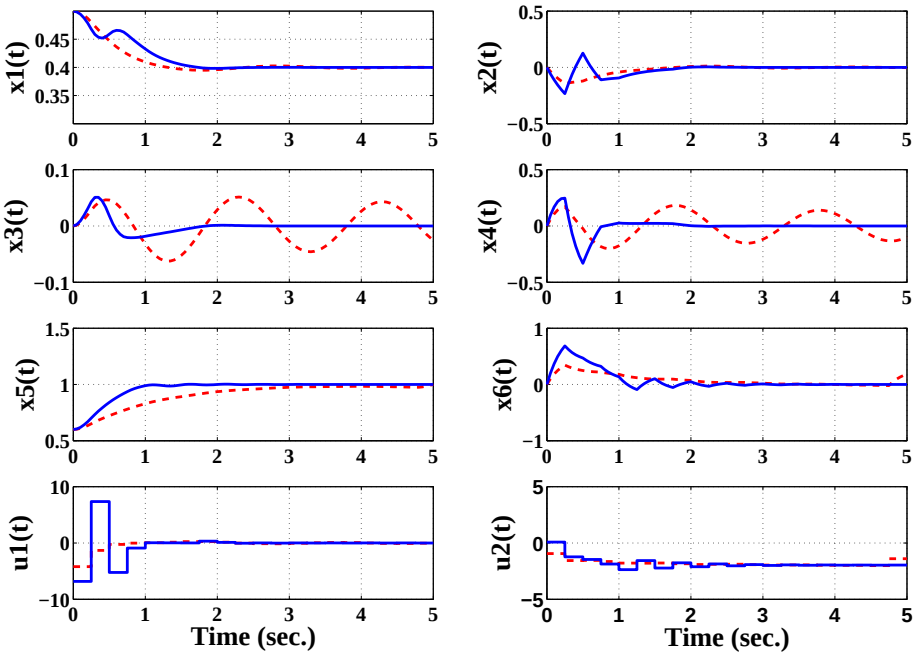

Figure 7.27: Optimal control problem solutions (states and controls) of loading bridge with and without adding auxiliary states, $x_i(t)$, $i = 1, ..., 6$. With auxiliary state (blue-solid), without auxiliary state (red-dashed).

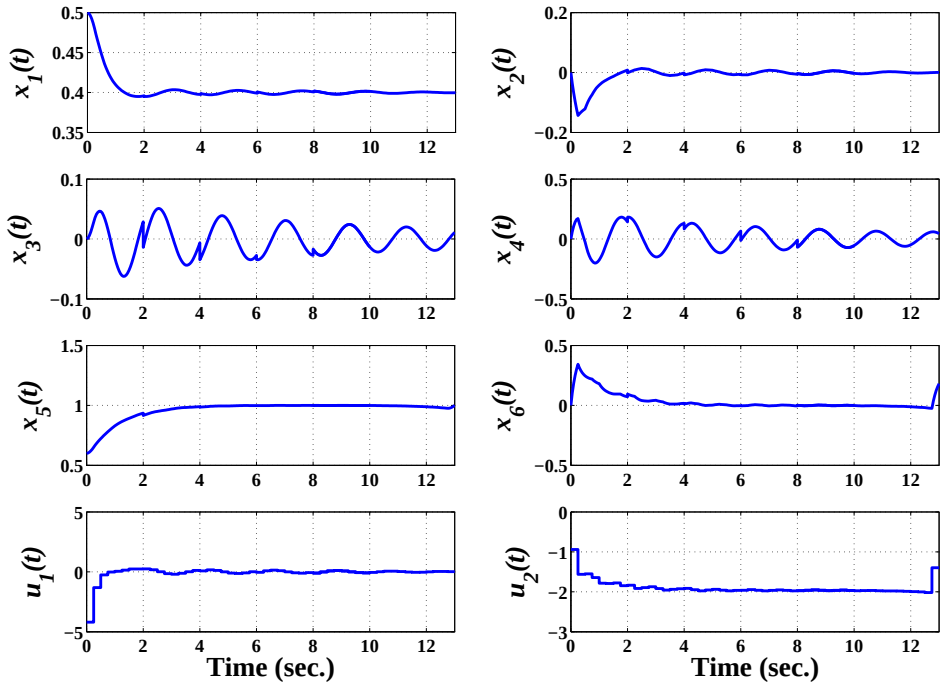

Figure 7.28: MPC solution (states and controls) of the loading bridge, states: $x_i(t)$, $i = 1, ..., 6$, and controls: $u_1(t)$ and $u_2(t)$.

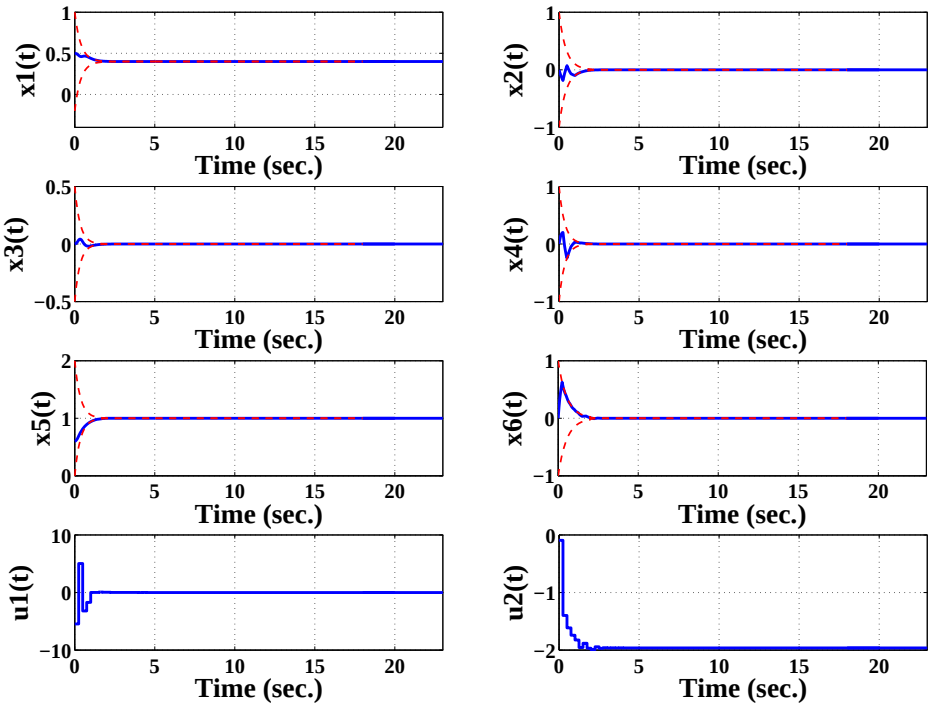

Figure 7.29: MPC solution (states and controls) of loading bridge, states: $x_i(t)$, $i = 1, ..., 6$, and controls: $u_1(t)$ and $u_2(t)$ (blue-solid), auxiliary states $\pm z_i(t)$, $i = 1, ..., 6$ (red-dashed).

