Universität Karlsruhe (TH), Faculty of Informatics
Institute of Computer Science and Engineering
Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Prof. Dr.-Ing. Uwe D. Hanebeck

Osaka University, Graduate School of Engineering
Department of Mechanical Engineering
Machine Control Laboratory (MCL)
Prof. Dr. Toshiyuki Ohtsuka

Diplomarbeit

An Online-Computation Approach to Optimal Finite-Horizon State-Feedback Control of Nonlinear Stochastic Systems

Marc Peter Deisenroth

August 30, 2006

Referent:	Prof. Dr.-Ing. Uwe D. Hanebeck
Koreferent:	Prof. Dr. Toshiyuki Ohtsuka
Betreuer:	Dipl.-Ing. Florian Weißel

Eidesstattliche Erklärung

Hiermit erkläre ich, die vorliegende Diplomarbeit selbstständig angefertigt zu haben. Die verwendeten Quellen sind im Text gekennzeichnet und im Literaturverzeichnis aufgeführt.

Karlsruhe, August 30, 2006

Marc Peter Deisenroth

Preface

This Diplomarbeit has emerged from work at two different laboratories. The theory and development of the algorithms have been established at the Machine Control Laboratory of the Osaka University under Professor Toshiyuki Ohtsuka's supervision. The thesis has been completed at the Intelligent-Sensor-Actuator Laboratory of the Universität Karlsruhe (TH) under Professor Uwe D. Hanebeck's supervision.

I want to thank my adviser Professor Uwe D. Hanebeck from the Universität Karlsruhe (TH) for supporting my study during the last two years. He is responsible for the recovery of my interest in my study, such that I could complete it quickly and successfully. Furthermore, Professor Hanebeck gave me the chance to write this thesis in Japan, which was a very interesting and important experience for my research and my life. Additionally, I am very grateful to my second adviser Professor Toshiyuki Ohtsuka from the Osaka University for the great supervision during the last months, numerous fruitful discussions on my thesis, and every kind of help and support I needed for my stay in Japan. For all the useful questions, hints, and the support I want to express my thanks to Mr. Florian Weißel, who supervised my work from Karlsruhe. Moreover, I thank Dr. Simon Julier from the University College London for very interesting and helpful discussions on the unscented transformation.

For a very nice time in Osaka, I want to thank all members of the Machine Control Laboratory of the Osaka University, who became good friends of mine.

Since the stay in a foreign country with little language skills is very difficult, the help of other people in everyday life problems is very important. Therefore, I want to say thank you, especially to Ms. Takako Ishikawa, Mr. Akarawit Limpyamitr, Mr. Naoki Miyatake, and Professor Hirata and his family for their help and care.

Finally, I want to thank my parents and my sister for supporting me in my study, my ideas, and my life without any doubts.

Karlsruhe, August 2006
Marc Deisenroth

Contents

List of Figures **IX**

List of Tables **XI**

List of Algorithms **XIII**

List of Examples **XV**

List of Review Material **XVII**

Notation **XX**

Abstract **1**

1 Introduction **3**

2 Considered Optimal Control Problem **9**

2.1 Problem Formulation 9

2.1.1 Measure of Quality 10

2.1.2 Objective 10

2.2 Dynamic Programming 11

2.2.1 Introduction and Assumptions 11

2.2.2 Value Function 12

2.2.3 Recursive Calculation of the Minimal Expected Cost-to-Go 12

2.2.4 Properties of Dynamic Programming 13

2.2.5 Computational Effort in Case of Discrete Problems 14

2.2.6 Limits of Dynamic Programming 15

2.3 Related Work 15

3 New Approach to Optimal Control of Nonlinear Noise-Affected Systems 19

3.1 Theoretical Results 20

3.1.1 Approximation of the Value Function 20

3.1.2 Minimum Principle 22

3.1.3 Consideration of Higher-Order Derivatives in the Value Function Approximation 25

3.1.4 Two-Point Boundary-Value Problem 28

3.1.5 Employment of Prior Knowledge 30

3.1.6 Accuracy of the Spline Interpolation 30

3.2 Practical Methods 35

3.2.1 Formulation of the Two-Point Boundary-Value Problem 36

3.2.2 Solution of the Nonlinear Equation System 37

3.2.3 Restriction of the State Space by Means of the Unscented Transformation 41

3.2.4 Determination of the Grid Points for the Value Function Interpolation . . 43

3.2.5 Interpolation of the Value Function by Means of Piecewisely Defined Cubic Splines 44

4 Simulation Results 53

4.1 Considered Systems and Setting 53

4.2 Application of the Minimum Principle 53

4.2.1 Noise Influence 55

4.2.2 Monte-Carlo Simulation 57

4.2.3 Quality of the Initializing Controller 57

4.3 Dynamic Programming Based on Spline Interpolation of the Value Function . . 60

4.3.1 Improvement of the Initial Solution 63

4.4 Replacement of Numerical Integration with the Unscented Transformation . . . 67

4.5 Comparison with Other Solutions 69

4.5.1 Dynamic Programming on an Adaptive Grid with Few Grid Points . . . 70

4.5.2 Dynamic Programming on a Static Grid with Many Grid Points 70

4.5.3 Dynamic Programming on an Adaptive Grid with Many Grid Points . . 71

4.6 Computational Effort 72
4.7 Survey of Applied Methods 74

5 Conclusions and Future Work **75**

5.1 Conclusions 75
5.2 Future Work 76

A Mathematical Definitions **79**

A.1 The $\otimes$-Operator 79
A.2 Multi-Dimensional Taylor Series Expansion 79
A.3 Topological Space 80
A.4 The L^p Spaces 80

B Introduction to Continuation Processes **83**

B.1 Basic Idea 83
B.2 Formal Definition and Construction of Continuation Processes 84

C Linear Quadratic Control **87**

D Unscented Transformation **91**

D.1 Motivation 91
D.2 Basic Idea 91
D.3 Accuracy 93
D.3.1 Mean Value 93
D.3.2 Covariance 94
D.4 Incorporation of Knowledge of Higher-Order Moments 95
D.5 Properties, Limitations, and Extensions 96

E Splines **99**

E.1 n-Dimensional Cubic Spline 99
E.2 Example of the Error Bound of a Scalar Cubic Spline Interpolant 99

F Implementation Details **103**

F.1 Application of the Minimum Principle 104

F.1.1 Continuation Process 104

F.1.2 Minimization 104

F.2 Application of Dynamic Programming Based on Interpolation of the Value Function 105

F.2.1 Mean- and Covariance Prediction 105

F.2.2 Definition of the Grid 106

F.2.3 Spline Interpolation of the Value Function 107

F.3 Dynamic Programming on a Static Grid 109

Bibliography **113**

List of Figures

1.1 Robots in current research applications. 3

1.2 Path planning. The robot's objective is to move across the room to reach its destination. 4

1.3 Path planning and model predictive control, initial steps. 6

1.4 Path planning and model predictive control, obstacle avoidance. 7

1.5 Optimal and MPC trajectories of the path planning example. 7

2.1 Considered stochastic system. 9

3.1 Example for an interpolating function, which is piecewisely defined by means of cubic splines. 32

3.2 $f(x,\gamma) = \gamma\left(\sin\left(\frac{3\pi}{4}x\right)\right) + (1-\gamma)\,x$. 38

3.3 Principle of the unscented transformation. 41

3.4 Possible restriction of the state space. 44

3.5 Scalar example: determination of the set $\mathcal{U}_k^{(i)}$. 45

3.6 Scalar example: computation of $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(3)})]$. 47

3.7 Scalar example: computation of the expected cost-to-go. 47

3.8 Scalar example: interpolation of the cost function with respect to u_k. 47

3.9 Scalar example: interpolation of the value function with respect to x_k. 48

4.1 Considered functions depending on x and a homotopy parameter γ. 54

4.2 Example state and control trajectories for $\sigma = 0$ and $\sigma = 0.1$ for system (4.1). . 56

4.3 Example state and control trajectories for system (4.2). 56

4.4 Monte-Carlo estimates of the value function resulting from the discrete DP algorithm and the proposed algorithm for system (4.1). 59

4.5 Monte-Carlo estimates of the value function resulting from the DP algorithm on a static grid and the proposed algorithm for system (4.2). 60

4.6 Restriction of the state space for both considered systems. 61

4.7 Spline interpolation of the value function J_k in the area of interest for $\hat{x}_0 = -1$ and $\sigma = 0.2$. 62

4.8 Average state and control trajectories of system (4.1) for the initializing solution and the proposed additional improvement for $\sigma = 0.2$ and $\hat{x}_0 = 1$. 63

4.9 Average state and control trajectories of system (4.2) for the initial solution and the proposed improvement for $\sigma = 5$ and $\hat{x}_0 = 10$. 64

4.10 Monte-Carlo estimates of the value functions $J^{init}_{\sigma=i}$ and $J^{spline}_{\sigma=i}$ of system (4.1) for increasing noise influence. 65

4.11 Monte-Carlo estimates of the value functions $J^{init}_{\sigma=i}$ and $J^{spline}_{\sigma=i}$ of system (4.2). . . 66

4.12 Comparison of the UT and numerical integration. 67

4.13 Monte-Carlo estimates of the value functions of system (4.1) to show the difficulties with the employment of the scaled UT. 68

4.14 Monte-Carlo estimates of the value functions of system (4.1) of two controllers employing the initial algorithm and the improved algorithm with the *un*scaled UT. 69

4.15 Comparison of the Monte-Carlo estimates $J^{MC\_spline}_{\sigma=5}$ and $J^{MC\_DP\ 11}_{\sigma=5}$ for system (4.2). 70

4.16 Comparison of the new controller with a controller, which employs a static grid based DP algorithm, for system (4.1). 71

4.17 Comparison of the new controller with a controller, which employs a static grid based DP approach, for system (4.2). 72

4.18 Comparison of the Monte-Carlo estimates of the value function for system (4.2) with $\sigma = 3,5$ for the new algorithm and an algorithm, which is based on DP on an adaptive grid with many grid points. 73

D.1 Principle of the unscented transformation. 92

F.1 Grid area around the predicted mean values of the successor states of $\hat{x}_0 = -1$ for $\sigma = 0.1$ 107

F.2 Extended and initial grid area around the predicted mean values of the successor states of $\hat{x}_0 = -1$ in system (4.1) for $\sigma = 0.1$. 108

F.3 Transition tensor for system (4.1) with $\sigma = 0.1$ for a fixed control variable. . . . 110

F.4 Discretized Gaussian in case of system (4.1) for $\sigma = 0.1$. 111

F.5 Discretized transition density for a fixed control variable. 111

F.6 Minimal expected cost-to-go for all time steps of the decision-making horizon. . 112

List of Tables

3.1 Values for ε_{mr} for the general error bound of the spline approximation. 32

3.2 Remaining parameters for the general error bound of the spline approximation. . 32

4.1 Discretization intervals and distance of the grid points of the state and control space in case of system (4.1) for different noise influences. 58

4.2 Discretization intervals and distance of the grid points of the state and control space in case of system (4.2) for different noise influences. 58

4.3 Relative execution time for different algorithms. 73

List of Algorithms

1 Dynamic programming 13

2 MPC: Application of the minimum principle 39

3 General Newton method 40

4 Update of the state-feedback control 46

5 Unscented transformation 92

6 System simulation 103

7 Implemented quasi-Newton method 105

8 Dynamic programming based on interpolation of the value function on an adaptive grid 109

List of Examples

Example 3.1: Homotopy between a linear function and a nonlinear function 38

Example 3.2: Possible restriction of the state space 43

Example D.1: Influence of the knowledge of derivatives and moments in the UT for $n = 2$. 94

List of Review Material

Problems caused by noise affecting nonlinear systems 14

Objective 19

Cubic splines 31

Basic idea of the unscented transformation 41

Probability density of the successor state 48

Nonlinear transformation of random variables — expectation value 50

Scaled unscented transformation 67

Notation

t	scalar variable
$\underline{t}$	vector-valued variable (column vector)
$\underline{t}^{\mathrm{T}}$	transposed vector-valued variable (row vector)
$\hat{\underline{t}}$	known value $\underline{t}$
$\boldsymbol{t}$	scalar random variable
$\underline{\boldsymbol{t}}$	vector-valued random variable
$\overline{\underline{t}}$	mean value of the vector-valued random variable $\underline{\boldsymbol{t}}$
$\mathbf{T}$	matrix
$\mathbb{T}$	tensor (multi-dimensional matrix)
f	function, $f : \mathbb{R}^m \to \mathbb{R}$, $m \geq 1$
$\underline{f}$	function, $\underline{f} : \mathbb{R}^m \to \mathbb{R}^n$, m, $n \geq 1$
$\frac{\partial \underline{f}}{\partial \underline{x}}$	partial derivative of $\underline{f}$ with respect to $\underline{x}$
$\frac{\mathrm{d}\underline{f}}{\mathrm{d}\underline{x}}$	total derivative of $\underline{f}$ with respect to $\underline{x}$
$\mathrm{P}(\,\cdot\,)$	probability density function
$\mathcal{N}(\underline{x}; \overline{\underline{x}}, \mathbf{\Sigma}_x)$	Gauss function with mean $\overline{\underline{x}}$ and covariance $\mathbf{\Sigma}_x$
$\mathcal{C}^m$	class of m times continuously differentiable functions
tr	trace operator
E	expectation value operator
$*$	convolution operator
■	end of example
□	end of proof

Abstract

In this thesis, an online-computation approach to optimal finite-horizon state-feedback control of nonlinear stochastic systems is presented. In the considered discrete-time case, the state space and the control space are continuous-valued sets. For nonlinear, noise-affected systems, exact analytic solutions to the optimal control problem do not exist in general. Therefore, appropriate approximations are inevitable.

A two-step algorithm is proposed in this thesis. In the first step, an optimal solution to a simplified problem is derived. In the second step of the algorithm, this result is employed as prior knowledge to derive an improved solution to the original optimal control problem. In both steps, dynamic programming is employed.

With the employment of dynamic programming, the optimal control problem is reformulated as a minimization problem. In the first step of the proposed algorithm, the value function of the stochastic dynamic programming algorithm is approximated by means of Taylor series expansion up to second-order derivatives, which results in a simplification of the problem. The approximation serves as a basis for the derivation of a stochastic minimum principle for the discrete-time case, where the properties of a stochastic Hamilton function are employed. With these theoretical results, the minimization problem is reformulated as a two-point boundary-value problem. The arising nonlinear equation system is solved numerically by means of a continuation process, which yields an optimal control sequence to the simplified problem.

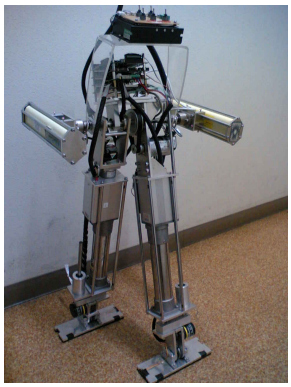
In the second step of the algorithm, the control sequence solving the two-point boundary-value problem is employed. The current system state is propagated through the system equation by means of the unscented transformation, which approximately yields sequences of expectation values and covariances of the successor states. Depending on these means and covariances, an adaptive grid of few points is defined, which changes each time step to incoporate current knowledge. On this grid, the stochastic dynamic programming algorithm can be employed to obtain an improved state-feedback control for the current time step. Instead of performing dynamic programming directly on this grid, the value function is piecewisely approximated by means of cubic splines. Hence, an approximate solution to the nonlinear optimal control problem for continuous-valued control variables as well as for a continuous-valued domain of the system state is obtained.

The proposed online-computation algorithm is analyzed by means of scalar example systems, where the technically important model predictive control is implemented.

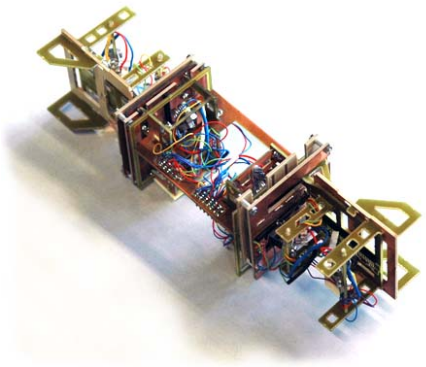
CHAPTER 1

Introduction

When considering control problems of practical engineering applications, nonlinear control theory is the basis to obtain optimal results. To control a system optimally means to influence its behavior, such that a desired goal is achieved in an optimal way. Current research treats the problem of controlling the robots depicted in Figure 1.1 in an optimal way.



(a) Walking machine HW-II, Machine Control Laboratory, Osaka University.



(b) Prototype of a 6 DOF robot, Intelligent Sensor-Actuator-Systems Laboratory, Universität Karlsruhe (TH).

Figure 1.1: Robots in current research applications.

A standard application in robotics is path planning as illustrated by Figure 1.2. The aim is that the robot in the lower left corner of the room moves across the room to reach the destination in the upper right corner. Several points have to be borne in mind to achieve the desired result.

Cost Function. In many cases, it is desirable that the robot reaches the destination efficiently, that is, the objective is a movement along the shortest path and the avoidance of collisions. Hence, a cost function is introduced to incorporate these constraints. In the considered path planning example, the cost function assigns a certain value to any state-input combination. To reach the destination optimally means to minimize this cost function, which is a main aspect

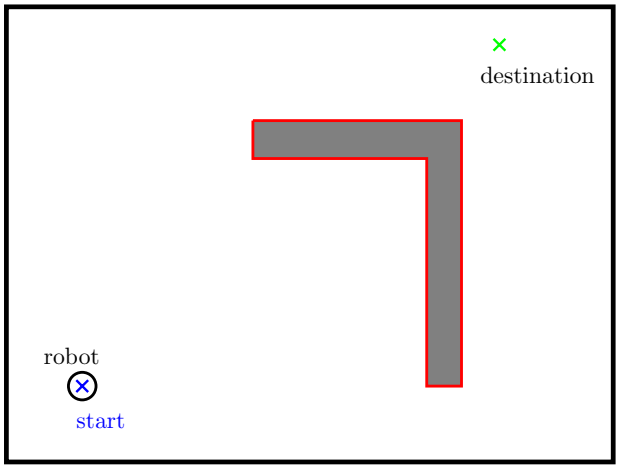


Figure 1.2: Path planning. The robot's objective is to move across the room to reach its destination.

of the desired controller. The incorporation of additional of constraints in the cost function is possible.

Nonlinearities. Most realistic systems have to be described with nonlinear system models. For example, motions of the depicted robots are described involving trigonometric functions. Even if these functions are well known, nonlinear controllers are not easy to design. Unfortunately, nonlinear systems have some special difficulties compared to linear systems. A main point is that, in contrast to linear optimization, an analytical solution to the nonlinear optimization problem cannot be obtained in general. Therefore, the minimization of the cost function is a challenging task, especially in case of nonlinear systems.

Since linear controllers are well known, an obvious approach to nonlinear optimal control is the linearization of the system function around the mean of the current system state and subsequent application of a linear controller. Compared to nonlinear controllers, linear controllers are easy to design. In case of a quadratic cost function, the linear quadratic controller (LQ controller), which employs measurements of the state (state-feedback), provides an optimal linear solution to linear control problems [ÅW97]. When the level of nonlinearity of the considered system increases, the quality of the solutions provided by the linearized controller decays. Therefore, alternative approaches to linearization have to be employed to improve the quality of the desired controller, even if no tool or methodology in nonlinear systems analysis is universally applicable [Vid02]. Therefore, for each controller, a tradeoff between simplicity and exactness has to be found.

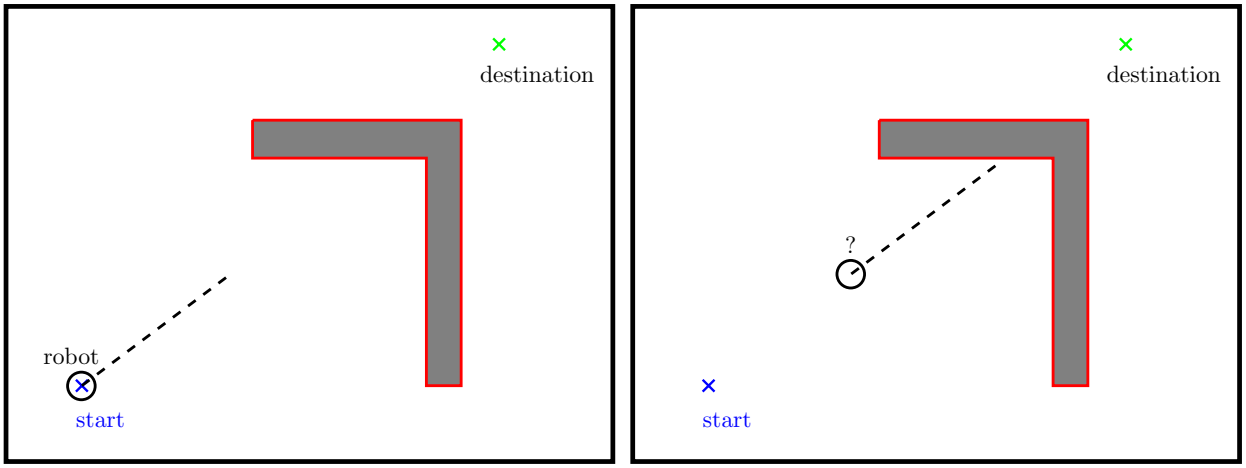
Noise Influence. Besides the nonlinearity of dynamic systems, noise is another issue to be considered to obtain a more realistic model of the true system. Usually, noise can be regarded as a random process with a given distribution. A common assumption is that white noise influences the system, when control problems for noise-affected systems are considered.

In case of a deterministic system, the successor state of the current state is exactly known, that is, this specific successor state will be attained with probability one. When a random process affects the system, it is not possible to predict the next state with certainty. However, depending on the distribution of the noise, it is possible to predict a range of possible successor states by means of the underlying system model. In case of the path planning example, inaccuracies of the measurement of the current robot position or incidences like friction or slip can be considered as noise disturbances. Moreover, a common assumption is that the environment of the robot is not known exactly. For example, the size of the room and the current position are known, but possible obstacles inside the room are unknown. Another possible reason to consider noise, is the simplification of a given system model. In case of a very complex system model, several parts of the system can be modeled as noise, such that the whole model is simplified. Systems suffering from noise processes are called *stochastic* or *probabilistic*. Since the state of a probabilistic system may deviate significantly from the deterministic one, that is, the system state without noise disturbances, the consideration of noise is desired when designing controllers for realistic systems. However, this additional aspect causes serious problems in calculations.

Decision-Making Horizon. When talking about optimal controllers, the decision-making horizon, that is, the time interval, which is considered to determine the optimal control, is a further point to be considered. On the one hand, the problem can theoretically be captured completely in case of the consideration of a long, maybe infinite, horizon. On the other hand, the analysis to derive an optimal solution becomes difficult. This is due to the fact that exact calculations may depend on the convergence of iterations, which cannot be guaranteed in general. In case of infinite-horizon problems, the behavior of the system over a long time interval and the additional information, which is needed to be precise, have to be considered. Thus, the model of the whole system gets more and more complex and results in a computational effort, which is not feasible anymore. Therefore, assumptions have to be made, which cannot be justified in general, for example, a discounting factor of future cost.

A possible solution to the described problem is provided by *model predictive control* (MPC), also known as *receding horizon control* or *moving horizon control*. MPC is often applied to real systems to avoid computational difficulties. Instead of the consideration of an infinite horizon, in MPC the optimal control problem is restricted to be solved for a finite horizon $[k, k+N]$, where the current time step is denoted by k. With this approach, the problem of the determination of closed-loop solutions is circumvented, since the optimal control problem is solved only for the current system state. The first part of the optimal state-feedback control is applied open-loop to the system for one time step. Since the optimal control problem has to be solved only for the current system state and not for all system states simultaneously, the solution is much easier to obtain [Fin04].

In case of the path planning example, where the robot has to reach the destination efficiently, the MPC steps are illustrated in Figures 1.3 and 1.4. In Figure 1.3(a), the initial robot position is shown, where the current decision-making horizon of N time steps is indicated by the dashed



(a) Initial position of the robot. The optimal path within the decision-making horizon is indicated by the black, dashed line.

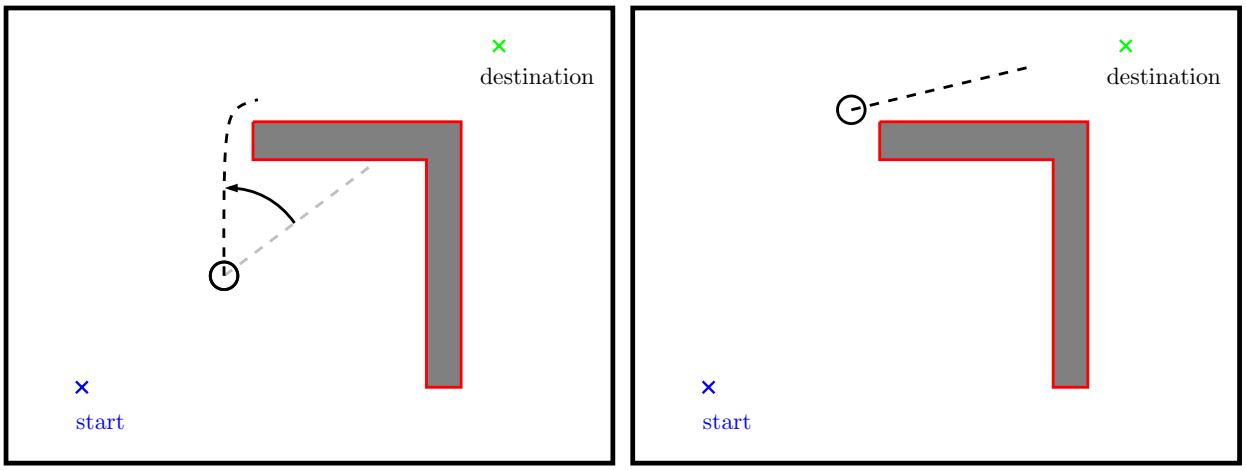
(b) Detection of an obstacle. To avoid a collision, the robot has to change its orientation and to determine a new path.

Figure 1.3: Path planning and model predictive control, initial steps. The robot moves toward the destination, while minimizing the cost function within the decision-making horizon.

line. Possible obstacles inside the room are not known. The robot determines a path minimizing the distance to the destination without hitting the obstacle, which is not considered in the current decision-making horizon. When the optimal state-feedback control for the current system state is determined and applied to the real system, the finite-horizon window is shifted by one time step to $[k+1, k+1+N]$, such that after the next sampling an appropriate control is again determined for the full N-step horizon. After several time steps, the obstacle is in reach, and the robot realized that it has to adjust the intended path as shown in Figure 1.3(b). The adjustment of the orientation of the robot and the resulting new path is depicted in Figure 1.4(a), where the robot plans to move aside the obstacle without collision. After passing the obstacle, the robot again adjusts its orientation and moves directly toward the destination as is depicted in Figure 1.4(b).

The path planning example illustrates that MPC provides only a suboptimal solution, since the robot does not necessarily move along the true optimal trajectory. This is caused by the fact that the obstacle cannot be incorporated into the control decision from the beginning. The comparison of a possibly optimal path and the path provided by MPC in case of the considered robot example is shown in Figure 1.5. The optimal path (blue, dotted) is shorter than the MPC path (black, dashed), due to the triangular inequality. Although MPC provides only suboptimal solutions, the computational effort and the complexity of the system model are reduced significantly in case of MPC compared to the infinite-horizon controller. Hence, MPC is an attractive approach and, therefore, often employed in technical applications.

Reflecting the properties and problems arising from the nonlinearity of the system, the noise influence, and the decision-making horizon, the design of an optimal state-feedback controller



(a) Change of the orientation and the intended path to avoid the collision. The robot moves aside the obstacle.

(b) Overcoming the obstacle and proceeding toward the desired destination.

Figure 1.4: Path planning and model predictive control, obstacle avoidance. After the identification of the obstacle, the intended path is changed to avoid collisions.

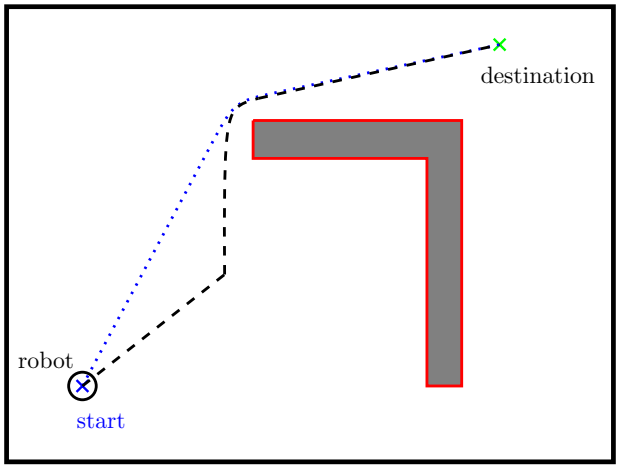


Figure 1.5: Optimal and MPC trajectories of the path planning example. Due to the finite horizon, MPC is not able to obtain the optimal solution (blue, dotted) in general. Therefore, a suboptimal solution (black, dashed) is obtained.

for nonlinear stochastic systems is both interesting and difficult and, therefore, worth to deal with.

In this thesis, an approach to optimal finite-horizon control of nonlinear, stochastic, discrete-time systems is provided. Starting from the original dynamic programming equation, the value function is approximated by means of Taylor series expansion, such that a minimum principle can be applied. Then, a candidate for the optimal control sequence for a finite horizon is obtained. These state-feedback controls are employed as prior knowledge to a subsequent algorithm, which determines an adaptive grid of few points, on which stochastic dynamic programming is performed. Piecewise cubic spline interpolation of the value function yields an approximate solution to the continuous-valued optimal control problem, in the control variable as well as in the state variable.

The remainder of this thesis is structured as follows. In Chapter 2, the considered system is introduced. Moreover, problems originating from nonlinear optimization are discussed. A possible solution method, that is, dynamic programming, is presented, which is the basic tool in this thesis. The chapter is concluded with an overview of related work. In Chapter 3, the idea of the proposed new approach to optimal control of nonlinear noise-affected systems is treated in theoretical as well as in practical methods. Simulation results by means of scalar example systems are presented in Chapter 4 to analyze the properties of the proposed algorithms. Finally, in Chapter 5, the contents of this thesis is summarized, and a survey of future work is given.

CHAPTER 2

Considered Optimal Control Problem

2.1 Problem Formulation

Let the considered discrete-time system be given by

$$\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{\boldsymbol{x}}_k, \underline{u}_k) + \underline{\boldsymbol{w}}_k \ , \quad k = 0, \ldots, N-1 \ , \tag{2.1}$$

where $\underline{x}_k \in \mathbb{R}^n$ denotes the system state at time step k, $\underline{u}_k \in \mathbb{R}^m$ the control variable, and $\underline{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ a nonlinear function. $\underline{\boldsymbol{w}}_k \in \mathbb{R}^n$ is an independent additive noise term with mean value zero and covariance $\boldsymbol{\Sigma}_w$. In the considered finite-horizon case, the terminal time is denoted by N. The initial state $\hat{\underline{x}}_0$ is assumed to be known. Moreover, the state is directly accessible after each time step, that is, the probability density function of the state can be interpreted as a Dirac function. Therefore, it is not necessary to estimate the state. The random noise $\underline{\boldsymbol{w}}_k$ is independent of prior disturbances $\underline{\boldsymbol{w}}_{k-1}, \ldots, \underline{\boldsymbol{w}}_0$ and characterized by a well-defined probability density function P_k^w. Since the density of the noise is assumed to be time-invariant, the time index k is omitted in the following. The considered system is depicted in Figure 2.1.

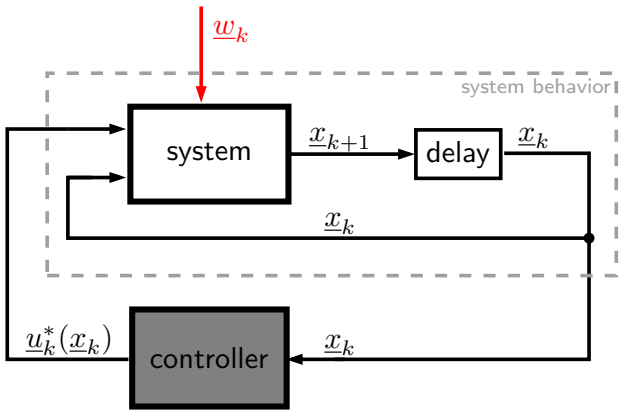


Figure 2.1: Considered stochastic system. The system state is directly accessible after each time step. The controller determines optimal state-feedback controls.