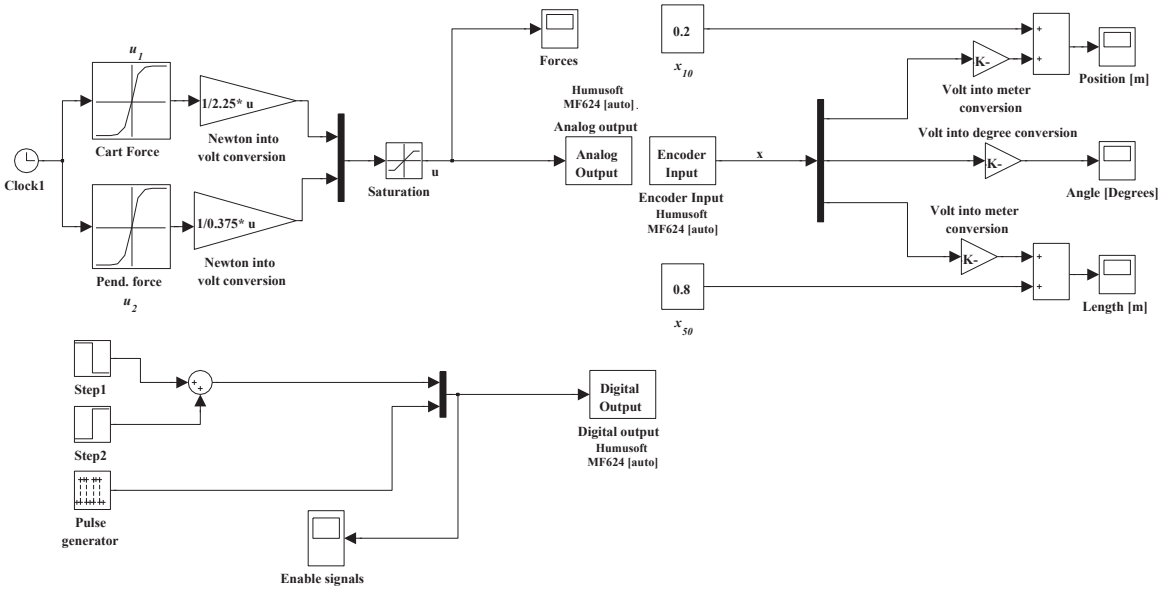

Figure 7.30: NMPC realization using Simulink interface.

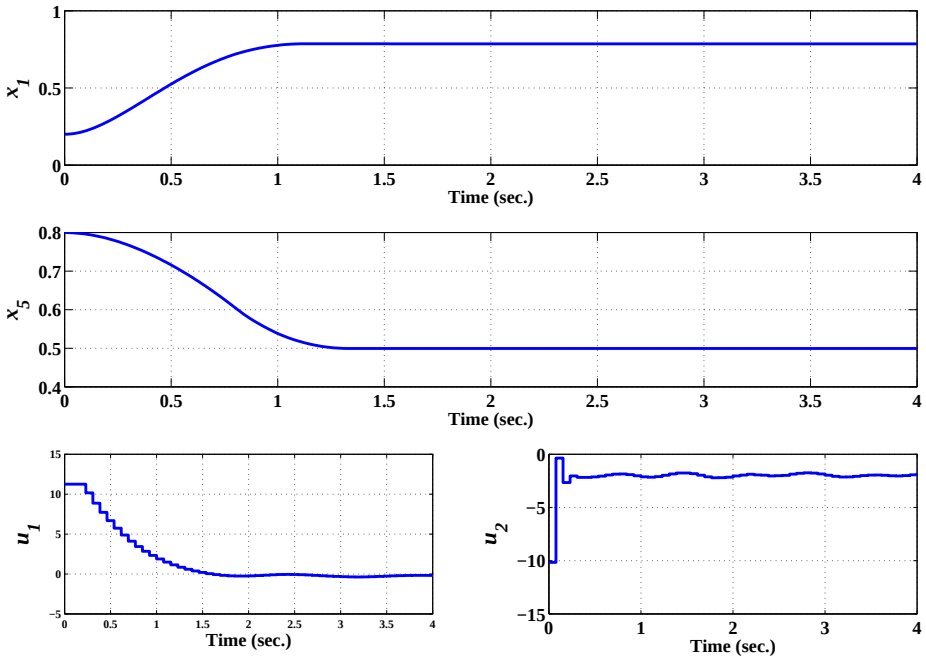

Figure 7.31: Open-loop control of the loading bridge.

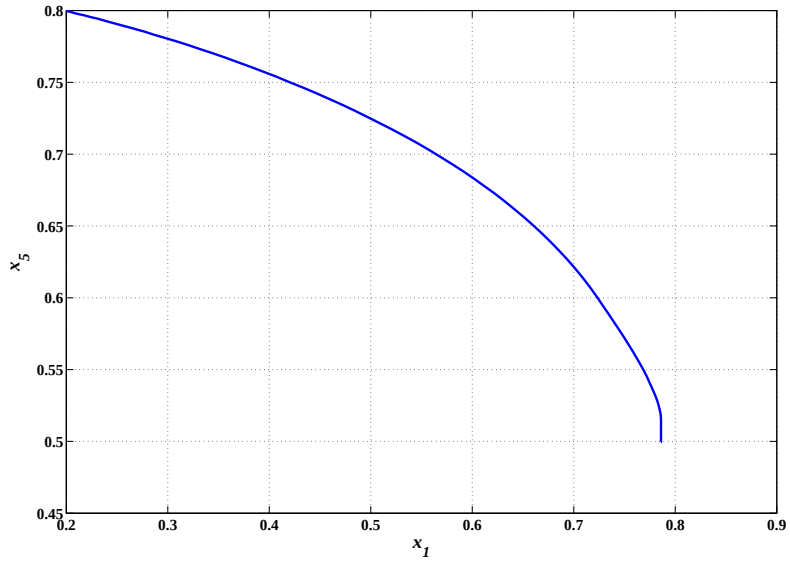

Figure 7.32: Open-loop control of the loading bridge (the cart and pendulum movements).