The system state satisfies the Markov property, that is, the state $\underline{x}_{k+1}$ depends only on its direct predecessor $\underline{x}_k$, the control variable $\underline{u}_k$, and the noise term $\underline{\boldsymbol{w}}_k$ at time step k. Therefore,

future states depend only on events of one time step before, and

$$\mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\ldots,\underline{x}_0,\underline{u}_k,\ldots,\underline{u}_0,\underline{w}_k,\ldots,\underline{w}_0) = \mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k,\underline{w}_k) \ . \tag{2.2}$$

Definition 2.1 (Control law) A *control law* is a mapping

$$\underline{\mu}_k : \begin{cases} \mathbb{R}^n & \to & \mathbb{R}^m \\ \underline{x}_k & \mapsto & \underline{\mu}_k(\underline{x}_k) = \underline{u}_k \end{cases}$$

that maps states $\underline{x}_k \in \mathbb{R}^n$ onto controls $\underline{u}_k \in \mathbb{R}^m$.

For an initial state $\hat{\underline{x}}_0$ and a policy[1] $\pi := (\underline{\mu}_0,\ldots,\underline{\mu}_{N-1})$, the states $\underline{x}_k$ can be described as random variables $\underline{\boldsymbol{x}}_k$, $k = 1,\ldots,N$, with distributions defined through the system equation (2.1), that is,

$$\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{\boldsymbol{x}}_k, \underline{\mu}_k(\underline{x}_k)) + \underline{\boldsymbol{w}}_k \ , \ k = 0,\ldots,N-1 \ .$$

2.1.1 Measure of Quality

To define optimality, a measure of quality is required. Therefore, a cost function[2] to be minimized is introduced. In case of stochastic systems, an obvious approach is to define the cost function at a specific time step as the *expected cost-to-go* up to the terminal time N. Therefore, the cost function of the control sequence π starting at $\hat{\underline{x}}_0$ is given by

$$V_\pi(\hat{\underline{x}}_0) := \mathop{\mathrm{E}}_{\underline{w}_0,\ldots,\underline{w}_{N-1}} \left[\tilde{g}_N(\underline{\boldsymbol{x}}_N) + \sum_{k=0}^{N-1} \tilde{g}_k(\underline{\boldsymbol{x}}_k, \underline{\mu}_k(\underline{x}_k), \underline{\boldsymbol{w}}_k) \right] \tag{2.3}$$

for known functions $\tilde{g}_k$, $k = 0,\ldots,N-1$. The function $\tilde{g}_k(\underline{x}_k,\underline{u}_k,\underline{w}_k)$ describes the transition cost from time step k to $k+1$ depending on the system state $\underline{x}_k$, the applied control action $\underline{u}_k$, and the noise term $\underline{\boldsymbol{w}}_k$. The function $\tilde{g}_N$ only depends on the terminal state and represents the terminal cost.

2.1.2 Objective

The objective is to establish a sequence of *optimal* control laws

$$\underline{\mu}_k^* : \begin{cases} \mathbb{R}^n & \to & \mathbb{R}^m \\ \underline{x}_k & \mapsto & \underline{\mu}_k^*(\underline{x}_k) = \underline{u}_k^* \end{cases} \ , \ k = 0,\ldots,N-1$$

mapping the states $\underline{x}_k$ onto optimal state-feedback controls $\underline{u}_k^* = \underline{\mu}_k^*(\underline{x}_k)$, such that the sequence $(\underline{u}_0^*,\ldots,\underline{u}_{N-1}^*)$ minimizes the cost function (2.3). This means, an optimal policy $\pi^* := (\underline{\mu}_0^*,\ldots,\underline{\mu}_{N-1}^*)$ is desired with

$$J(\hat{\underline{x}}_0) := V_{\pi^*}(\hat{\underline{x}}_0) = \min_\pi \left(\mathop{\mathrm{E}}_{\underline{w}_0,\ldots,\underline{w}_{N-1}} \left[\tilde{g}_N(\underline{\boldsymbol{x}}_N) + \sum_{k=0}^{N-1} \tilde{g}_k(\underline{\boldsymbol{x}}_k, \underline{\mu}_k(\underline{x}_k), \underline{\boldsymbol{w}}_k) \right] \right) \ . \tag{2.4}$$

1 sequence of control laws
2 also known as performance index

Definition 2.2 (Value function) The function J in (2.4), that is, the minimal expected cost-to-go, is called *minimal cost function* or *value function*.

2.2 Dynamic Programming

When considering (2.4), a seemingly obvious solution to the minimization problem is to calculate the corresponding cost functions (2.3) for all possible combinations of control laws $\underline{\mu}_k$, $k = 0, \ldots, N-1$, and then to determine the minimizing policy π^*. Since this approach is computationally infeasible, a more sophisticated procedure is desired, for instance, the dynamic programming algorithm, which is introduced in the following.

2.2.1 Introduction and Assumptions

In most cases, nonlinear optimization is based on search algorithms or on the employment of the Karush-Kuhn-Tucker conditions for optimality [SS02]. In contrast to these methods, dynamic programming is an optimization algorithm, which theoretically determines the exact solution to the optimal control problem for finite-horizon Markov decision processes.

Remark 2.1 To apply dynamic programming, the system state has to satisfy the Markov property (2.2). Furthermore, the cost function to be minimized has to be additive over time.

In the considered case of finite-horizon optimal state-feedback control, the objective is to minimize the cost function (2.3) and to determine the minimizing control variables for the entire decision-making horizon. Instead of the consideration of all possible combinations of system states and control decisions for all time steps, dynamic programming recursively determines the solution to appropriate subproblems. Although the use of dynamic programming in nonlinear optimization is limited to some fairly straightforward applications, its use in these is both valuable and instructive [Ber00a]. A detailed introduction to dynamic programming and its applications is given in [Ber87, Ber00a, Ber00b, Put05].

Theorem 2.1 *Let $\pi^* = (\underline{\mu}_0^*, \ldots, \underline{\mu}_{N-1}^*)$ be a sequence of optimal control laws for the minimization problem (2.4). The subproblem to be considered is that the current state is given by $\underline{x}_i$, and the expected cost-to-go from time step i up to time step N*

$$V_{\pi_i}(\underline{x}_i) = \mathop{\mathrm{E}}_{\underline{w}_i, \ldots, \underline{w}_{N-1}} \left[\tilde{g}_N(\underline{\boldsymbol{x}}_N) + \sum_{k=i}^{N-1} \tilde{g}_k(\underline{\boldsymbol{x}}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k) \right]$$

shall be minimized. Then, the truncated sequence

$$\pi_i^* = (\underline{\mu}_i^*, \ldots, \underline{\mu}_{N-1}^*)$$

is optimal for this subproblem [Ber00a].

This property is called principle of optimality*.*

Proof. A proof of the principle of optimality is given in [Put05]. □

2.2.2 Value Function

Theorem 2.1 reveals the principle of optimality, the basic idea of dynamic programming. The dynamic programming algorithm starts the optimization procedure at the terminal time N and proceeds backward in time to the starting time 0. The recursively defined value function at time step k depends on the result of the previous calculation and is defined as

$$J_k(\underline{x}_k) := \min_{\underline{u}_k} \left(\mathop{\mathrm{E}}_{\underline{\boldsymbol{w}}_k} \left[\tilde{g}_k(\underline{x}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k) + J_{k+1}(\underline{\boldsymbol{x}}_{k+1}) \right] \right) \ , \tag{2.5}$$

where the terminal cost incurring at time step N is denoted by

$$J_N(\underline{x}_N) = \tilde{g}_N(\underline{x}_N) \ .$$

The term $J_{k+1}(\underline{x}_{k+1})$ summarizes the minimal expected cost up to the terminal state $\underline{x}_N$ starting from a fixed state $\underline{x}_{k+1}$. When the algorithm calculates the cost for the currently considered state $\underline{x}_k$, the state $\underline{x}_{k+1}$ is not exactly known anymore. Therefore, the state $\underline{x}_{k+1}$ has to be considered as a random variable $\underline{\boldsymbol{x}}_{k+1}$, which is obtained by a one-step prediction starting from $\underline{x}_k$ for given $\underline{u}_k$ according to (2.1). Because of the additivity of the noise term in (2.1), the distribution P^x_{k+1} of $\underline{\boldsymbol{x}}_{k+1}$ depends on the distribution of the noise vector $\underline{\boldsymbol{w}}_k$.

Remark 2.2 With the assumption of a Markov decision process and the additivity of the value function (2.5), the requirements of Remark 2.1 are fulfilled.

Reformulation of the Value Function

According to [BS96], (2.5) can be substituted by

$$J_k(\underline{x}_k) = \min_{\underline{u}_k} \left(g_k(\underline{x}_k, \underline{u}_k) + \mathop{\mathrm{E}}_{\underline{\boldsymbol{w}}_k} \left[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}) \right] \right) \ , \tag{2.6}$$

where

$$g_k(\underline{x}_k, \underline{u}_k) := \int_{\mathbb{R}^n} \tilde{g}_k(\underline{x}_k, \underline{u}_k, \underline{w}_k) \, \mathrm{P}^w(\underline{w}_k) \, \mathrm{d}\underline{w}_k \ ,$$

and $\mathrm{P}^w(\underline{w}_k)$ denotes the probability density function of the noise term. The terminal cost

$$J_N(\underline{x}_N) = g_N(\underline{x}_N) \tag{2.7}$$

is independent of the control variable.

Definition 2.3 (Bellman equation) Equation (2.6) is referred to as *Bellman equation*, *optimality equation*, or *dynamic programming equation*.

2.2.3 Recursive Calculation of the Minimal Expected Cost-to-Go

Theorem 2.2 *For every initial state $\hat{\underline{x}}_0$, the minimal expected cost-to-go $J(\hat{\underline{x}}_0)$ of (2.4) is equal to $J_0(\hat{\underline{x}}_0)$, where the function J_0 is given by the last step of the following algorithm, which proceeds backward in time from time step N to time step 0.*

Algorithm 1 Dynamic programming

1: $J_N(\underline{x}_N) = g_N(\underline{x}_N)$ ▷ determination of the terminal cost
2: **for** $k = N-1$ to 0 **do**
3:

$$J_k(\underline{x}_k) = \min_{\underline{u}_k} \left(g_k(\underline{x}_k, \underline{u}_k) + \mathop{\mathrm{E}}_{\underline{w}_k} \left[J_{k+1} \left(\underline{f}(\underline{x}_k, \underline{u}_k) + \underline{\boldsymbol{w}}_k \right) \right] \right) \tag{2.8}$$

$$\underline{\mu}_k^*(\underline{x}_k) = \arg\min_{\underline{u}_k} \left(g_k(\underline{x}_k, \underline{u}_k) + \mathop{\mathrm{E}}_{\underline{w}_k} \left[J_{k+1} \left(\underline{f}(\underline{x}_k, \underline{u}_k) + \underline{\boldsymbol{w}}_k \right) \right] \right) \tag{2.9}$$

4: **end for**
5: $J(\hat{\underline{x}}_0) = J_0(\hat{\underline{x}}_0)$

Proof. A general proof can be found in [Ber00a] or [Put05]. □

2.2.4 Properties of Dynamic Programming

Dynamic programming provides a general framework for solving nonlinear optimization problems with additive cost functions. The efficient backward algorithm computes the expected total cost for the entire decision-making horizon. Furthermore, the minimal expected cost-to-go from each time step to the end of the regarded horizon is obtained for any optimal sequence of control laws. At each time step, the results of the previous step are employed. Then, the computation at time step k requires only the knowledge of the transition cost g_k, the transition density $\mathrm{P}_{k+1}^x(\,\cdot\,|\underline{x}_k, \underline{u}_k)$, and the result J_{k+1} of the previous time step. Because of the Markov property, J_k depends on former states and decisions only through the state $\underline{x}_k$. Therefore, the computational effort is reduced significantly, compared with a brute force algorithm, that is, the evaluation of all possible control sequences. If $\underline{u}_k^* = \underline{\mu}_k^*(\underline{x}_k)$ minimizes the right-hand side of (2.8) for each $\underline{x}_k$ and k, the control sequence $(\underline{u}_0^*, \ldots, \underline{u}_{N-1}^*)$ is optimal.

Remark 2.3 Instead of performing the minimization over all possible control sequences $\pi_k = (\underline{\mu}_k, \ldots, \underline{\mu}_{N-1})$, dynamic programming recursively determines the optimal state-feedback control $\underline{u}_k^*$ for each time step, which reduces the computational effort significantly.

Even if Algorithm 1 can be employed to derive closed-form expressions for the value function J_k, analytical solutions cannot be obtained in many practical problems. In these cases, numerical solutions are inevitable, which require much computation time. Nevertheless, dynamic programming is the only general approach for sequential optimization in case of stochastic systems. Especially in case of finite-horizon, discrete-time Markov decision processes with a finite number of states, dynamic programming provides an efficient method for solving the optimal control problem. Thus, dynamic programming is often employed as the basis for practical, but suboptimal, approaches [Ber00a].

Remark 2.4 In the special case of finitely many states in each computation step, the structure of dynamic programming allows an initial offline computation of the algorithm. Then, a look-up table is obtained, which contains the optimal state-feedback control for each state and each time step of the considered horizon. During runtime, the optimal control can be determined with the pre-calculated look-up table.

Review: Problems caused by noise affecting nonlinear systems

To obtain the optimal value $\underline{u}_k^*$ solving the minimization problem (2.6), a nonlinear optimization method is needed, which is a difficult task in general, especially when noise affects the system. Additional difficulties originating from the system stochastics to derive an analytically exact solution are indicated in the following. Equation (2.6) can be reformulated, such that

$$\begin{aligned} J_k(\underline{x}_k) &= g_k(\underline{x}_k, \underline{u}_k^*) + \underset{\underline{w}_k}{\mathrm{E}}\left[J_{k+1}(\underline{\boldsymbol{x}}_{k+1})\right] \\ &= g_k(\underline{x}_k, \underline{u}_k^*) + \int\limits_{\mathbb{R}^n} \mathrm{P}_{k+1}^x(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k^*) J_{k+1}(\underline{x}_{k+1})\,\mathrm{d}\underline{x}_{k+1} \end{aligned} \tag{2.10}$$

with a transition density $\mathrm{P}_{k+1}^x(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k^*)$. Using the fact that noise affects the system additively, (2.10) can be rewritten by means of

$$J_k(\underline{x}_k) = g_k(\underline{x}_k, \underline{u}_k^*) + \int\limits_{\mathbb{R}^n} \mathrm{P}^w(\underline{x}_{k+1} - \underline{f}(\underline{x}_k, \underline{u}_k^*)) J_{k+1}(\underline{x}_{k+1})\,\mathrm{d}\underline{x}_{k+1} \ , \tag{2.11}$$

where $\mathrm{P}^w(\underline{w}_k)$ denotes the noise density. It is important to mention that $\underline{w}_k = \underline{x}_{k+1} - \underline{f}(\underline{x}_k, \underline{u}_k^*)$. Owing to the integral, no general analytical solution to (2.11) is known, even if P^w is a Gauss function. The complexity of the equation arises, since the stochastics of the system require the consideration of the expectation value. Therefore, exact solutions to the optimal control problem for nonlinear probabilistic systems do not exist in general, and suboptimal solutions have to be derived.

2.2.5 Computational Effort in Case of Discrete Problems

With Q states and M possible controls at each computation step, the dynamic programming algorithm requires NMQ^2 multiplications to evaluate and determine the optimal policy. Therefore, the dynamic programming algorithm is of order

$$\mathcal{O}\left(NMQ^2\right) \ . \tag{2.12}$$

By contrast, a brute force algorithm, where $(M^Q)^N$ deterministic Markov policies are considered, is of exponential order. The direct evaluation of each of these policies requires NMQ multiplications. Taking everything into account, the brute force algorithm is of order

$$\mathcal{O}\left(M^{QN+1}QN\right) \ .$$

Thus, the dynamic programming algorithm yields a significant reduction of computation time.