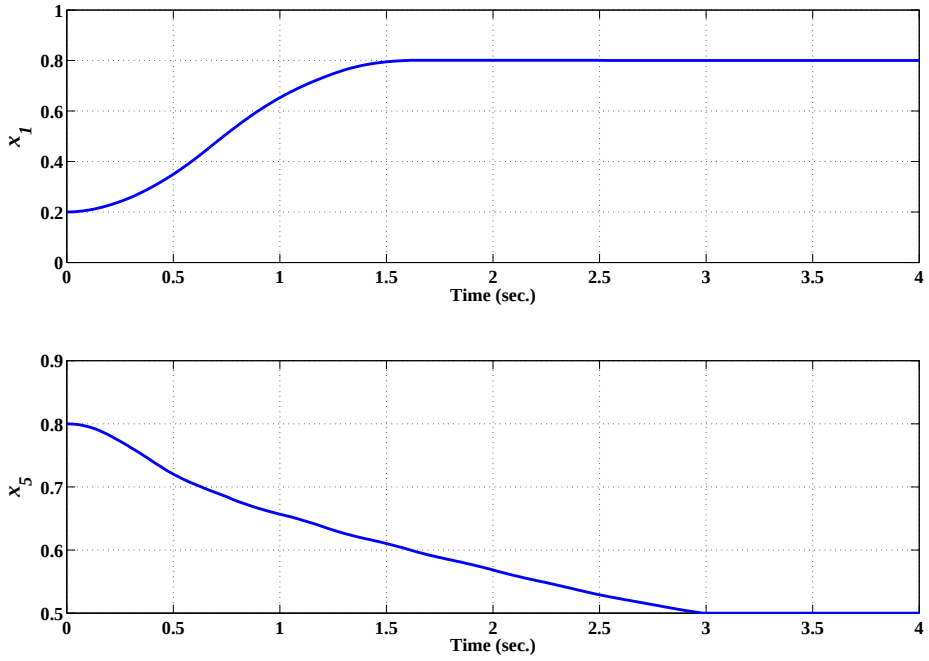

Figure 7.33: Open-loop control of the loading bridge using auxiliary constraints.

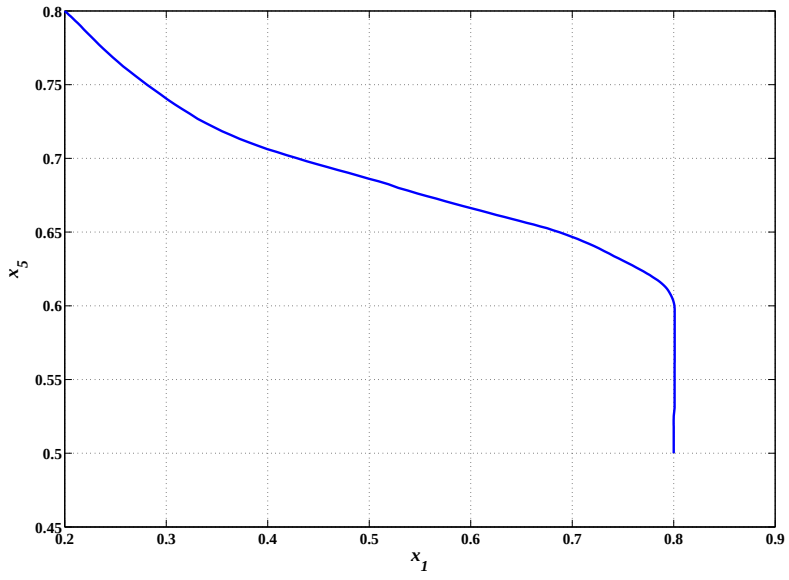

Figure 7.34: Open-loop control of the loading bridge (the cart and pendulum movements) using auxiliary constraints.

8 Conclusions and Future Work

8.1 Conclusions

In this dissertation, we proposed a novel algorithm for the solution of optimal control problems as they apear in NMPC. This NMPC algorithm is based on solving the nonlinear optimal control problem using a combination of the multiple shooting method, in which the NLP problem will be handled, with the collocation method, in which the function values and gradients required in the NLP will be computed.

We consider a general formulation of the constrained optimal control problem of continuous systems. An efficient discretization method is used to discertize the optimal control problem. A piecewise constant representation is used to parameterize the control variables on each subinterval and a pointwise parametrization is used for initial conditions of the state variables in each subinterval. Then all of the discretized and parameterized variables as well as system model equations are used to construct a NLP problem. To compute state variables in each subinterval and the sensitivities of these variables with respect to parameterized controls and initial conditions we use collocation on finite elements.

Using collocation on finite elements to calculate state values and its derivatives instead of using an DAE solver adds more accuracy to the DAE approximation, i.e. the accuracy of the optimal control solution will be enhanced. The truncation error of the state variable will be significantly reduced if the roots of Lagender polynomials are used to locate the positions of the collocation points in each element. while more truncation error will be yielded in the state variables if a typical DAE solver is used.

In addition, as shown in collocation on finite element method is often less expensive than typical DAE solvers. Since the collocation on finite elements method approximates each state variable by an interpolating function in each time interval with a number of collocation points. This system of nonlinear equations is solved using an efficient root-finding technique, e.g. Newton-Raphson, to compute the collocation points. Furthermore, the collocation on finite elements method needs only to solve a system of linear equations, which is obtained by taking the first order Taylor expansion of the system of nonlinear equations, to compute the necessary sensitivities for the NLP problem, and thus a fast and well-suited LU factorization method can be used for this solution. On the other hand, typical DAE solvers use the integration of multisteps to compute the state variables in each time interval. This integration normally costs additional time if the number of steps is increased. In addition, the variable sensitivities will be also computed using a multisteps method if the typical DAE solver is used. This method uses the derivatives of the integrated steps and the chain-rule to these derivatives, which also increase the time expense. Accordingly, more efficiency can be added to the multiple shooting method. On the other hand, in comparison to the collocation method, by using multiple shooting in the proposed approach the size of the resulted NLP problem is much smaller. This is because the size of the NLP problem in multiple shooting is only composed of the parameterized

states and controls at the time nodes, while if the collocation method is used the state and control variables at the collocation points in each finite elements are also considered as the variables of the NLP problem.

Since in multiple shooting the computation of each shoot is independent, this makes a parallel computation possible. Therefore, the increase of the number of shoots (i.e. finite elements) will not lead to more computation expense if a parallel computation is implemented.

For NMPC analysis, we proposed a new approach to ensure the stabilization of finite NMPC systems. This approach is proposed by reformulating the optimal control problem within which the optimal control problem is extended. We introduced auxiliary linear states to the optimal control problem, which enforce the system states to be constructed by adding inequality constraints. Thus the stability properties of system states will conform to those of the auxiliary states, i.e. the system states will be stable, if the auxiliary states are stable. The eigenvalues of the linear state equations introduced will be determined to stabilize the auxiliary state variables and at the same time make the optimal control problem feasible. This is achieved by considering the eigenvalues as optimization variables in the optimal control problem. Therefore, the solution of the optimal control problem guarantees the feasibility, stability and optimality of the NMPC system. To penalize the auxiliary state, we introduced an additive integral and quadratic term which is added to the performance index of the optimal control problem. We suggested procedures to select a sufficient solution for this penalty term. The optimal eigenvalues lead to the exponential stability of the linear states which restrict and thus stabilize the system states exponentially in the open-loop case.

Moreover, using this formulation, although the number of the state variables in the setup of the optimal control problem is doubled, the size of the NLP problem produced from this problem will not significantly increase. This is because the auxiliary states introduced are linear, and therefore the additional size of the NLP problem comes only from the size of the eigenvector. In addition, we do not need to introduce additional terminal constraints for stability purposes, since the introduced auxiliary constraints will steer the model states to the equilibrium point. However, the new optimal control problem formulation may cause additional computational expense due to the restriction of the system states.

The features of the system dynamics are analyzed at the stationary point when the new optimal control formulation is used. It is concluded that the stable poles of the auxiliary linear states stabilize the system poles of the linearized system at the stationary point.

According to this formulation, we propose a new RHC algorithm to guarantee the stabilization of the NMPC systems. We analyze the stability of the NMPC system when closed-loop system measurements are applied using this Algorithm. If there are no disturbances, the feedback at every sampling time is not necessary since the prediction is exact. When disturbances are considered and the disturbance enters the system is equal to the disturbance out of the system then the equilibrium point will not be changed. Otherwise, if the disturbance causes an additional amount added to the states then the equilibrium point will be changed. In this situation, to keep the desired equilibrium point, we proposed closed-loop NMPC algorithm to compensate the disturbance. We prove that using the proposed RHC algorithm an asymptotical stability of the closed-loop system can be guaranteed within which the system states will approach the equilibrium point much more quickly than if the typical RHC algorithms are used.

All of the proposed algorithms have been realized in the framework of the numerical

algorithm group (NAG) and the interior point optimizer (IPOPT) in C/C++ environment. Several case studies as application examples using optimal control problems with challenging and highly nonlinear dynamics are presented to show the high performance and reliability of the proposed approach and to prove usefulness of the proposed optimal control formulation for the stability insurance.

Several case studies are compared with some of existing NMPC algorithm, such as multiple shooting code (MUSCOD II) and quasi-sequential algorithm, in terms of the computational expense, integration accuracy and optimization efforts and values. From these results it can be seen that the proposed algorithm is more efficient when a large-scale NMPC problem is needed to be solved. In addition the proposed method for the stability insurance has also shown the effectiveness of the new NMPC formulation.

The proposed algorithm is successfully applied to a challenging optimal control problem with highly nonlinear system dynamics, namely the nonlinear model predictive control of a loading bridge at the Institut für Automatisierungs- und Systemtechnik laboratory at Ilmeau of Technical University.

8.2 Outlook

This dissertation can be extended in several ways

- Numerical analysis can be drawn if the nonlinear optimal control problem is discretized into unequal subintervals or even using different number of collocation points in different subintervals. This way of discretization can be decided according to the stiffness of the state variables as well as the accuracy of the optimal solutions.

- Since the implementation of the parallel computation is possible, an extending result can be obtained with lower computational expenses.

- The proposed algorithms which were realized by NAG and IPOPT under C/C++ environment can be reformulated to be more compatible and easily insertable in a general form of the optimal control problem.

Bibliography

[1] Z. Aktas and H. J. Stetter. A classification and survey of numerical methods for boundary value problems in ordinary differential equations. *International journal for numerical methods in engineering*, 11:771–796, 1977.

[2] M. Alamir and G. Bornard. On the stability of receding horizon control of nonlinear discrete-time systems. *Systems and Control Letters*, 23:291–296, 1994.

[3] Amira GmbH, Duisburg. *Laboratory experiment loading bridge*, June 2001. PS600.

[4] B. Anderson and J. Morre. *Optimal control: linear quadratic methods*. Prentice Hall, 1990.

[5] N. Andreasson, A. Evgrafov, and M. Patriksson. *An introduction to continuous optimization: foundations and fundamental algorithms*. Studentlitteratur AB, 2007.

[6] G. F. Andrews. A multiple-shooting technique for optimal control. *Journal of Optimization Theory and Application*, 102(2):299–313, 1999.

[7] U. Ascher, J. J. Christiansen, and R. D. Russell. Math. comp. *A collocation solver for mixed-order systems of boundary value problems*, 33(146):659–679, 1979.

[8] U. M. Ascher. *Numerical methods for evolutionary differential equations*. SIAM, 2008.

[9] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential algebraic equations*. SIAM,Philadephia, 1998.

[10] W. Auzinger, R. Frank, and H. J. Stetter. Vienna contributions to the development of RK-methods. *Applied Numerical Mathematics*, 22(1-3):35 – 49, 1996. Special Issue Celebrating the Centenary of Runge-Kutta Methods.

[11] W. Auzinger, O. Koch, and E. Weinmüller. Efficient collocation schemes for singular boundary value problems. *Numerical algorithms*, 31:5–25, 2002.

[12] V. Balakrishnan, Z. Zheng, and M. Morari. Constrained stabilization of discrete-time systems. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 205–216, Oxford, UK, 1994. Oxford Univ. Press.

[13] M. Bartholomew Biggs. *Nonlinear optimization with engineering applications*, volume 19 of *Springer Optimization and Its Applications*. Springer, 2008.

[14] R. G. Bartle. *The elements of real analysis*. John Wiley and Sons, New York, 2nd edition, 1976.

[15] R. E. Bellman. *Applied Dynamic Programing*. Princeton Univercity Press, 1962.

[16] B. W. Bequette. Nonlinear control of chemical processes: A review. *Industrial and Engineering Chemistry Research*, 30:1391–1413, 1991.

[17] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific, Belmont, Massachusetts, 1995.

[18] L. T. Biegler. Solution of dynamic optimization problem by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*, 8:243–248, 1984.

[19] L. T. Biegler. Efficient solution of dynamic optimization and NMPC problems. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control*, volume 26, pages 219–245, Basel, 2000. Birkhaeuser.

[20] L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57(4):575–593, 2002.

[21] T. Binder, L. Blank, H.G. Bock, R. Burlisch, W. Dahmen, M. Diehl, W. Marquardt, J.B. Schlöder, and O.-von Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Grötschel, S.O. Krumke, and J. Rambau, editors, *Online optimization of large scale systems*, pages 295–339. Springer, 2001.

[22] R. R. Bitmead, M. Gevers, and V. Wertz. *Adaptive Optimal Control: The thinking man's GPC*. International Series in Systems and Control Engineering. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[23] F. Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced Lyapunov functions. *Automatic Control*, 39(2):428 –433, Feb. 1994.

[24] K. U. Bletzinger. Extended method of moving asymptotes based on second-order information. *Structural and Multidisciplinary Optimization*, 5(3):175, 183 1993.

[25] H. G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K. H. Ebert, P. Beufhard, and W. Jäger, editors, *Modelling of Chemical Reaction System*, Springer Series in Chemical Physics 18, Heidelberg, 1981. Springer.

[26] H. G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatmenat of Inverse Problems in Differential and Integral Equations*, Boston, 1983. Birkhaeser.