2.2.6 Limits of Dynamic Programming

According to (2.12), the computational requirements of dynamic programming are overwhelming, when the number of states and controls becomes large. Thus, suboptimal solutions are required in many technically applications. There are two general approaches to obtain approximate solutions. In the first approach, the original problem is approximated. In the second approach, the dynamic programming algorithm for the original problem is approximated by a computationally simpler one. These methods are called *problem approximation* and *algorithm approximation*, respectively.

Problem Approximation

Discretization of continuous state- or control spaces is a common method to simplify the problem. The infinite (and possibly uncountable) state space is replaced with a finite set of states. Furthermore, the system equation has to be adapted appropriately. In this case, the transition mappings can be described by means of matrices. Then, the dynamic programming algorithm yields an optimal solution to the simplified problem. Detailed examples of this kind of problem approximation are given in Appendix F.3 and [Dei04].

Value Function Approximation

When considering the approximation of the algorithm, an explicit discretization of the state space is not necessary. The general idea is to approximate the value function $J_k(\underline{x}_k)$ by means of a suitable function $\tilde{J}_k(\underline{x}_k, \underline{r}_k)$, where $\underline{r}_k$ denotes a vector of proper parameters. An example for value function approximation is given in [NB03], where the value function is approximated by means of a radial basis function network with a finite number of Gaussian kernels. Evaluation of this network at the mean values of the Gauss functions yields a finite Markov decision problem, which can be solved approximately.

2.3 Related Work

Since dynamic programming offers a general framework to solve the optimal control problem for nonlinear systems, numerous approaches are based on this algorithm. In case of finite state and control spaces, dynamic programming is able to derive the correct solution to the optimal control problem for deterministic and stochastic systems [Ber00a].

If the state and control spaces are considered as continuous-valued sets, dynamic programming is not directly applicable in general, since the solution of the Bellman equation (2.6) is a difficult

task in general. Several approaches to obtain approximate solutions to the optimal control problem for nonlinear systems with continuous state spaces can be found in the literature. Standard approximations employ state space discretization to apply dynamic programming, since dynamic programming is in general not directly applicable in case of nonlinear systems with continuous-valued state spaces. All these discretization schemes suffer from the curse of dimensionality. To reduce this effect, value function approximation is often employed. For an infinite horizon, such an approximation scheme by means of radial basis functions is proposed in [NB03], which also leads to an indirect discretization of the problem.

Alternatively, optimality conditions can be derived to avoid the evaluation of the Bellman equation (2.6). Employing the dynamic programming equation (2.6), Pontryagin's maximum principle[3] offers necessary optimality conditions in case of deterministic systems, in continuous time as well as in discrete time [Ber00a]. These conditions can be employed to reformulate the optimal control problem as a two-point boundary-value problem (TPBVP) that is numerically solvable.

Pontryagin's maximum principle has been extensively discussed in numerous articles in case of stochastic, but continuous-time, systems. In [CH94], a stochastic maximum principle for singular control problems is introduced, where linear dynamics, convex cost, and convex constraints are considered, such that convex optimization methods can be employed. This approach is extended in [BM05], where necessary optimality conditions for singular control problems are derived for a continuous-time, nonlinear system. The considered control domain is not necessarily convex anymore. Hence, convex analysis methods are replaced with the perturbation of the optimal system input. To treat continuous-time stochastic problems, a common assumption is an underlying Ito process to represent the influence of white noise. A general stochastic minimum principle for continuous-time systems in local form is presented in [Pen90]. There, a variational approach is employed to derive adjoint equations and, subsequently, a global maximum principle in case of non-convex control domains. Combining Ito's Lemma with the stochastic dynamic programming formulation, Pontryagin's maximum principle is extended to continuous-time stochastic systems in [RRD04]. This approach is based on the equivalence of the adjoint equations of the maximum principle and the partial derivatives of the objective function with respect to the state variable of the dynamic programming approach.

Based on this assumption, in [LW02], the existence of solutions to a stochastic two-point boundary-value problem is examined, resulting from the application of the maximum principle. An expedient approach to solve the nonlinear equation system resulting from the TPBVP numerically, is to employ a continuation process [RD83]. Thereby, a solution to an easily solvable initial problem can be calculated. While the initial problem is being continuously transformed into the original problem, the solution is being traced. In [OF94], stabilizing continuation processes are proposed to derive a solution to an optimal control problem of nonlinear, continuous-time systems with general boundary constraints. The optimal control problem is

3 Pontryagin's maximum principle can be employed to minimize a function f by maximizing $-f$.

reformulated as an initial value problem of finite-dimensional ordinary differential equations, which can be solved numerically. In [Oht00], the time domain of a continuous-time system with general boundary constraints is modified by means of a continuation process. The initialization of a one-point horizon yields an optimal input to a nonlinear system, which is traced toward the solution for the whole considered horizon of length T, such that model predictive control can be applied. Another approach to employ a continuation method in the area of optimal control of nonlinear systems is to initialize the continuation process with a related linear system. A continuous transformation of this linear system toward the original nonlinear system, while tracing the solution, yields the desired control of the original system. For a deterministic continuous-time system, this idea has been successfully applied in [TJ79].

Nevertheless, an equivalent to the maximum principle or the TPBVP for stochastic systems in the technically important discrete-time case has not been found in literature yet. This lack may be due to the fact that the properties of the Ito process and Taylor series expansion with respect to the time variable cannot be equivalently applied to nonlinear, discrete-time systems. Therefore, alternative solutions have to be employed to solve the optimal control problem for discrete-time, stochastic systems approximately. A possible approximation scheme is given by value function approximation.

A common approach to value function approximation is to interpolate the value function. The interpolating function offers an easier representation of the original function. Only a small parameter set has to be stored to describe the interpolant completely. Besides, only a small number of grid points are required, compared with simple state space discretization. Therefore, dynamic programming can be performed more efficiently. The Bellman equation of the interpolation scheme is formally equivalent to the Bellman equation of a stochastic model on a fine grid, and all the convergence properties for this model carry over to the interpolation model. Linear interpolation is very popular and easy, but suffers from low accuracy [JK01]. Hence, higher-degree polynomials are employed to improve the accuracy. One main problem with these polynomials is that they tend to oscillations. Moreover, minimization becomes a serious problem. Because of that, piecewisely defined lower-degree polynomials, that is, linear, quadratic [Sch83], or cubic [JSS$^+$93], are exploited to avoid those problems. In case of known properties of the value function, for instance, concavity or monotony, general parametric approximation schemes may fail to preserve these properties [Jud98]. Therefore, in [Sch83], shape-preserving splines are proposed to keep known properties of the value function. Unfortunately, it may happen that the value iteration algorithm [Ber00b], which solves the Hamilton-Jacobi-Bellman equation iteratively for an infinite horizon, diverges, when these splines are employed. Only with several assumptions, shape-preserving splines avoid this divergence, which is proved in [JS94]. In many practical cases, the value function is assumed to be twice continuously differentiable. To guarantee continuity of the approximating function up to and including the second derivative, cubic polynomials, such as Hermitian polynomials or cubic splines, are required [dB78].

Taking everything into account, dynamic programming provides exact solutions to the optimal control problem in case of finite sets of states and controls. In the more realistic case of

continuous-valued state and control spaces, promising approaches to solve the optimal control problem, exist for deterministic (discrete-time and continuous-time) systems, when Pontryagin's maximum principle is employed. This approach can be extended to continuous-time, noise affected systems. However, for the technically important discrete-time systems suffering from noise disturbances, no general approach exists. Therefore, in this thesis, an approach to optimal control of nonlinear, stochastic, discrete-time systems with continuous state and control spaces is provided.

CHAPTER 3

New Approach to Optimal Control of Nonlinear Noise-Affected Systems

Exact analytical solutions to the continuous-valued optimal control problem for nonlinear, noise-affected systems do not exist in general, and suboptimal solutions have to be derived [Ber00a]. One possible approximation scheme to obtain such a solution by means of the dynamic programming approach is described in the remainder of this chapter.

Review: Objective

For the discrete-time system

$$\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{\boldsymbol{x}}_k, \underline{u}_k) + \underline{\boldsymbol{w}}_k \ , \ \ k = 0, \ldots, N-1 \ , \tag{3.1}$$

the objective is to establish an optimal control law

$$\underline{\mu}_k^* : \left\{ \begin{array}{rcl} \mathbb{R}^n & \rightarrow & \mathbb{R}^m \\ \underline{x}_k & \mapsto & \underline{\mu}_k^*(\underline{x}_k) = \underline{u}_k^* \end{array} \right.$$

that maps the states $\underline{x}_k$ onto optimal controls $\underline{u}_k^* = \underline{\mu}_k^*(\underline{x}_k)$ minimizing the expected cost-to-go. The recursively defined value function employed by DP to determine the minimal expected cost-to-go, is given by

$$J_N(\underline{x}_N) = g_N(\underline{x}_N) \tag{3.2}$$

$$J_k(\underline{x}_k) = \min_{\underline{u}_k} \left(g_k(\underline{x}_k, \underline{u}_k) + \mathop{\mathrm{E}}_{\underline{w}_k} \left[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}) \right] \right) \ , \ k = N-1, \ldots, 0 \ . \tag{3.3}$$

Starting from the recursive Bellman equation (3.3), the value function is approximated by means of Taylor series expansion up to second-order derivatives in the proposed approach. Then, a stochastic version of the minimum principle is derived, where the definition of a stochastic Hamilton function is employed. Incorporating costate equations[1], the optimal control problem

1 also known as adjoint equations

is reduced to a two-point boundary-value problem for the approximated value function. The resulting nonlinear equation system is solved numerically. A continuation process is a promising method to reduce numerical problems when solving this equation system and, therefore, employed in this thesis. The control sequence solving the two-point boundary-value problem is employed as prior knowledge to a subsequent algorithm, which explicitly considers the noise influence. An adaptive grid of few points is determined to restrict the state space. On this grid, stochastic dynamic programming is performed, where a piecewise interpolation of the value function by means of cubic splines is employed. Finally, an approximate solution to the optimal control problem of stochastic, nonlinear systems is derived, which treats the control variable as well as the state variable continuously.

3.1 Theoretical Results

In the following, the theoretical results of this thesis are described to develop the approximate solution to the considered continuous-valued optimal control problem of nonlinear, noise-affected systems.

In Section 3.1.1, the value function is approximated in order to rewrite the optimal control problem as an equivalent minimization problem[2]. Section 3.1.2 deals with the derivation of a stochastic minimum principle, which is applied to the considered approximated value function. Possible extensions of the value function approximation are discussed in Section 3.1.3. The minimization problem is treated in Section 3.1.4. How the resulting control sequence is employed as prior knowledge to a subsequent algorithm to restrict the state space, is described in Section 3.1.5. Finally, the accuracy of cubic spline interpolation is analyzed in Section 3.1.6.

3.1.1 Approximation of the Value Function

There are two reasons to approximate the value function. As described in Section 2.2.6, the value function can be approximated to perform dynamic programming without explicit discretization of the state space. Furthermore, the approximation of the value function can be employed to determine an estimate of the minimal expected cost-to-go without the desire to apply the minimizing control sequence. Possible applications can be found, for instance, in economy, finance, and engineering, where estimates of the minimal expected cost-to-go are often required.

Notation. In the following, $\underline{u}_k^*$ denotes a minimizing vector $\underline{u}_k$ of (3.3).

With the assumption that third- and higher-order derivatives are negligible, an approximation of J_{k+1} in (3.3) by means of Taylor series expansion up to second-order derivatives around the

2 with respect to the approximated value function

deterministic part $\underline{f}(\underline{x}_k, \underline{u}_k^*)$ of the state

$$\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{x}_k, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k$$

is given by

$$J_{k+1}(\underline{\boldsymbol{x}}_{k+1}) \approx J_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right) + \frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}} \underline{\boldsymbol{w}}_k + \frac{1}{2}\, \underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{H}_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right) \underline{\boldsymbol{w}}_k \ , \tag{3.4}$$

where $\mathbf{H}_k$ denotes the Hesse matrix of the value function. Then, the approximation of the value function $J_k(\underline{x}_k)$ in (3.3) is given by

$$J_k(\underline{x}_k) \approx g(\underline{x}_k, \underline{u}_k^*) + J_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right) + \frac{1}{2} \operatorname{tr}\left(\boldsymbol{\Sigma}_w \mathbf{H}_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right)\right) \ , \tag{3.5}$$

where the property

$$\underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{H}_{k+1} \underline{\boldsymbol{w}}_k = \operatorname{tr}\left(\underline{\boldsymbol{w}}_k \underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{H}_{k+1}\right)$$

has been exploited. The covariance matrix of the noise term is given by

$$\boldsymbol{\Sigma}_w = \underset{\underline{w}_k}{\mathrm{E}}\left[\underline{\boldsymbol{w}}_k \underline{\boldsymbol{w}}_k^{\mathrm{T}}\right] \ .$$

Moreover, the gradient in (3.4) vanishes, due to the zero-mean of the noise term together with the computation of the expectation value. A recursive calculation of the Hessian is given later in Theorem 3.3.

Remark 3.1 It is important to mention that (3.5) is similar to the value function of a deterministic system. The only difference is the last term

$$\frac{1}{2} \operatorname{tr}\left(\boldsymbol{\Sigma}_w \mathbf{H}_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right)\right) \ , \tag{3.6}$$

which is caused by the incorporation of the noise influence.

Considering (3.5), the value function J_k and its Hesse matrix $\mathbf{H}_k$ are evaluated at states, which would originate from a deterministic state propagation given by

$$\underline{x}_{k+1} = \underline{f}(\underline{x}_k, \underline{u}_k^*) \ . \tag{3.7}$$

Because of that, the state propagation (3.7) is sufficient to determine the value of the value function, if the approximation (3.5) is employed at each time step, that is,

$$J_0(\underline{x}_0) \approx g(\underline{x}_0, \underline{u}_0^*) + \sum_{k=1}^{N} \left(\frac{1}{2} \operatorname{tr}\left(\boldsymbol{\Sigma}_w \mathbf{H}_k(\underline{x}_k)\right) + g_k(\underline{x}_k, \underline{u}_k^*)\right) \ .$$

In this case, the expectation value is not required anymore, and the additional term in the value function accounts for the noise that affects the system.

Employing Taylor series expansion again to approximate the gradient of the value function up to second-order derivatives, the linearized gradient of the value function is given by

$$\frac{\partial J_{k+1}(\underline{x}_{k+1})}{\partial \underline{x}_{k+1}} \approx \frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}} + \frac{\partial^2 J_{k+1}\left(\underline{f}(\underline{x}_k, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^2} \underline{\boldsymbol{w}}_k \ . \tag{3.8}$$

With the approximations (3.5), (3.7), and (3.8) of the stochastic system, a minimum principle can be applied to the considered stochastic system as described in the following.

3.1.2 Minimum Principle

In case of the stochastic system (3.1), a necessary minimum condition to obtain the original value function $J_k(\underline{x}_k)$ is to find the roots of the partial derivative with respect to $\underline{u}_k$ of the term to be minimized. This leads to

$$\frac{\partial g_k(\underline{x}_k, \underline{u}_k^*)}{\partial \underline{u}_k} + \mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}(\underline{\boldsymbol{x}}_{k+1})}{\partial \underline{x}_{k+1}}\right] \frac{\partial \underline{f}(\underline{x}_k, \underline{u}_k^*)}{\partial \underline{u}_k} = \underline{0}^{\mathrm{T}} \ , \tag{3.9}$$

when the chain rule is employed.

Definition 3.1 (Costate) The *costate* is defined as the gradient of the value function evaluated at one specific point $\underline{x}_k$, that is,

$$\underline{p}_k^{\mathrm{T}} := \frac{\partial J_k(\underline{x}_k)}{\partial \underline{x}_k} \ . \tag{3.10}$$

Therefore, $\underline{p}_k \in \mathbb{R}^n$ is a vector, *not* a function.