[27] H. G. Bock. *Randwetproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*. PhD thesis, University of Bonn, Bonn, 1985.

[28] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress*, pages 243–247, Budapest, 1984. Pergamon Press.

[29] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornüjols, I. E. Grossmann, C. D. Laird, J. Leeand A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. Research Report RC 213771, IBM T. J. Watson Research Center, Yorktown, USA, November 2005.

[30] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*. SIAM, 1996.

[31] R. Brockett. Asymptotic stability and feedback stabilization. in Differential Geometric Control Theory, pages 181–193. Birkhaeuser, 1983.

[32] C. G. Broyden. The convergence of single-rank quasi-newton methods. *Mathematics of Computation*, 24:365–382, 1970.

[33] A. E. Bryson and Y. C. Ho. *Applied optimal control*. Hemisphere publication corporation, Levittown, 1975.

[34] R. L. Burden and J. D. Faires. *Numerical Analysis*. Thomson, 7th edition, 2001.

[35] J. C. Butcher. *Numerical method for ordinary differential equations*. Wiley, 2nd edition, 2008.

[36] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer-Verlag, 1998.

[37] P. J. Campo and M Morari. Robust model predictive control. In *American Control Conference*, pages 1021–1026, 1987.

[38] A. Cervantes and L. T. Beglier. Large-scale DAE optimization using a simultanous NLP formultion. *AIChE*, 44(5):1038–1050, 1998.

[39] A. M. Cervantes, A. Wächter, R. H. Tuetuencue, and L. T. Biegler. A reduced space interior point strategy for optimization of differential algebraic systems. *Computers and Chemical Engineering*, 24(1):39 – 51, 2000.

[40] H. Chen and A. Allgöwer. A computationally attractive nonlinear predictive control scheme with guaranteed stability for stable systems. *Journal of Process Control*, 8(5-6):475–485, 1998.

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

[42] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear predictive control scheme for stable systems: Application to a CSTR. In *Proceedings International Symposium on Advanced Control of Chemical Processes, ADCHEM*, pages 471–476, Bauff, Canada, 1997.

[43] H. Chen, Scherer, C. W., and F. Allgöwer. A robust model predictive control scheme for constrained linear systems. In *Fifth IFAC Symposium on Dynamics and Control of Process Systems, DYCOPS-5*, pages 60–65, Korfu, 1998.

[44] H. Chen, C. W. Scherer, and F. Allgöwer. A game theoretic approach to nonlinear robust receding horizon control of constrained systems. In *Proceedings American Control Conference*, pages 3073–3077, Albuquerque, 1997.

[45] H. S. Chen and M. A. Stadtherr. Enhancements of the Han-Powell method for successive quadratic programming. *Computers and Chemical Engineering*, 8(3-4):229 – 234, 1984.

[46] W. H. Chen, J. O'Reilly, and D. J. Ballance. On the terminal region of model predictive control for non-linear systems with input/state constraints. *Int. J. Adapt. Control Signal Process*, 17:195–207, 2003.

[47] D. Chmielewski and V. Manousiouthakis. On constrained infinite-time linear quadratic optimal control. *Systems and Control Letters*, 29:121–129, 1996.

[48] F. J. Christophersen. *Optimal control of constrained piecewise affine systems*. Lecture Notes in Control and Information Sciences 359. Springer, 2007.

[49] D. W. Clarke. Application of generralized predictive control to industrial processes. *IEEE Control Systems Magazine*, 122:49–55, 1988.

[50] J. E. Cuthrell. *On the optimization of differential-algebraic systems of equations in chemical engineering*. PhD thesis, Carnegie-Mellon University, Pittsburgh, 1986.

[51] J. E. Cuthrell and L. T. Biegler. Simultaneous optimization and solution methods for batch reactor control profiles. *Computer Chem. Eng.*, 13(1/2):49–62, 1989.

[52] J.E. Cuthrell and L.T. Biegler. On the optimization of differential algebraic process systems. *AIChE Journal*, 33:1257–1270, 1987.

[53] C. De Boor and B. Swartz. Collocation at gaussian points. *SIAM J. Numer. Anal.*, 10(4):582–606, 1973.

[54] C. DeBoor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[55] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[56] G. DeNicolao, L. Magni, and R. Scattolini. On the robustness of receding horizon control with terminal constraints. *IEEE Transactions on Automatic Control*, 43(3):451–453, 1996.

[57] G. DeNicolao, L. Magni, and R. Scattolini. Stability and robustness of nonlinear receding horizon control. In F. Allgöwer and A. Zheng, editors, *Proceedings of International Symposium on Nonlinear Model Predictive Control: Assessment and Future Directions*, pages 77–90, 1998.

[58] G. DeNicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear time-varying systems. *IEEE Transactions on Automatic Control*, 43(3):1030–1037, 1998.

[59] C. Desoer and M. Vidyasager. *Feedback Systems: Input-Output Properties (classical in applied mathmatics)*. SIAM, 2009.

[60] M. Diehl. *Real-time optimization for large scale nonlinear processes*. PhD thesis, University of Heidelbrg, 2001.

[61] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[62] M. Diehl, H.G. Bock, H. Diedam, and P.-B. Wieber. Fast direct multiple shooting algorithms for optimal robot control. In K. Mombaur, editor, *Fast Motions in Biomechanics and Robotics*, volume 340 of *Lecture Notes in Control and Information Sciences*, pages 65–94. Springer Berlin Heidelberg, 2006.

[63] M. Diehl, H.G. Bock, and J.P. Schloder. Real-time iterations for nonlinear optimal feedback control. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 5871 – 5876, 12-15 2005.

[64] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H. Bock, T. Buerner, E. Gilles, A. Kienle, J. Schlöder, and E. Stein. Real-time optimization of large scale process models: nonlinear model predictive control of a high purity distillation column. In M. Grötschel, S. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*. Springer, Berlin, 2001.

[65] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, H.G. Allgöwer F. Bock, E.D. Gilles, and J.P. Schlöder. An efficient algorithm for nonlinear model predictive control of large-scale systems. Part I: Description of the method. *at Automatisierungstechnik*, 50(12):557–567, 2002.

[66] M. Diehl, A. Schäfer, and D. B. Leineweber. MUSCOD II user manual. Technical report, Interdisciplinary center for scientific computing (IWR),University of Heidelberg, 2001.

[67] P. Dorato, C. A. Abdallah, and V. Cerone. *Linear Quadratic Control: An Introduction*. Printce Hall, 1995.

[68] P. Dua, K. Kouramas, V. Dua, and E. N. Pistikopoulos. MPC on a chip-recent advances on the application of multi-parametric model-based control. *Computers and Chemical Engineering*, 32:754–765, 2008.