Notation. In the following, for an optimal sequence of state-feedback controls $(\underline{u}_0^*, \ldots, \underline{u}_{N-1}^*)$ minimizing (3.3) for $k = 0, \ldots, N-1$, the corresponding state sequence, according to the state propagation (3.7), is denoted by $(\underline{x}_0^*, \ldots, \underline{x}_N^*)$.

Theorem 3.1 *Employing the approximations (3.5), (3.7), and (3.8), a recursive calculation of the costate along the optimal sequence of state-feedback controls and the corresponding state sequence is given by*

$$\underline{p}_N^{\mathrm{T}} = \frac{\partial g_N(\underline{x}_N^*)}{\partial \underline{x}_N} \ , \tag{3.11}$$

$$\underline{p}_k^{\mathrm{T}} = \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} + \underline{p}_{k+1}^{\mathrm{T}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} \ , \ k = N-1, \ldots, 0 \ . \tag{3.12}$$

Proof. $k = N$:

$$\underline{p}_N^{\mathrm{T}} = \frac{\partial J_N(\underline{x}_N^*)}{\partial \underline{x}_N} = \frac{\partial g_N(\underline{x}_N^*)}{\partial \underline{x}_N}$$

according to Definition 3.1.

$k \in \{N-1, \ldots, 0\}$: The dynamic programming equation (3.3) yields

$$\begin{aligned}
\underline{p}_k^{\mathrm{T}} &= \frac{\partial J_k(\underline{x}_k^*)}{\partial \underline{x}_k} \\
&= \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} + \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{u}_k} \frac{\partial \underline{\mu}_k(\underline{x}_k^*)}{\partial \underline{x}_k} + \mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}(\underline{\boldsymbol{x}}_{k+1})}{\partial \underline{x}_{k+1}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k}\right] \\
&\quad + \mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}(\underline{\boldsymbol{x}}_{k+1})}{\partial \underline{x}_{k+1}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{u}_k} \frac{\partial \underline{\mu}_k(\underline{x}_k^*)}{\partial \underline{x}_k}\right] \ ,
\end{aligned} \tag{3.13}$$

where $\underline{\boldsymbol{x}}_{k+1}$ denotes the one-step prediction by means of the system function (3.1) starting from the state $\underline{x}_k^*$. Employing the necessary minimum condition (3.9) for $J_k(\underline{x}_k^*)$, (3.13) can be

rewritten as

$$\frac{\partial J_k(\underline{x}_k^*)}{\partial \underline{x}_k} = \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} + \underset{\underline{w}_k}{\mathrm{E}} \left[\frac{\partial J_{k+1}(\underline{\boldsymbol{x}}_{k+1})}{\partial \underline{x}_{k+1}} \right] \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} \tag{3.14}$$

for $k = N-1, \ldots, 0$. Considering (3.10), (3.12), and (3.14), it remains to show that for a given state $\underline{x}_k^*$ the equality

$$\underset{\underline{w}_k}{\mathrm{E}} \left[\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k\right)}{\partial \underline{x}_{k+1}} \right] = \frac{\partial J_{k+1}(\underline{x}_{k+1}^*)}{\partial \underline{x}_{k+1}} \tag{3.15}$$

is satisfied. Because of the assumptions of Theorem 3.1, the states $\underline{x}_k, k = 1, \ldots, N$, are calculated by means of (3.7). Taking the expectation value, (3.8) can be rewritten as (3.15) and the proof of (3.12) is concluded. □

Definition 3.2 (Stochastic Hamiltonian) To define a *stochastic Hamilton function*, the influence of noise has to be incorporated. In case of system (3.1), this leads to the definition

$$H_k(\underline{x}_k, \underline{p}_{k+1}, \underline{u}_k, \underline{\boldsymbol{w}}_k) := g_k(\underline{x}_k, \underline{u}_k) + \underline{p}_{k+1}^{\mathrm{T}} \left(\underline{f}(\underline{x}_k, \underline{u}_k) + \underline{\boldsymbol{w}}_k\right) \tag{3.16}$$

for $k = N-1, \ldots, 0$.

Theorem 3.2 *Along the sequence of optimal state-feedback controls and the corresponding state sequence, the properties*

$$\frac{\partial}{\partial \underline{x}_k} \left(H_k(\underline{x}_k^*, \underline{p}_{k+1}, \underline{u}_k^*, \underline{\boldsymbol{w}}_k) \right) = \underline{p}_k^{\mathrm{T}} \ , \tag{3.17}$$

$$\frac{\partial}{\partial \underline{x}_k} \left(H_k(\underline{x}_k^*, \underline{p}_{k+1}, \underline{u}_k^*, \underline{\boldsymbol{w}}_k) \right) = \frac{\partial J_k(\underline{x}_k^*)}{\partial \underline{x}_k} \ , \tag{3.18}$$

$$\frac{\partial}{\partial \underline{u}_k} \left(H_k(\underline{x}_k^*, \underline{p}_{k+1}, \underline{u}_k^*, \underline{\boldsymbol{w}}_k) \right) = \underline{0}^{\mathrm{T}} \tag{3.19}$$

hold for $k = N-1, \ldots, 0$.

Proof. *Equation (3.17):* Let $k \in \{0, \ldots, N-1\}$. Then,

$$\frac{\partial H_k}{\partial \underline{x}_k} = \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} + \underline{p}_{k+1}^{\mathrm{T}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} = \underline{p}_k^{\mathrm{T}} \ ,$$

which proves (3.17).

Equation (3.18): follows immediately from (3.10) and (3.17).

Equation (3.19): Because of the approximation of the gradient of the value function by means of Taylor series expansion given by (3.8) and the necessary minimum condition (3.9) for the value function $J_k(\underline{x}_k^*)$, (3.15) holds. With property (3.15)

$$\begin{aligned} \frac{\partial H_k}{\partial \underline{u}_k} &= \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{u}_k} + \underline{p}_{k+1}^{\mathrm{T}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{u}_k} \\ &= \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{u}_k} + \frac{\partial J_{k+1}(\underline{x}_{k+1}^*)}{\partial \underline{x}_{k+1}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{u}_k} \end{aligned} \tag{3.20}$$

holds, which is equivalent to (3.9), and, therefore, concludes the proof. □

Corollary 3.1 (Stochastic Minimum Principle) With the assumptions of Theorem 3.1, a necessary minimum condition for the considered stochastic system along the optimal control sequence and the corresponding state sequence is given by (3.19), which can be evaluated by means of the stochastic Hamiltonian.

Theorem 3.3 *The Hesse matrix in (3.5) can be recursively calculated as follows, where the arguments of the functions are omitted to simplify the readability.*

$$\begin{aligned}
\mathbf{H}_N &= \frac{\partial^2 g_N}{\partial \underline{x}_N^2} \\
\mathbf{H}_k &= \frac{\partial^2 H_k}{\partial \underline{x}_k^2} + \left(\frac{\partial \underline{f}}{\partial \underline{x}_k}\right)^{\mathrm{T}} \mathbf{H}_{k+1} \frac{\partial \underline{f}}{\partial \underline{x}_k} \\
&\quad - \left[\left(\frac{\partial \underline{f}}{\partial \underline{x}_k}\right)^{\mathrm{T}} \mathbf{H}_{k+1} \left(\frac{\partial \underline{f}}{\partial \underline{u}_k}\right) + \frac{\partial^2 H_k}{\partial \underline{x}_k \partial \underline{u}_k}\right] \cdot \left[\left(\frac{\partial \underline{f}}{\partial \underline{u}_k}\right)^{\mathrm{T}} \mathbf{H}_{k+1} \frac{\partial \underline{f}}{\partial \underline{u}_k} + \frac{\partial^2 H_k}{\partial \underline{u}_k^2}\right]^{-1} \\
&\quad \cdot \left[\frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{x}_k} + \left(\frac{\partial \underline{f}}{\partial \underline{u}_k}\right)^{\mathrm{T}} \mathbf{H}_{k+1} \frac{\partial \underline{f}}{\partial \underline{x}_k}\right]
\end{aligned} \tag{3.21}$$

for $k = N-1, \ldots, 0$, where $\mathbf{H}_k$ denotes the Hesse matrix of the value function, and H_k refers to the Hamilton function.

Proof. *$k = N$:*

$$\mathbf{H}_N = \frac{\partial^2 J_N}{\partial \underline{x}_N^2} = \frac{\partial^2 g_N}{\partial \underline{x}_N^2} \ .$$

$k \in \{N-1, \ldots, 0\}$: Because of (3.18), the Hessian $\mathbf{H}_k$ can be calculated by means of the Hamiltonian along the optimal sequence of controls and the corresponding state sequence. Considering the gradient of H_k as a function of $\underline{x}_k$, $\underline{p}_{k+1}$, and $\underline{u}_k$[3], and assuming that $\mathbf{H}_{k+1}$ has already been computed, $\mathbf{H}_k$ is given as the second partial derivative of the Hamiltonian with respect to $\underline{x}_k$, that is,

$$\mathbf{H}_k = \frac{\partial^2 H_k}{\partial \underline{x}_k^2} + \frac{\partial^2 H_k}{\partial \underline{x}_k \partial \underline{p}_{k+1}} \mathbf{H}_{k+1} \frac{\partial \underline{f}}{\partial \underline{x}_k} + \left(\frac{\partial^2 H_k}{\partial \underline{x}_k \partial \underline{p}_{k+1}} \mathbf{H}_{k+1} \frac{\partial \underline{f}}{\partial \underline{u}_k} + \frac{\partial^2 H_k}{\partial \underline{x}_k \partial \underline{u}_k}\right) \frac{\partial \underline{\mu}_k}{\partial \underline{x}_k} \tag{3.22}$$

with unknowns $\frac{\partial^2 H_k}{\partial \underline{x}_k \partial \underline{p}_{k+1}}$ and $\frac{\partial \underline{\mu}_k}{\partial \underline{x}_k}$. If (3.17) and the costate recursion (3.12) are employed,

$$\frac{\partial^2 H_k}{\partial \underline{x}_k \partial \underline{p}_{k+1}} = \left(\frac{\partial \underline{f}}{\partial \underline{x}_k}\right)^{\mathrm{T}} \tag{3.23}$$

is fulfilled. Since (3.19) is satisfied for all $\underline{x}_k$,

$$\frac{\partial}{\partial \underline{x}_k}\left(\frac{\partial H_k(\underline{x}_k, \underline{p}_{k+1}, \underline{u}_k^*, \underline{\boldsymbol{w}}_k)}{\partial \underline{u}_k}\right) = \mathbf{0}$$

holds, that is,

$$\frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{x}_k} + \frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{p}_{k+1}} \mathbf{H}_{k+1} \left(\frac{\partial \underline{f}}{\partial \underline{x}_k} + \frac{\partial \underline{f}}{\partial \underline{u}_k} \frac{\partial \underline{\mu}_k}{\partial \underline{x}_k}\right) + \frac{\partial^2 H_k}{\partial \underline{u}_k^2} \frac{\partial \underline{\mu}_k}{\partial \underline{x}_k} = \mathbf{0} \ ,$$

3 The noise term $\underline{\boldsymbol{w}}_k$ vanishes, due to the additivity and its independence.

which leads to

$$\frac{\partial \underline{\mu}_k}{\partial \underline{x}_k} = -\left(\frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{p}_{k+1}} \mathbf{H}_{k+1} \frac{\partial \underline{f}}{\underline{u}_k} + \frac{\partial^2 H_k}{\partial \underline{u}_k^2} \right)^{-1} \cdot \left(\frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{x}_k} + \frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{p}_{k+1}} \mathbf{H}_{k+1} \frac{\partial \underline{f}}{\partial \underline{x}_k} \right) \quad , \tag{3.24}$$

where $\frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{p}_{k+1}}$ is unknown. With

$$\frac{\partial H_k}{\partial \underline{u}_k} = \frac{\partial g_k}{\partial \underline{u}_k} + \underline{p}_{k+1}^{\mathrm{T}} \frac{\partial \underline{f}}{\partial \underline{u}_k} \quad ,$$

it follows that

$$\frac{\partial^2 H_k}{\partial \underline{u}_k \partial \underline{p}_{k+1}} = \left(\frac{\partial \underline{f}}{\partial \underline{u}_k} \right)^{\mathrm{T}} \quad . \tag{3.25}$$

Insertion of (3.25) into (3.24), and subsequent insertion of (3.23) and (3.24) into (3.22) yields proposition (3.21) and concludes the proof of Theorem 3.3. □

3.1.3 Consideration of Higher-Order Derivatives in the Value Function Approximation

In principle, the consideration of higher-order derivatives in (3.8) is possible, if the existence of an inverse mapping of $\frac{\partial J_{k+1}}{\partial \underline{x}_{k+1}}$ can be guaranteed to satisfy (3.15), which is indicated in the following.

Dynamic Programming. In accordance with Theorem 2.2, the dynamic programming equation at time step $k+1$ is given by

$$J_{k+1}(\underline{x}_{k+1}) = g_{k+1}(\underline{x}_{k+1}, \underline{u}_{k+1}^*) + \underset{\underline{w}_{k+1}}{\mathrm{E}} \left[J_{k+2}(\underline{\boldsymbol{x}}_{k+2}) \right] \quad . \tag{3.26}$$

The next step in the DP algorithm (Algorithm 1) is the computation of the value function for the previous time step k, that is,

$$J_k(\underline{x}_k) = g_k(\underline{x}_k, \underline{u}_k^*) + \underset{\underline{w}_k}{\mathrm{E}} \left[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}) \right] \quad , \tag{3.27}$$

where the successor states $\underline{x}_{k+1}$ are not exactly known anymore, but given by their probability density functions. Therefore, the knowledge about the state $\underline{x}_{k+1}$ has changed from time step $k+1$ to k. Because of that, a one-step prediction has to be performed. In general,

$$J_{k+1}(\underline{x}_{k+1}) \neq \underset{\underline{w}_k}{\mathrm{E}} [J_{k+1}(\underline{\boldsymbol{x}}_{k+1})]$$

arising from (3.26) and (3.27), even if the system function (3.1) is known.

Costate Calculation. Similar to the previous paragraph, the costate recursion is discussed in the following. Reflecting the definition of the costate, that is,

$$\underline{p}_k^{\mathrm{T}} = \frac{\partial J_k(\underline{x}_k)}{\partial \underline{x}_k}$$

for $k = N, \ldots, 0$, the state realization of $\underline{x}_k$ must be known to determine the costate vector. In contrast to the consideration of the value function in (3.27), the gradient of the value function is employed in (3.10), evaluated at a point $\underline{x}_k$. It is important to mention that the costate is a deterministic value, not a random variable, and given by

$$\underline{p}_k^{\mathrm{T}} = \frac{\partial J_k(\underline{x}_k)}{\partial \underline{x}_k} = \frac{g_k(\underline{x}_k, \underline{u}_k^*)}{\partial \underline{x}_k} + \mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}(\underline{\boldsymbol{x}}_{k+1})}{\partial \underline{x}_{k+1}}\right] \frac{\partial \underline{f}(\underline{x}_k, \underline{u}_k^*)}{\partial \underline{x}_k} \ , \tag{3.28}$$

where the minimum condition (3.9) has been exploited. In (3.28), the state $\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{x}_k, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k$ is not known anymore, but only given by its probability density function depending on the probability density function of the noise term $\underline{\boldsymbol{w}}_k$.

To show that the costate recursion according to Theorem 3.1 is valid, that is,

$$\underline{p}_k^{\mathrm{T}} = \frac{\partial g_k(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} + \underline{p}_{k+1}^{\mathrm{T}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} \ ,$$

it has to be proved that

$$\frac{\partial J_{k+1}(\underline{x}_{k+1})}{\partial \underline{x}_{k+1}} \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} = \mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k\right)}{\partial \underline{x}_{k+1}}\right] \frac{\partial \underline{f}(\underline{x}_k^*, \underline{u}_k^*)}{\partial \underline{x}_k} \ ,$$

which is equivalent to

$$\frac{\partial J_{k+1}(\underline{x}_{k+1}^*)}{\partial \underline{x}_{k+1}} = \mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k\right)}{\partial \underline{x}_{k+1}}\right] \ . \tag{3.29}$$

The left-hand side of (3.29) describes the knowledge used at time step $k+1$, the right-hand side the knowledge at time step k. In case of a deterministic system, where the transition density can be interpreted as a Dirac function, (3.29) always holds. In case of stochastic systems, a deterministic approximation of the stochastic system function around the optimal control sequence and the corresponding states is desired, that is, a system function with

$$\underline{\tilde{f}}(\underline{x}_k^*, \underline{u}_k^*) \approx \underline{f}(\underline{x}_k^*, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k \ , \tag{3.30}$$

such that (3.29) holds. Taylor series expansion of the gradient of the value function is one way to obtain this result. According to Appendix A.2, the multi-dimensional Taylor series expansion of the gradient of the value function for additive noise in the system function (3.1) is given by

$$\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k\right)}{\partial \underline{x}_{k+1}} = \sum_{i=0}^{\infty} \frac{1}{i!} \frac{\partial^{i+1} J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^{i+1}} \bigotimes_{j=1}^{i} \underline{\boldsymbol{w}}_k \ ,$$

where the $\otimes$-operator has been employed. The $\otimes$-operator is introduced in Appendix A.1. Removing the first three terms from the summation, the gradient of the value function is given

by

$$\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)+\underline{\boldsymbol{w}}_k\right)}{\partial \underline{x}_{k+1}} = \frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}} + \frac{\partial^2 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^2}\underline{\boldsymbol{w}}_k + \frac{1}{2}\left(\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3}\otimes\underline{\boldsymbol{w}}_k\otimes\underline{\boldsymbol{w}}_k\right) + \sum_{i=3}^{\infty}\frac{1}{i!}\frac{\partial^{i+1} J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^{i+1}}\bigotimes_{j=3}^{i}\underline{\boldsymbol{w}}_k \quad . \tag{3.31}$$

Exploiting the properties of the $\otimes$-operator, the third term of (3.31) can be written as

$$\frac{1}{2}\left(\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3}\otimes(\underline{\boldsymbol{w}}_k\underline{\boldsymbol{w}}_k^{\mathrm{T}})\right) \quad .$$

Therefore, the expectation value of the gradient can be rewritten as

$$\mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)+\underline{\boldsymbol{w}}_k\right)}{\partial \underline{x}_{k+1}}\right] = \frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}} + \frac{1}{2}\mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3}\otimes(\underline{\boldsymbol{w}}_k\underline{\boldsymbol{w}}_k^{\mathrm{T}})\right] + \mathop{\mathrm{E}}_{\underline{w}_k}\left[\sum_{i=3}^{\infty}\frac{1}{i!}\frac{\partial^{i+1} J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^{i+1}}\bigotimes_{j=3}^{i}\underline{\boldsymbol{w}}_k\right] \quad , \tag{3.32}$$

since the second-order derivatives vanish, due to the zero-mean of the noise term. The term of the third-order derivatives of (3.32) can be simplified by means of

$$\frac{1}{2}\mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3}\otimes(\underline{\boldsymbol{w}}_k\underline{\boldsymbol{w}}_k^{\mathrm{T}})\right] = \frac{1}{2}\left(\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3}\otimes\boldsymbol{\Sigma}_w\right) \quad ,$$

where $\boldsymbol{\Sigma}_w$ denotes the covariance matrix of the noise term $\underline{\boldsymbol{w}}_k$.

To fulfill (3.29), the condition

$$\mathop{\mathrm{E}}_{\underline{w}_k}\left[\frac{\partial J_{k+1}(\underline{\boldsymbol{x}}_{k+1})}{\partial \underline{x}_{k+1}}\right] \stackrel{!}{=} \frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}} + \frac{1}{2}\left(\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3}\otimes\boldsymbol{\Sigma}_w\right) + \mathop{\mathrm{E}}_{\underline{w}_k}\left[\sum_{i=3}^{\infty}\frac{1}{i!}\frac{\partial^{i+1} J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^{i+1}}\bigotimes_{j=3}^{i}\underline{\boldsymbol{w}}_k\right]$$

must be satisfied for the Taylor series expansion of the gradient of the value function given by (3.32). Then, the function $\underline{\tilde{f}}$ in (3.30) can be defined as

$$\underline{\tilde{f}}(\underline{x}_k^*,\underline{u}_k^*) := \left[\frac{\partial J_{k+1}}{\partial \underline{x}_{k+1}}\right]^{-1}\left(\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}} + \frac{1}{2}\left(\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3}\otimes\boldsymbol{\Sigma}_w\right) + \mathop{\mathrm{E}}_{\underline{w}_k}\left[\sum_{i=3}^{\infty}\frac{1}{i!}\frac{\partial^{i+1} J_{k+1}\left(\underline{f}(\underline{x}_k^*,\underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^{i+1}}\bigotimes_{j=3}^{i}\underline{\boldsymbol{w}}_k\right]\right) \quad , \tag{3.33}$$

where $\left[\frac{\partial J_{k+1}}{\partial \underline{x}_{k+1}}\right]^{-1}$ denotes the inverse mapping of the gradient. If the Taylor series expansion of the gradient of the value function is truncated after the second-order derivative, the function

$\underline{\tilde{f}}$ is given by the deterministic function

$$\underline{x}_{k+1} = \underline{\tilde{f}}(\underline{x}_k^*, \underline{u}_k^*) = \left[\frac{\partial J_{k+1}}{\partial \underline{x}_{k+1}}\right]^{-1} \left(\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}}\right) = \underline{f}(\underline{x}_k^*, \underline{u}_k^*) \ . \tag{3.34}$$

Since the existence of the inverse mapping $\left[\frac{\partial J_{k+1}}{\partial \underline{x}_{k+1}}\right]^{-1}$ at the point

$$\frac{\partial J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}} + \frac{1}{2}\left(\frac{\partial^3 J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^3} \otimes \mathbf{\Sigma}_w\right) + \underset{\underline{w}_k}{\mathrm{E}}\left[\sum_{i=3}^{\infty} \frac{1}{i!} \frac{\partial^{i+1} J_{k+1}\left(\underline{f}(\underline{x}_k^*, \underline{u}_k^*)\right)}{\partial \underline{x}_{k+1}^{i+1}} \bigotimes_{j=3}^{i} \underline{\boldsymbol{w}}_k\right]$$

cannot be guaranteed in general, the simplified version (3.34) has been employed throughout this thesis. Even in case of the consideration of third-order derivatives, the mentioned problems arise.

3.1.4 Two-Point Boundary-Value Problem

Definition 3.3 (Two-point boundary-value problem) When ordinary differential equations are required to satisfy boundary conditions at more than one value of the independent variable, the resulting problem is called a multi-point boundary-value problem. In the special case of two boundary conditions, the problem is called *two-point boundary-value problem* (TPBVP). The most common case is, where boundary conditions are supposed to be satisfied at two points [PTVF02].

Remark 3.2 The distinction between initial value problems and TPBVPs is that in the first case, it is possible to start an acceptable solution at its beginning (initial values) and just march it along by numerical integration to its end (final values). In case of a TPBVP, the boundary conditions at the starting point do not define a unique solution to start with. An arbitrary choice among these solutions, satisfying these incomplete starting boundary conditions, is almost certain not to satisfy the second boundary condition [PTVF02].

Formulation

When assuming that the sequence of controls $(\underline{u}_0^*, \ldots, \underline{u}_{N-1}^*)$ is known, the corresponding states can be calculated by means of (3.7). After that, the corresponding costate sequence $(\underline{p}_N, \ldots, \underline{p}_0)$ is obtained by means of (3.11) and (3.12), starting from the final state $\underline{x}_N$ of the system iteration. Thus, the knowledge of the $\underline{u}_k^*$-sequence is sufficient to obtain the remaining information.

Remark 3.3 Since the only unknowns in the considered discrete-time case are the values $\underline{x}_N$ and $\underline{p}_0$, which can be determined by means of the state iteration (3.7) and the costate recursion (3.12), respectively, this problem is a two-point boundary-value problem.[4]

The boundary conditions are given by condition (3.11) and the costate $\underline{p}_0$. The control variable $\underline{u}_k^*$ is determined through the knowledge of $\underline{x}_k$ and $\underline{p}_{k+1}$ by means of the necessary minimum

4 The sequence of optimal controls is assumed to be known.

condition (3.19). Therefore, for given values $\underline{x}_k$ and $\underline{p}_{k+1}$, (3.19) and the state iteration yield a terminal state $\underline{x}_N$ depending on the costate $\underline{p}_0$. Hence, the initialization of the costate recursion is a function of the costate $\underline{p}_0$ for unknown values $\underline{u}_0^*, \ldots, \underline{u}_{N-1}^*$. That is,

$$\frac{\partial J_N(\underline{x}_N)}{\partial \underline{x}_N} = \frac{\partial J_N\big(\underline{x}_N(\underline{p}_0)\big)}{\partial \underline{x}_N}$$

implicitly depends on the value $\underline{p}_0$.

Solution

To solve the two point-boundary value problem, the necessary minimum condition (3.19) is solved simultaneously for all time steps of the entire decision-making horizon as follows. Introducing an augmented vector

$$\underline{U}^* := \left[\left(\underline{u}_0^*\right)^{\mathrm{T}} \quad \cdots \quad \left(\underline{u}_{N-1}^*\right)^{\mathrm{T}} \right]^{\mathrm{T}} \tag{3.35}$$

of the unknown optimal controls $\underline{u}_k^*$, $k = 0, \ldots, N-1$, the optimal state-feedback control for the current state $\underline{x}_0^* := \hat{\underline{x}}_0$ is given by $\underline{u}_0^*$. Employing (3.35), the necessary minimum conditions (3.19) for all time steps of the entire decision-making horizon can be rewritten by means of the nonlinear equation system

$$\underline{F}(\underline{U}^*) := \begin{bmatrix} \left(\frac{\partial}{\partial \underline{u}_0} H_0(\underline{x}_0^*, \underline{p}_1, \underline{u}_0^*, \underline{\boldsymbol{w}}_0)\right)^{\mathrm{T}} \\ \left(\frac{\partial}{\partial \underline{u}_1} H_1(\underline{x}_1^*, \underline{p}_2, \underline{u}_1^*, \underline{\boldsymbol{w}}_1)\right)^{\mathrm{T}} \\ \vdots \\ \left(\frac{\partial}{\partial \underline{u}_{N-1}} H_{N-1}(\underline{x}_{N-1}^*, \underline{p}_N, \underline{u}_{N-1}^*, \underline{\boldsymbol{w}}_{N-1})\right)^{\mathrm{T}} \end{bmatrix} = \underline{0} \tag{3.36}$$

with N nonlinear equations for the N unknown optimal controls $\underline{u}_0^*, \ldots, \underline{u}_{N-1}^*$. The nonlinear equation system (3.36) is numerically solvable. The employed method is described in Section 3.2.2

Remark 3.4 On the one hand, the employment of the proposed approximations of the value function and its gradient in the minimization procedure yields exactly the same result as treating a deterministic problem in case of the considered optimal control problem. Therefore, the noise term in (3.1) could have been neglected from the beginning. The reason is that the expectation value of the approximated gradient, which is employed in the costate recursion (Theorem 3.1) and the necessary minimum condition (3.19), equals the deterministic one. On the other hand, the proposed method is more general than the consideration of a deterministic system, since the noise vanishes during the approximation process and is not neglected from the beginning. Therefore, extensions are possible in principle, for example by using higher-order terms in the approximation of the gradient. In this case, Section 3.1.3 has to be considered to guarantee the validity. When only an estimate of the minimal expected cost-to-go is desired, the difference between a deterministic and a stochastic consideration of the problem is still given by the additional terms in (3.5).

3.1.5 Employment of Prior Knowledge

Up to this point, a sequence of state-feedback controls has been determined, which satisfies the necessary minimum condition (3.19) for $k = 0, \ldots, N-1$ in case of a value function approximated by means of (3.5). In case of value function approximation by means of Taylor series expansion up to second-order derivatives, that control sequence coincides with the control sequence, which would have been obtained, if just a *deterministic* system

$$\underline{x}_{k+1} = \underline{f}(\underline{x}_k, \underline{u}_k)$$

had been considered. Depending on the system properties and the strength of the noise influence, this approximation is more or less suitable. However, the sequence of controls resulting from the solution of (3.36) can be regarded as an approximation of the true optimal control sequence for the original *stochastic* system (3.1). This knowledge is exploited in the following to derive an improved solution to the original problem.

A promising approach to improve this approximate solution is the search around the sequence of successor states of the initial value $\hat{\underline{x}}_0$, which are obtained by the employment of the recently determined control sequence solving (3.36). Going back to the original system function (3.1), a restricted region of the state space has to be found, which covers possible successor states of $\hat{\underline{x}}_0$.

To obtain such a restriction of the state space, that is, an area of interest, the means and covariances of the successor states of $\hat{\underline{x}}_0$ are employed, where the control sequence for the N-step horizon from the solution of the nonlinear equation system (3.36) is applied. According to Section 2.2, in general, this computation cannot be solved analytically. Around the mean value $\overline{\underline{x}}_k$, a symmetric set $\mathcal{P}_k$ of $p+1$ points is heuristically determined. The set $\mathcal{P}_k$ depends on the covariance $\mathbf{C}_k^x$ of the random variable $\underline{\boldsymbol{x}}_k$ through a function $\underline{s}$, that is,

$$\mathcal{P}_k := \{\overline{\underline{x}}_k,\ \overline{\underline{x}}_k \pm \underline{s}(\mathbf{C}_k^x, i),\ i = 1, \ldots, p\} \quad . \tag{3.37}$$

Depending on the uncertainty of the random variable $\underline{\boldsymbol{x}}_k$, the sets $\mathcal{P}_k$, $k = 0, \ldots, N$, heuristically discretize the state space around the mean values $\overline{\underline{x}}_k$. Therefore, the spread of this region depends on the uncertainty of the state $\boldsymbol{x}_k$ for a k-step prediction. Within the range of the sets $\mathcal{P}_k$, $k = 0, \ldots, N$, the value function is interpolated by means of piecewisely defined cubic splines, such that stochastic dynamic programming can be applied to treat the continuous-valued optimal control problem approximately.

3.1.6 Accuracy of the Spline Interpolation

Remark 3.5 To derive error bounds for the interpolating spline of the value function, the following part is restricted to the scalar case.

Review: Cubic splines

Definition 3.4 (Cubic spline interpolant) In the scalar case, a *cubic spline interpolant* is a piecewise polynomial function $S : [a, b] \to \mathbb{R}$ consisting of cubic polynomial pieces $S_i : [\lambda_i, \lambda_{i+1}] \to \mathbb{R}$, where $a = \lambda_0 < \lambda_1 < \cdots < \lambda_N = b$ is a partition of the interval $[a, b]$. That is,

$$S_i(x) = a_i(x - \lambda_i)^3 + b_i(x - \lambda_i)^2 + c_i(x - \lambda_i) + d_i$$

and

$$S_i(x) \approx f(x)$$

for $x \in [\lambda_{i-1}, \lambda_i]$, $i = 1, \ldots, N$, where f denotes the function to be approximated. The given $N + 1$ points λ_i are called *knots*. The parameters a_i, b_i, c_i, d_i, $i = 0, \ldots, N$, are chosen, such that the polynomial pieces S_i of the interpolant S satisfy

$$S_i(\lambda_i) = f(\lambda_i),\ i = 0, \ldots, N \tag{3.38}$$

$$S_i(\lambda_{i+1}) = f(\lambda_{i+1}),\ i = 0, \ldots, N - 1 \tag{3.39}$$

$$\frac{\mathrm{d}}{\mathrm{d}x} S_{i-1}(\lambda_i) = \frac{\mathrm{d}}{\mathrm{d}x} S_i(\lambda_i),\ i = 1, \ldots, N - 1 \tag{3.40}$$

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} S_{i-1}(\lambda_i) = \frac{\mathrm{d}^2}{\mathrm{d}x^2} S_i(\lambda_i),\ i = 1, \ldots, N - 1 \ . \tag{3.41}$$

Furthermore, a possible boundary condition is given by

$$\frac{\mathrm{d}}{\mathrm{d}x} S_0(\lambda_0) = \frac{\mathrm{d}}{\mathrm{d}x} f(\lambda_0)$$
$$\frac{\mathrm{d}}{\mathrm{d}x} S_N(\lambda_N) = \frac{\mathrm{d}}{\mathrm{d}x} f(\lambda_N) \ ,$$

which define the slopes at the end of the interval $[a, b]$, [dB78].[5] The extension to higher dimensions is given in Appendix E.1.