[69] S. Engell, S. Kowalewski, and B.H. Krogh. Discrete events and hybrid systems in process control. In *Proceedings of the 5th international conference on chemical process control*, volume 93 of *AIChE Symposium*, 1997.

[70] B. C. Fabian. A java application for the solution of optimal control problems. Technical report, Stevens Way, Box 352600 Seattle, WA 98195, USA, 1998.

[71] E. Fehlberg. Klassische Runge-Kutta Formeln fünfter und siebter Ordnung mit Schrittweitenkontrolle. *Computing*, 4:93–106, 1969.

[72] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[73] H.J. Ferreau, G. Lorini, and M. Diehl. Fast nonlinear model predictive control of gasoline engines. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pages 2754 –2759, 4-6 2006.

[74] R. Findeisen and F. Allgöwer. Nonlinear model predictive control for index-one. volume 26, pages 145–161. Birkhauser, 2000.

[75] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, pages 1–23, Veldhoven, 2002.

[76] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss. State and output feedback nonlinear model predictive control: An overview. *Eur. J. Control*, 9(2-3):190–206, 2003.

[77] B. A. Finlayson. *The method of weighted residuals and variational principles.* Acadimic Press, 5th edition, 1972.

[78] B. A. Finlayson. *Nonlinear analysis in chemical engineering.* McGraw-Hill, New York, 1980.

[79] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.

[80] R. Frank. The method of iterated defect correction and its application to two-point boundary value problems, Part I. *Numerische Mathematik*, 25:409–419, 1976.

[81] H. Genceli and N. Nikolaou. Robust stability snalysis of constrained: 1-norm model predictive control. *AIChE Journal*, 39(12):1954–1965, 1993.

[82] P. E. Gill, E. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM J. Optim.*, 12(1):979–1006, 2002.

[83] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*, 47:99–131, 2005.

[84] P. E. Gill, W. Murray, and M. A. Saunders. User guide for SQOPT version 7: Software for large-scale linear and quadratic programming. Technical report, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2006.

[85] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.

[86] G. H. Golub and C. F. Van-Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[87] N. Goto and H. Kawable. Direct optimization methods applied to a nonlinear optimal control problem. *Mathematics and Computers in Simulation*, 51(6):557 – 577, 2000.

[88] Numerical Algorithms Group. NAG C library manual mark 8, 2005.

[89] L. Gruene, D. Nesic, and J. Pannek. Assessment and future directions of nonlinear model predictive control. volume 358 of *Lecture notes in control and information sciences*, pages 105–113. Springer Berlin-Heidelberg, 2007.

[90] M. A. Henson and D. E. Seborg. *Nonlinear Process Control*. Prentice Hall, Upper Saddle River. NJ, 1997.

[91] W. R. Hong, S. Q. Wang, G. Li, P.and Wozny, and L. T. Biegler. A quasi-sequential approach to large-scale dynamic optimization problems. *AIChE Journal*, 52(1):255–268, 2006.

[92] E. Isaacson and H. B. Keller. *Analysis of numerical methods*. Dover Publications, 1996.

[93] H. M. Jaddu. *Numerical methods for solving optimal control problems using chebyshev polynomials*. PhD thesis, Japan Advanced Institute of Science and Technolgy, 1999.

[94] D. Karft. Algorithm 733: TOMP-Fortran modules for optimal control caluclations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994.

[95] S. S. Keerthi and E. G. Gilbert. Optimal infinite-horizon feedback laws for a genral class of constrainted discrete-time systems: Satability and moving-horizon approximations. *Journal of Optimization Theory and Apllication*, 57(2):265–293, 1988.

[96] H. K. Khalil. *Nonlinear Systems*. Macmillan, 1992.

[97] D. E. Kirk. *Optimal Control Thoery*. Prentice Hall Inc, Englewood Cliffs, 1970.

[98] O. Koch and E. Weinmüller. Iterated defect correction for the solution of singular initial value problems. *SIAM Journal on Numerical Analysis*, 38:1784–1799, 2001.

[99] S. L. Kothare. *Model predictive control (MPC) for constrained nonlinear systems*. PhD thesis, California Institute of Technology, Pasadena, CA, Mar 1996.

[100] S. L. Kothare and M. Morari. Contractive model predictive control for constraind nonlinear systems. *IEEE Transaction on Automatic Control*, 45(6):1053–1071, 2000.

[101] D. Kraft. Computational mathematical programming. In K. Schittkowski, editor, *On converting optimal control problems into nonlinear programming problems*, volume F15, pages 261–280. Springer, 1985.

[102] E. B. Lee and L. Markus. *Foundations of optimal control theory*. John Wiley, USA, 1967.

[103] D. B. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblem, The Theory of MUSCOD II in a Nutshell. Master's thesis, University of Heidelberg, 1995.

[104] D. B. Leineweber, I. B., H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: theoretical aspects. *Computers and Chemical Engineering*, 27(2):157 – 166, 2003.

[105] D. B. Leineweber, H. G. Bock, J. P. Schlöder, J. V. Gallitzendörfer, A. Schäfer, and P. Jansohn. A boundary value problem approach to the optimization of chemical processes described by DAE models. Technical report preprint 97-14, IWR, University of Heidelberg, 1997.

[106] D. B. Leineweber, A. S., H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization: Part II: Software aspects and applications. *Computers and Chemical Engineering*, 27(2):167 – 174, 2003.

[107] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[108] M. A. Lelic and P. E. Wellstead. Genarlized pole placement self tuning controller. Part 1. Basic algorithm. *Int. J. Control*, 46(2):547–568, 1987.

[109] F. L. Lewis and V. L. Syrmos. *Optimal Control theory*. Wiley Interscience, 2nd edition, 1995.

[110] S. Li and L. Petzold. Software and algorithms for sensitivity analysis of large-scale differential algebraic systems. *J. Comput. Appl. Math.*, 125(1-2):131–145, 2000.

[111] F. Lin. *Robust control design: An optimal control approach*. John Wiley and Sons, 2007.

[112] D. A. Linkers and M. Mahfonf. Advanced in model-based predictive control. In *chpter Generalized Predictive Control in Clinical Anaesthesia*. Oxford University Press, 1994.

[113] J. Liu, D. M. DelaPena, P. D. Christofides, and J. F. Davis. Lyapunov-based model predictive control of particulate processes subject to asynchronous measurements. *Particle and Particle Systems Characterization*, 25:260–375, 2008.

[114] D.G. Lünberger. *Linear and nonlinear programming*. Addison-Wesley Publishing Company, 2nd, 1989.

[115] J. M. Maciejowski. *Predictive control with constraint*. Prentice Hall, 1st edition, 2002.

[116] L. Magni, G. DeNicolao, L. Magnani, and R. Scattolini. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica*, 37:1351–1362, 2001.

[117] L. Magnia and R. Scattolini. Stabilizing model predictive control of nonlinear continuous time systems. *Annual Reviews in Control*, 28:1–11, 2004.