According to (3.38)–(3.41), a cubic spline is continuous in the function value, the first derivative, and the second derivative at the knots and within the subintervals $[\lambda_i, \lambda_{i+1}]$, $i = 0, \ldots, N$. A scalar example for a piecewise cubic interpolating function is given in Figure 3.1. A possibly high-degree function is piecewisely approximated by means of low-degree polynomials, that is the function $f(x)$, where only the knots and the corresponding function values are known.

In accordance with [Ker71], an error bound for the piecewise approximation $S_f \in \mathcal{C}^2[a, b]$ of a function $f \in \mathcal{C}^4$ by means of cubic splines in an interval $[a, b]$, which is partitioned into $\lambda_0 = a < \ldots < \lambda_N = b$, is given by

$$\|S_f - f\|_\infty \in \mathcal{O}(h^4),\ h \to 0 \ , \tag{3.42}$$

5 Further existing possible boundary conditions are not mentioned here, since they are not employed in this thesis.

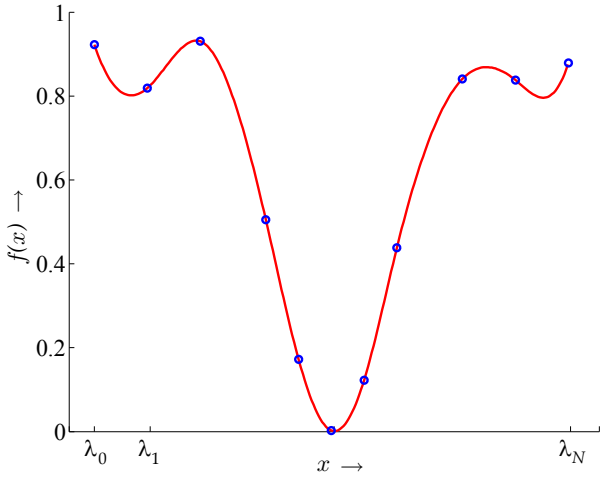


Figure 3.1: Example for an interpolating function, which is piecewisely defined by means of cubic splines. The interpolating function $f(x)$ is twice differentiable within the whole definition interval and coincides with the true, but possibly unknown, function at all knots.

where

$$h := \max_j (\lambda_{j+1} - \lambda_j) \ .$$

A definition of the ∞-norm is given in Appendix A.4.

In [Hal73], a more general bound for $f \in \mathcal{C}^m$, $m = 2, 3, 4$ is provided, such that

$$\|(S_f - f)^{(r)}\|_\infty \leq \varepsilon_{mr} \|f^{(m)}\|_\infty h^{m-r} + K_m \beta_r \left(2^{1-j} - 2^{1-N+j}\right) \ h \in \mathcal{O}(h^{2-r} + h^{m-r}) \qquad (3.43)$$

for $h \to 0$ and $\lambda_j \leq x \leq \lambda_{j+1}$. For $r = 0, 1, 2$, the values ε_{mr} are given in Table 3.1.

Table 3.1: Values for ε_{mr} for the general error bound of the spline approximation.

ε_{mr}	$r = 0$	$r = 1$	$r = 2$
$m = 2$	$\frac{9}{8}$	4	10
$m = 3$	$\frac{71}{216}$	$\frac{31}{27}$	5
$m = 4$	$\frac{5}{384}$	$\frac{9+\sqrt{3}}{216}$	5

Moreover, with

$$\Delta\lambda_j := \lambda_{j+1} - \lambda_j$$

the remaining parameters are given by Table 3.2.

Table 3.2: Remaining parameters for the general error bound of the spline approximation.

$\beta_0 = \frac{\Delta\lambda_j}{4}$	$\beta_1 = 1$	$\beta_2 = \frac{6}{\Delta\lambda_j}$
$K_2 = \frac{5}{2}\|f^{(2)}\|_\infty + R$	$K_3 = \|f^{(3)}\|_\infty h + \frac{R}{2}$	$K_4 = \frac{7}{24}\|f^{(4)}\|_\infty h^2 + \frac{R}{2}$

The last parameter R depends on the boundaries of the approximation interval and is given by

$$R = \max\left\{\|f^{(2)}(\lambda_0) - S_f^{(2)}(\lambda_0)\|, \ \|f^{(2)}(\lambda_N) - S_f^{(2)}(\lambda_N)\|\right\} \ .$$

An error bound for the approximation of the value function

$$J_k(x_k) = g_k(x_k, u_k^*) + \underset{w_k}{\mathrm{E}}[J_{k+1}(\boldsymbol{x}_{k+1})]$$

by means of a piecewisely defined cubic spline J_k^{spline}, to be employed in the proposed algorithm, is derived in the following. Therefore, the norm

$$\left\| J_k - J_k^{spline} \right\|_\infty$$

is considered.

According to Table 3.2, the norm

$$\|J_k^{(2)}\|_\infty$$

of the value function is required to compute an error bound for the accuracy of the interpolating spline. Since

$$J_k(x_k) = g_k(x_k, u_k^*) + \underset{w_k}{\mathrm{E}}[J_{k+1}(\boldsymbol{x}_{k+1})] \ ,$$

the problem of the treatment of the expectation value has to be solved. With

$$J_{k+1}(x_{k+1}) \geq 0$$

it follows that

$$\underset{w_k}{\mathrm{E}}[J_{k+1}(\boldsymbol{x}_{k+1})] \geq 0 \ ,$$

because the expectation value can be regarded as a convolution with the noise density P^w, which is the Gauss function in the considered case. Several important properties of the convolution operator are mentioned in the following part.

Properties of the Convolution Operator

One way to treat the expectation value is, to show that in the considered case the convolution $*$ operator can be employed to derive an upper bound of the L^∞-norm of the original function f according to

$$\|f * \mathrm{P}^w\|_\infty \leq \|f\|_\infty \ , \tag{3.44}$$

where P^w is a probability density function. In this case, an upper bound of the norm of the expectation value is given by

$$\left\| \underset{w_k}{\mathrm{E}}[J_{k+1}(\boldsymbol{x}_{k+1})] \right\|_\infty = \|J_{k+1}(x_{k+1}) * \mathrm{P}^w\|_\infty \leq \|J_{k+1}(x_{k+1})\|_\infty \ .$$

Theorem 3.4 *For $f \in L^p$ and $g \in L^1$, $p \geq 1$, the convolution $f * g$ exists and*

$$\|f * g\|_p \leq \|g\|_1 \|f\|_p \tag{3.45}$$

is satisfied.

Proof. The proof is given in [Aub00] and [Hol70]. For the definition of the L^p spaces it is referred to Appendix A.4. □

Theorem 3.5 *For $f \in L^\infty[a,b], \mathrm{P}^w \in L^1(\mathbb{R})$ the inequality*

$$\| f * \mathrm{P}^w \|_\infty \leq \|f\|_\infty$$

holds.

Proof. Let $p = \infty$. Then, the functions $\mathrm{P}^w \in L^1(\mathbb{R})$ and $f \in L^\infty[a,b]$ satisfy

$$\| f * \mathrm{P}^w \|_\infty \leq \|f\|_\infty \| \mathrm{P}^w \|_1 = \|f\|_\infty \ , \tag{3.46}$$

because of (3.45) and

$$\| \mathrm{P}^w \|_1 = \int_{\mathbb{R}} | \mathrm{P}^w(x)| \, \mathrm{d}x = \int_{\mathbb{R}} \mathrm{P}^w(x) \, \mathrm{d}x = 1 \ ,$$

where the density property of P^w has been employed. □

In the considered case, P^w is a Gaussian $\mathcal{N} = \mathcal{N}(x; \overline{x}, \sigma_x^2)$ with mean $\overline{x}$ and variance σ_x^2, and

$$\underset{w_k}{\mathrm{E}}[f] = f * \mathcal{N}$$

holds. Thus,

$$\|f\|_\infty \geq \|f * \mathcal{N}\|_\infty \ ,$$

and the desired upper bound is obtained.

Remark 3.6 For $f \in \mathcal{C}^0$ with compact support,

$$f * \mathrm{P}^w \in \mathcal{C}^\infty \ ,$$

which reveals the smoothing property of a probability density [Aub00].

Error Bound for the Spline Approximation

In the following, the value function J_{k+1} is assumed to be already approximated by means of a piecewisely defined cubic spline J_{k+1}^{spline}. The value function approximation J_{k+1}^{spline} is at least twice continuously differentiable because of its construction, that is $J_{k+1} \in \mathcal{C}^k$, $k \geq 2$.

However, the objective of the following part is to derive an upper bound for $\|J_k^{(2)}\|_\infty$ to apply (3.43), where the property (3.44) of the convolution operator is exploited.

Convention. The approximation error at time step $k+1$ is ignored at this point to derive a one-step error bound for the approximation of J_k at time step k. Therefore, the function J_{k+1}^{spline} is denoted by J_{k+1} in the following.

With (3.46)

$$\left\| \underset{w_k}{\mathrm{E}}[J_{k+1}(\boldsymbol{x}_{k+1})] \right\|_\infty \leq \|J_{k+1}(x_{k+1})\|_\infty \tag{3.47}$$

holds. Furthermore, the desired second derivative of the value function is given by

$$\begin{aligned}\left\|\frac{\partial}{\partial x_k}\left(\frac{\partial J_k(x_k)}{\partial x_k}\right)\right\|_\infty &= \left\|\frac{\partial}{\partial x_k}\left(\frac{\partial}{\partial x_k}\left(g_k(x_k,u_k^*)+\mathop{\mathrm{E}}_{w_k}\left[J_{k+1}(\boldsymbol{x}_{k+1})\right]\right)\right)\right\|_\infty \\ &= \left\|\frac{\partial^2}{\partial x_k^2}g_k(x_k,u_k^*)+\frac{\partial^2}{\partial x_k^2}\left(\mathop{\mathrm{E}}_{w_k}\left[J_{k+1}(\boldsymbol{x}_{k+1})\right]\right)\right\|_\infty .\end{aligned}$$

Since the expectation operator can be regarded as the convolution of two functions, which is just an integration, it is possible to exchange the order of differentiation and the expectation. Then,

$$\begin{aligned}\left\|\frac{\partial^2 g_k}{\partial x_k^2}+\frac{\partial^2}{\partial x_k^2}\left(\mathop{\mathrm{E}}_{w_k}\left[J_{k+1}(\boldsymbol{x}_{k+1})\right]\right)\right\|_\infty &= \left\|\frac{\partial^2 g_k}{\partial x_k^2}+\mathop{\mathrm{E}}_{w_k}\left[\frac{\partial^2}{\partial x_k^2}\left(J_{k+1}(\boldsymbol{x}_{k+1})\right)\right]\right\|_\infty \\ &= \left\|\mathop{\mathrm{E}}_{w_k}\left[\frac{\partial^2 g_k}{\partial x_k^2}+\frac{\partial^2}{\partial x_k^2}\left(J_{k+1}(\boldsymbol{x}_{k+1})\right)\right]\right\|_\infty ,\end{aligned}$$

due to the fact that $g_k = g_k(x_k, u_k^*)$ is independent of $\boldsymbol{w}_k$. With (3.47) it follows that

$$\begin{aligned}\left\|\frac{\partial}{\partial x_k}\left(\frac{\partial J_k(x_k)}{\partial x_k}\right)\right\|_\infty &\overset{(3.47)}{\leq} \left\|\frac{\partial^2 g_k}{\partial x_k^2}+\frac{\partial^2}{\partial x_k^2}\left(J_{k+1}(x_{k+1})\right)\right\|_\infty \\ &= \left\|\frac{\partial^2 g_k}{\partial x_k^2}+\frac{\partial}{\partial x_k}\left(\frac{\partial J_{k+1}}{\partial x_{k+1}}\frac{\partial f}{\partial x_k}\right)\right\|_\infty \\ &= \left\|\frac{\partial^2 g_k}{\partial x_k^2}+\frac{\partial^2 J_{k+1}}{\partial x_{k+1}^2}\left(\frac{\partial f}{\partial x_k}\right)^2+\frac{\partial J_{k+1}}{\partial x_{k+1}}\frac{\partial^2 f}{\partial x_k^2}\right\|_\infty ,\end{aligned}$$

and the desired upper bound for $\|J_k^{(2)}\|_\infty$ is obtained. With this result, an upper bound for the spline interpolation is given, when applying (3.43). In Appendix E.2, this bound is computed for a concrete example.

Remark 3.7 Similar to the previous part, where the interpolation is performed over the x-variable, an interpolation with respect to the u-variable of the cost function V is performed by the proposed algorithm.[6] This interpolation scheme is employed, such that the optimal control problem is treated continuously in the control variable. In this case, the approximation error of the derivative

$$\|(V_k^{spline} - V_k)^{(1)}\|_\infty$$

is important, since the derivative of V_k^{spline} is employed to find the desired minimum of the cost function. According to [Hal73], an approximate error bound is given by

$$\|(V_k^{spline} - V_k)^{(1)}\|_\infty \in \mathcal{O}(h)\,,\ h \to 0\ ,$$

that is, the error decreases at least linearly for $h \to 0$.

3.2 Practical Methods

This section deals with methods to put the theoretical results into practice. Two alternative methods to formulate the two-point boundary value problem are described in Section 3.2.1. In

6 The cost function is the non-minimized value function.

Section 3.2.2, a possible approach to solve the nonlinear equation system (3.36) is introduced. In Section 3.2.3, an approximate method is described to determine the desired restriction of the state space mentioned in Section 3.1.5. The determination of the grid points, which are employed to perform approximate dynamic programming, is motivated in Section 3.2.4. An approximation scheme of the value function on this grid to treat the continuous-valued optimal control problem is discussed in Section 3.2.5 in detail.

3.2.1 Formulation of the Two-Point Boundary-Value Problem

Alternative approaches to formulate the TPBVP as described in Section 3.1.4 are given by the shooting method or the direct method to be briefly introduced in the following.

Shooting Method

According to[SB02], the shooting method starts the iteration from the given initial state $\underline{\hat{x}}_0$ and an assumed costate $\underline{\tilde{p}}_0$ and calculates $\underline{x}_N$ and $\underline{\tilde{p}}_N$. That is, the second boundary condition (3.11) is a function of $\underline{\tilde{p}}_0$. In most cases,

$$\underline{\tilde{p}}_N^{\mathrm{T}} - \frac{\partial J_N\left(\underline{x}_N(\underline{\tilde{p}}_0)\right)}{\partial \underline{x}_N} \neq 0 \ .$$

This means, the boundary condition (3.11) is not fulfilled. Then, the initial vector $\underline{\tilde{p}}_0$ is modified to obtain the boundary condition $\underline{\tilde{p}}_N^{\mathrm{T}} = \frac{\partial J_N(\underline{x}_N(\underline{\tilde{p}}_0))}{\partial \underline{x}_N}$. The TPBVP is rewritten as a nonlinear equation system, which needs not to coincide with (3.36).

Since the optimal state-feedback control $\underline{u}_k^*$ is a function of the state and the costate, it is determined by the knowledge of $\underline{x}_k$ and $\underline{\tilde{p}}_{k+1}$.

Direct Method

Instead of an initial guess of the gradient of the value function, in the direct method initially a policy $\tilde{\pi} = (\underline{\tilde{u}}_0, \ldots, \underline{\tilde{u}}_{N-1})$ is assumed, and the corresponding state sequence $(\underline{\tilde{x}}_0, \ldots, \underline{\tilde{x}}_{N-1})$ is computed. After that, the costate sequence $(\underline{\tilde{p}}_{N-1}, \ldots, \underline{\tilde{p}}_0)$ is calculated starting from

$$\underline{\tilde{p}}_N^{\mathrm{T}} = \frac{\partial J_N(\underline{\tilde{x}}_N)}{\partial \underline{x}_N} \ .$$

For each $k \in \{0, \ldots, N-1\}$, the necessary minimum condition

$$\frac{\partial \tilde{H}_k(\underline{\tilde{x}}_k, \underline{\tilde{p}}_{k+1}, \underline{\tilde{u}}_k, \underline{\boldsymbol{w}}_k)}{\partial \underline{u}_k} = \underline{0}^{\mathrm{T}}$$

is evaluated. If this condition does not hold, the assumed policy $\tilde{\pi} = (\underline{\tilde{u}}_0, \ldots, \underline{\tilde{u}}_{N-1})$ has to be modified.