[118] A. Malmgren and K. Nordström. A contraction property for state feedback design of linear discrete-time systems. *Automatica*, 30(9):1485 – 1489, 1994.

[119] A. Malmgren and K. Nordström. Optimal state feedback control with a prescribed contraction property. *Automatica*, 30(11):1751 – 1756, 1994.

[120] T. Maly and L. R. Petzold. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Appl. Numer. Math.*, 20(1-2):57–79, 1996.

[121] T.E. Marlin and A.N. Hrymak. Real-time operations optimisation of continuous processes. In *Proceedings of the 5th international conference on chemical process control*, volume 93 of *AIChE Symposium*, 1997.

[122] D. Q. Mayne. Control of constrained dynamic systems. *European Journal of Control*, 7:78–99, 2001.

[123] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–825, 1990.

[124] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokärt. Constrained model predictive control: stability and optimality. *Automatica*, 36:789–814, 2000.

[125] E. S. Meadows, M. A. Henson, J. W. Eaton, and J. B. Rawlings. Receding horizon control and discontinuous state feedback stabilization. *International Journal of Control*, 62(5):1217–1229, 1995.

[126] P. Mhaskar, N.H. El-Farra, and P.D. Christofides. Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. In *American Control Conference, 2005. Proceedings of the 2005*, volume 2, pages 828 – 833, 8-10 2005.

[127] H. Michalska and D. Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transaction on Automatic Control*, 38:1623–1632, 1993.

[128] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computer and chemical engineering*, 23(4-5):667–682, 1999.

[129] K. R. Morrison and R. W. H. Sargent. Optimization of multistage processes described by differential-algebraic equations. In *Lecture Notes in Mathematics*, volume 1230 of *Numerical Analysis*, pages 86–102. Springer-Verlag, 1986.

[130] R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[131] Z. Nagy, R. Findeisen, M. Diehl, F. Allgöwer, H. G. Bock, S. Agachi, J. P. Schloder, and D. Leineweber. Real-time feasibility of nonlinear predictive control for large scale processes-a case study. In *American Control Conference*, volume 6, pages 4249 –4253, 2000.

[132] D. S. Naidu. *Optimal control system*. CRC Press, Baco Raton, Fl, USA, 4th edition, 2003.

[133] N. S. Nise. *Control systems engineering*. John Wiley and Sons, 4th edition, 2004.

[134] J. Nocedal, A. Wächter, and R. A. Waltz. Adaptive barrier strategies for nonlinear interior methods. Research Report RC 23563, IBM T. J. Watson Research Center, Yorktown, USA, March 2005.

[135] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operation Research and Financial Engineering. Springer, 2nd edition, 2006.

[136] G. Pannocchia and J. Rawlings. Disturbance models for offset-free model predictive control. *AIChE*, 49:426–437, 2003.

[137] S. Patra and S.K. Goswami. A non-interior point approach to optimum power flow solution. *Electric Power Systems Research*, 74(1):17 – 26, 2005.

[138] E. N. Pistikopoulos, N. A. Bozinis, V. Dua, J. D. Perkins, and V. Sakizlis. Improved process control. 02/097540 A1, World Patent Applications WO, 2002.

[139] K. J. Plitt. Ein konverhents Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerung. Master's thesis, University of Bonn, 1981.

[140] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):1096–1104, 2003.

[141] W.R. Ramirez. *Computational Methods for Process Simulation*. Butterworth-Heinemann, Oxford, 2nd edition, 1997.

[142] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE control system magazine*, 20(7):38–52, 2000.

[143] J. B. Rawlings and K. R. Muske. The stability of constrained receding horizon control. *IEEE Transaction on Automatic Control*, 38(10):1512–1516, 1993.

[144] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29(5):1251–1274, 1993.

[145] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristtic control: Application to industrial processes. *Automatica*, 14(2):413–428, 1978.

[146] S. M. Robert and J. S. Shipman. Two-point boundary problems: shooting methods. *Amirican Elesevier*, 1972.

[147] R. D. Robinett, D. G. Wilson, G. R. Eisler, and J. E. Hurtado. *Applied dynamic programming for optimization of dynamical systems*. Advances in Design and Control. SIAM, Philadelphia, 2005.

[148] R. Ross. Revolutionising model-based predictive control. *IEE Computers and Control Engineering*, page 26, 2004.

[149] R. D. Russell and J. Christiansen. Adaptive mesh selection strategies for solving boundary-value problems. *SIAM J. Numerical Anal.*, 15(1):59–80, 1978.

[150] Y. Saad. *Iterative Methods for Sparse Linear System*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[151] A. Sage. *Optimum systems control*. Printice Hall, 2nd edition, 1977.

[152] S. Sager, U. Brandt-Pollmann, M. Diehl, D. Lebiedz, and H. G. Bock. Exploiting system homogeneities in large scale optimal control problems for speedup of multiple shooting based SQP methods. *Computers and Chemical Engineering*, 31(9):1181 – 1186, 2007.

[153] L. O. Santos and L. T. Biegler. A tool to analyze robust stability for model predictive controllers. *Journal of Process Control*, 9:233–246, 1999.

[154] L. O. Santos, L. T. Biegler, and J. A. Castro. A tool to analyze robust stability for constrained nonlinear MPC. *Journal of Process Control*, 18:383–390, 2008.

[155] R. W. H. Sargent and G. R. Sullivan. The development of an efficient optimal control package. In *Proceedings of the 8th IFIP Conference on Optimization Techniques: Part 2*, volume 7 of *J. Stör*, pages 158–168, Heidelberg, 1978. Springer.

[156] A. Schäfer, P. Kühl, M. Diehl, J. Schlöder, and H. G. Bock. Fast reduced multiple shooting methods for nonlinear model predictive control. *Chemical Engineering and Processing*, 46(11):1200–1214, 2007.

[157] F. Scheid. *Schaum's outline of theory and problems of numerical analysis*. McGRAW Hill, 1968.

[158] C. Schmid and L.T. Biegler. Quadratic programming methods for reduced hessian SQP. *Computers and Chemical Engineering*, 18(9):817 – 832, 1994. An International Journal of Computer Applications in Chemical Engineering.

[159] S. Scholz. On Stetters global error estimation in the smooth phase on stiff differential equations. *Computing*, 36:43–55, 1986.

[160] V. H. Schulz. *Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots*. PhD thesis, University of Heidelberg, 1996.

[161] P. O. M. Scokärt and J. B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1169, 1998.

[162] R. Serban and L. R. Petzold. COOPT – a software package for optimal trol of large-scale differential-algebraic equation systems. *Mathematics and Computers in Simulation*, 56(2):187 – 203, 2001.

[163] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24:647–656, 1970.

[164] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Number 6 in Textbooks in Applied Mathematics. Springer, New York, 2nd edition, 1998.

[165] H.J. Stetter. The defect correction principle and discretization methods. *Numerische Mathematik*, 29:425–443, 1978.

[166] O.von Stryke. Numerical solution of optimal control problems by direct collocation. In Bulirsch et al, editor, *Optimal control: Calculus of variation, optimal control theory and bumerical methods*, volume 129, 1993.

[167] E. Sueli and D. F. Mayers. *An introduction to numerical analysis*. Cambridge University Press, 2003.

[168] M. Sznaier and M.J. Damborg. Heuristically enhanced feedback control of constrained discrete-time linear systems. *Automatica*, 26(3):521 – 532, 1990.

[169] J. Tamimi and P. Li. Nonlinear model predictive control using multiple shooting combined with collocation on finite elements. In *International Symposium on Advanced Control of Chemical Processes*, Istanbul, Turkey, July 2009. IFAC. CD Proceeding.

[170] D. J. Ternet and L. T. Biegler. Interior-point methods for reduced Hessian successive quadratic programming. *Computers and Chemical Engineering*, 23(7):859 – 873, 1999.

[171] G.L. Torres and V.H. Quintana. An interior-point method for nonlinear optimal power flow using voltage rectangular coordinates. *Power Systems, IEEE Transactions on Power Systems*, 13(4):1211 –1218, November 1998.

[172] G.L. Torres and V.H. Quintana. Optimal power flow by a nonlinear complimentary method. *Power Systems, IEEE Transactions on Power Systems*, 15(3):1028 –1033, August 2000.

[173] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, 1997.

[174] S. Vasantharajan, J. Viswanathan, and L. T. Biegler. Reduced successive quadratic programming implementation for large-scale optimization problems with smaller degrees of freedom. *Computers and Chemical Engineering*, 14(8):907 – 915, 1990.

[175] D. Verscheure, B. Demeulenäre, J. Swevers, J. De Schutter, and M. Diehl. Practical time-optimal trajectory planning for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, (Accepted), 2009.

[176] M. Vidyasagar. *Nonlinear systems analysis*. SIAM, 2nd edition, 2002.

[177] J. Villadson and M. L. Michälsen. *Solution of differential equation models by polynomial approximation*. Printice Hall, Ne York, 1978.

[178] A. Wächter. Introduction to IPOPT: A tutorial for downloading, installing and using IPOPT. Technical report, Department of Mathematical Sciences, IBM, IBM T.J. Watson Research Center, Yorktown Heights. NY, 2008.

[179] A. Wächter and L. T. Biegler. Global and local convergence of a reduced space quasi-newton barrier algorithm for large-scale nonlinear programming. CAPD Technical Report B-00-06, Carnegie Mellon Univeristy, August 2000.

[180] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming mathematical programming. Technical report, IBM T. J. Watson Research Center, Yorktown, USA, 2006.

[181] H. Wei, H. Sasaki, J. Kubokawa, and R. Yokoyama. An interior point nonlinear programming for optimal power flow problems with a novel data structure. In *20th International Conference on Power Industry Computer Applications*, pages 134 –141, 11-16 1997.

[182] S. J. Wright. Applying new optimization algorithms to model predictive control. In *Proceedings of the 5th international conference on chemical process control*, volume 93 of *AIChE Symposium*, 1997.

[183] Y.-C. Wu, A. S. Debs, and R. E. Marsten. A direct nonlinear predictor-corrector primal-dual interior point algorithm for optimal power flows. *IEEE Transactions on Power Systems*, 9(2):876 –883, May 1994.

[184] T. H. Yang and E. Polak. Moving horizon control of nonlinear systems with input saturation, disturbances and plant uncertainty. *International Journal of Control*, 85(4):875–903, 1993.

[185] P. E. Zadunaisky. On the estimation of errors propagated in the numerical integration of ODEs. *Numerische Mathematik*, 27:21–39, 1976.

[186] A. Zheng. Stability of model predictive control with time-varying weights. *Computers Chem. Engng.*, 21(12):1389–1393, 1997.

[187] Z. Q. Zheng. *Robust control of systems subject to constraints.* PhD thesis, California Institute of Technology, Pasadena, CA, 1995.

Thesis Statements

1. We propose a new approach to the solution of dynamic optimization problems as they appear in the *nonlinear model predictive control* (NMPC) problems. The approach is based on the principle of direct methods to solve the nonlinear optimal control problem by using a combination of multiple shooting with collocation on finite elements.

2. We use the multiple shooting approach to convert the nonlinear optimal control problem into the nonlinear programming (NLP) problem.

3. Within multiple shooting, the finite time horizon are discertized into subintervals and then the controls in each subinterval are parameterized. At the same time the initial values of the states at the beginning of each subinterval will be parameterized as well.

4. We use collocation on finite elements for the integration of the model equations and the computation of the gradients required. Due this combinations, the proposed control strategy possesses a higher computation efficiency; it requires a smaller amount of computation expense compared with the existing NMPC optimization algorithms.

5. The contributions of the proposed approach to existing multiple shooting approaches can be summarized as:

   a) The main contribution is the employment of collocation on finite elements to calculate state values and its derivatives instead of using a typical ODE solver.

   b) In comparison to the collocation method, by using multiple shooting the size of the resulted NLP problem is much smaller.

   c) Since in multiple shooting the computation of each shoot is independent, this makes a parallel computation possible. Therefore, the increase of the number of shoots (i.e. finite elements) will not lead to more computation expense if a parallel computation is implemented.

6. We propose a new approach to ensure the stability of NMPC systems by introducing auxiliary state variables and corresponding linear state equations. The features of the proposed approach can be summarized as:

   a) System states are enforced to be contracted in a similar way as the auxiliary state variables by adding inequality constraints. Thus the stability properties of system states conform to those of the auxiliary states.

   b) It leads to a new formulation of NMPC problems, in which a new term is added to the performance index to penalize these auxiliary states.

   c) The system states are stable, if the auxiliary states are stable and the problem is feasible.

d) The eigenvalues of the linear state equations introduced are determined to stabilize the auxiliary state variables and at the same time make the optimal control problem feasible. This is achieved by considering the eigenvalues as optimization variables in the optimal control problem. Therefore, the solution of the optimal control problem guarantees the feasibility, stability and optimality of the NMPC system.

7. We analyze the features of this new formulation of NMPC inside the terminal region of the equilibrium point.

8. We prove that if the prediction horizon is chosen such that the system and auxiliary states reach a neighborhood of the equilibrium point, the optimal eigenvector and auxiliary states can shift instable poles of the linearized system to a stable region.

9. The efficiency of the proposed approach is demonstrated through several case studies. The computation time taken to solve these control problems is in the order milisecond. Therefore the NMPC algorithm can be applied to fast systems. In addition it is successfully implemented to control a laboratory loading bridge. Satisfactory control performance of a moving cart with a single pendulum between two points with highly nonlinear dynamics is achieved.