3.2.2 Solution of the Nonlinear Equation System

The nonlinear equation system (3.36) is in general difficult to solve, since most methods require an initial value $\hat{\underline{y}}_0$ sufficiently close to the solution, such that an iteration process

$$\underline{y}_{n+1} = \underline{\Phi}(\underline{y}_n) \tag{3.48}$$

converges. This means, $\hat{\underline{y}}_0$ lies in the domain of attraction of the iteration process, which converges to the fixed-point

$$\underline{y} = \underline{\Phi}(\underline{y})$$

of the function $\underline{\Phi}$. In case of the famous Newton iteration, a necessary minimum condition is evaluated. Linearization of the function $\underline{\Phi}$ yields the iteration scheme (3.48). To prove the convergence of the Newton iteration, Banach's fixed-point theorem can be employed. Since only a necessary minimum condition is evaluated, it is possible that the provided solution only reveals a local minimum, which is one further basic problem in nonlinear minimization.

To solve the two-point boundary-value problem, that is, equation system (3.36), a continuation process is implemented in this thesis, which is discussed in more detail in the following.

Continuation Process

The advantage of a continuation method, which is introduced in more detail in Appendix B, is that it is exhaustive under some conditions. That is, it is not necessary to know a starting value $\underline{y}_0$ close to the solution of the iteration (3.48). Furthermore, all solutions can be found. Moreover, continuation methods are well-suited for higher-dimensional problems with respect to conditioning [RD83].

With the employment of a continuation process, the nonlinear equation system (3.36) is embedded into a parameterized family of equation systems

$$\underline{F}(\underline{U}^*(\gamma)) := \begin{bmatrix} \left(\frac{\partial}{\partial \underline{u}_0} H_0(\underline{x}_0^*, \underline{p}_1(\gamma), \underline{u}_0^*(\gamma), \underline{\boldsymbol{w}}_0)\right)^{\mathrm{T}} \\ \left(\frac{\partial}{\partial \underline{u}_1} H_1(\underline{x}_1^*(\gamma), \underline{p}_2(\gamma), \underline{u}_1^*(\gamma), \underline{\boldsymbol{w}}_1)\right)^{\mathrm{T}} \\ \vdots \\ \left(\frac{\partial}{\partial \underline{u}_{N-1}} H_{N-1}(\underline{x}_{N-1}^*(\gamma), \underline{p}_N(\gamma), \underline{u}_{N-1}^*(\gamma), \underline{\boldsymbol{w}}_{N-1})\right)^{\mathrm{T}} \end{bmatrix} \stackrel{!}{=} \underline{0}\,,\ \gamma \in [0,1]\ , \tag{3.49}$$

such that for the parameter $\gamma = 0$ the solution to an easy problem is obtained, and for $\gamma = 1$ the original problem is described. Employing a discrete curve follower, the easy problem is being transformed into the original problem with an increasing parameter $\gamma_i,\ i = 0, \ldots, t$, where $0 = \gamma_0 \leq \gamma_i \leq \gamma_t = 1$. During this process, the solution to the problem is being traced. This means, that the solution for the previous value γ_{i-1} serves as an initial guess to calculate the solution for γ_i. Alternatively, a continuous curve follower can be employed as explained in Appendix B.

Definition 3.5 (Homotopy) For $\gamma \in [0,1]$, the continuously parameterized family of mappings $\underline{f}(\underline{x},\gamma)$ is called *homotopy between the functions* $\underline{f}(\underline{x},0)$ *and* $\underline{f}(\underline{x},1)$.

The nonlinear equation system (3.49) can be solved, for example, by means of a Newton iteration, where the initial guess of each step of the transformation process is assumed to be sufficiently close to the solution for problem depending on the current parameter γ_i. The desired solution is obtained for $\gamma_t = 1$. Instead of applying an iterative minimization method directly to (3.36) and suffering from poor initial values, this homotopy approach yields good starting values at each step, if the function $\underline{F}$ is sufficiently smooth for increasing γ_i. Therefore, it is possible to obtain the desired solution to the considered nonlinear system (3.36) numerically.

In the considered case, the stochastic nonlinear system (3.1) is parameterized, such that the easy problem is to find the optimal control for a linear system. This is due to the existence of optimal linear controllers for linear systems in case of a quadratic cost function, which is additive over time. These linear quadratic controllers (LQ controllers) provide optimal state-feedback controls by solving the discrete-time Riccati equation recursively. The derivation of such an LQ controller is given in Appendix C.

To obtain the desired linear system for $\gamma = 0$, the system description (3.1) can be changed into

$$\underline{\boldsymbol{x}}_{k+1} = \gamma\, \underline{f}(\underline{\boldsymbol{x}}_k, \underline{u}_k) + (1-\gamma)\, \underline{l}(\underline{\boldsymbol{x}}_k, \underline{u}_k) + \underline{\boldsymbol{w}}_k \tag{3.50}$$

with a linear function $\underline{l}$, such that the problem for $\gamma = 0$ consists in solving the LQ control problem. The original nonlinear system (3.1) is obtained for $\gamma = 1$.

Example 3.1: Homotopy between a linear function and a nonlinear function
In Figure 3.2, a scalar function is displayed, which depends on a homotopy parameter γ.

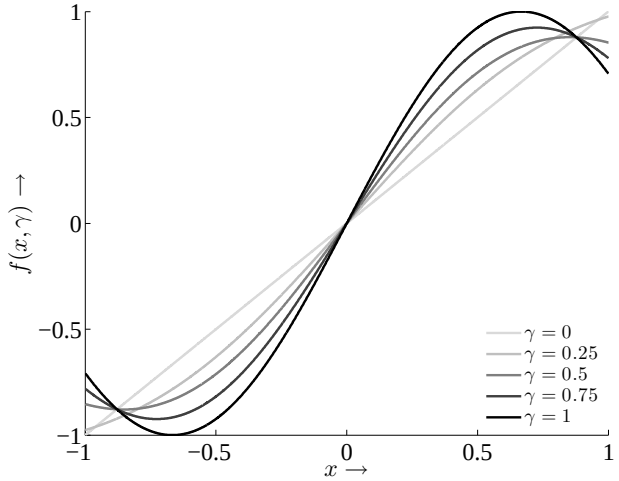


Figure 3.2: $f(x,\gamma) = \gamma\left(\sin\left(\frac{3\pi}{4}x\right)\right) + (1-\gamma)\,x.$

With increasing γ, the function $f(x,\gamma)$ changes from the linear function

$$l(x) = f(x,0) = x$$

to the nonlinear function

$$f(x) = f(x,1) = \sin\left(\frac{3\,\pi}{4}x\right) \quad .$$

■

Candidates for the desired optimal state-feedback controls $\underline{u}_k^*$, $k = 0, \ldots, N-1$, for the nonlinear system are determined as summarized in Algorithm 2, where the finite-horizon window to apply model predictive control is set to N steps. The current state $\underline{\hat{x}}_k$ is directly accessible and is employed as the new initial value. The continuation process initially solves the LQ control problem and yields the solution $\underline{U}^*(\gamma_0) = \underline{U}^*(0)$. The solution of step $i-1$ serves an initial guess $\underline{U}_{\text{init}}(\gamma_i)$ for a Newton method calculating $\underline{U}^*(\gamma_i)$ for increasing γ_i to satisfy condition (3.49). The desired state-feedback control $\underline{u}_k^*$ is given as the first entry of $\underline{U}^*(\gamma_t) = \underline{U}^*(1)$.

Algorithm 2 MPC: Application of the minimum principle

1: **procedure** MPC_INITIAL_SOLUTION
2: $N :=$ end of finite horizon window ▷ initialization
3: **for** $k = 0$ to ∞ **do**
4: $\underline{\hat{x}}_0 := \underline{\hat{x}}_k$
5: $\underline{U}^*(0) = \texttt{LQC}(\underline{\hat{x}}_0,\, N)$ ▷ initialization continuation process (LQ control)
6: **for** $i = 1$ to t **do** ▷ for increasing γ
7: $\underline{U}_{\text{init}}(\gamma_i) = \underline{U}^*(\gamma_{i-1})$ ▷ old solution as starting value
8: $\underline{U}^*(\gamma_i) = \texttt{Newton}(\underline{U}_{\text{init}}(\gamma_i))$ ▷ calculation of $\underline{U}^*$ via Newton method
9: **end for**
10: $\underline{u}_k^* := \underline{u}_0^*(1)$ ▷ optimal state-feedback control for current state
11: $\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{\hat{x}}_k, \underline{u}_k^*) + \underline{\boldsymbol{w}}_k$ ▷ time update
12: **end for**
13: **end procedure**

Remark 3.8 Reapplication of Algorithm 2 after each time step yields a closed-loop solution to the considered optimal control problem.

The Newton iteration is described in Algorithm 3 and works as follows. If

$$\|\underline{F}(\underline{U}_{\text{init}}(\gamma_i))\| > \varepsilon > 0 \quad ,$$

truncated Taylor series expansion around the initializing value $\underline{U}_{\text{init}}(\gamma_i)$ yields the condition

$$\underline{F}(\underline{U}_{\text{init}}(\gamma_i)) + \frac{\partial \underline{F}(\underline{U}_{\text{init}}\,(\gamma_i))}{\partial \underline{U}} \cdot \Delta\underline{U}_{\text{init}}(\gamma_i) = \underline{0} \quad .$$

Approximating the Jacobian $\frac{\partial \underline{F}(\underline{U}_{\text{init}}\,(\gamma_i))}{\partial \underline{U}}$, for example, by means of finite differences as proposed in [SB02], the desired update of the vector $\underline{U}_{\text{init}}(\gamma_i)$ is given as

$$\underline{U}_{\text{update}}(\gamma_i) = \underline{U}_{\text{init}}(\gamma_i) + \Delta\underline{U}_{\text{init}}(\gamma_i) \quad ,$$

where $\Delta\underline{U}_{\text{init}}(\gamma_i)$ solves

$$\frac{\partial \underline{F}(\underline{U}_{\text{init}}\,(\gamma_i))}{\partial \underline{U}} \Delta\underline{U}_{\text{init}}(\gamma_i) = -\underline{F}(\underline{U}_{\text{init}})(\gamma_i) \ .$$

This equation system may be solved by means of the QR-decomposition [SB02]. If

$$\|\underline{F}(\underline{U}_{\text{update}}(\gamma_i))\| \leq \varepsilon \ ,$$

the vector $\underline{U}^*(\gamma_i) := \underline{U}_{\text{update}}(\gamma_i)$ is returned. Otherwise, iteration yields the desired solution $\underline{U}^*(\gamma_i)$. Implementation details are given in Appendix F.

Algorithm 3 General Newton method

```
1:  function NEWTON(U_init(γ_i))
2:      tolF := ε > 0                                  ▷ initialization: threshold
3:      choose MaxIt                                   ▷ max. number of iterations
4:      U_0 := U_init(γ_i)
5:      for j = 0 to MaxIt do
6:          if ‖F(U_j)‖ ≤ ε then                       ▷ solution sufficiently good?
7:              U*(γ_i) := U_j
8:              return(U*(γ_i))                        ▷ yes: return solution
9:          else
10:             JacApprox = FiniteDifferences(F(U_j))  ▷ no: approximation of Jacobian
11:             U_j = solve(JacApprox · ΔU_j = −F(U_j))  ▷ solve linear equation system
12:             U_{j+1} = U_j + ΔU_j                   ▷ update of starting value for iteration
13:         end if
14:     end for
15:     print('not converged')                         ▷ Newton method not converged
16:     U*(γ_i) := U_MaxIt
17:     return(U*(γ_i))
18: end function
```

Remark 3.9 The initial value to the numerical algorithm is a good choice, since the initial guess is the assumed optimal solution of the previous step $i-1$ of the homotopy between the linear and the nonlinear system. In case of sufficiently small steps of the discrete curve follower and a sufficiently smooth value function, the Newton iteration yields the correct solution, since the initial guess is close to the solution.

Remark 3.10 The control sequence $(\underline{u}_0^*, \ldots, \underline{u}_{N-1}^*)$, which solves the nonlinear equation system (3.36), is assumed to be a suitable solution to the optimal control problem, if the value function is approximated by means of (3.5). This assumption is based on the uniqueness of the solution to the LQ control problem and the employment of the continuation process. Nevertheless, only a necessary condition is evaluated. Hence, there is no guarantee to obtain a global minimum in the unrestricted case, for example without restriction to convex sets.

3.2.3 Restriction of the State Space by Means of the Unscented Transformation

Discretization of the state and control spaces is a common approach to apply dynamic programming to technical applications, which usually have to treat continuous-valued problems. If the state space can be restricted in any way, for example, there is knowledge about improbable or impossible system states, the grid to be defined can be concentrated in the restricted part of the state space. Thus, the quality of the solutions of the DP algorithm increases.

Notation. In the following, the control variables solving the nonlinear equation system (3.36), will be denoted by $\underline{\hat{u}}_0, \ldots, \underline{\hat{u}}_{N-1}$, since the considered problem changes. That is, in contrast to the value function approximation (3.5) and the state propagation (3.7) to evaluate the value function, the original system function (3.1) and the original value function (3.3) are considered. Therefore, that control sequence is still known, but not assumed to be optimal anymore for the currently treated problems.

The sequence $(\underline{\hat{u}}_0, \ldots, \underline{\hat{u}}_{N-1})$ is employed to restrict the state space. To determine this restriction according to (3.37), the sequences of means and covariances of the successor states $\underline{\boldsymbol{x}}_{k+1}$ of the initial state $\underline{\hat{x}}_0$ are required. For an approximate calculation of these values, the unscented transformation and the known sequence $(\underline{\hat{u}}_0, \ldots, \underline{\hat{u}}_{N-1})$ are employed.

Review: Basic idea of the unscented transformation

According to [JU96], the unscented transformation is one method to obtain estimates of the mean and the covariance of a nonlinearly transformed random variable

$$\underline{\boldsymbol{y}} = \underline{f}(\underline{\boldsymbol{x}}) \ . \tag{3.51}$$

Instead of approximating the nonlinear function $\underline{f}$, which is for example done by the extended Kalman filter, the *density* of the random variable $\underline{\boldsymbol{x}}$ is approximated with a fixed number of few samples. These samples are individually transformed by means of the original function $\underline{f}$ of (3.51) as shown in Figure 3.3. Then, the mean and the covariance of the random variable $\underline{\boldsymbol{y}}$ can be determined approximately.

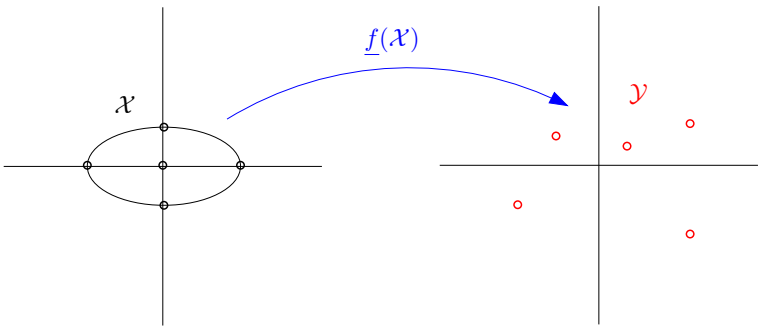


Figure 3.3: Principle of the unscented transformation. A specific set $\mathcal{X}$ of sampling points is nonlinearly transformed. The mean and the covariance of the resulting discrete distribution of the points approximate the corresponding values of the nonlinearly transformed random variable.

The UT employs a set $\mathcal{X}$ of sigma points $\mathcal{X}_i$ and corresponding weights $\omega_i,\ i = 0, \ldots, p$, according to

$$\mathcal{X} := \{\mathcal{X}_0, \ldots, \mathcal{X}_p, \omega_0, \ldots, \omega_p\} \quad .$$

The sigma points and the weights are selected, such that $\mathcal{X}$ captures mean and covariance of a random variable $\underline{\boldsymbol{x}}_k$. Then, the estimate of the mean of $\underline{\boldsymbol{y}}$ is accurate up to second order. The covariance of the nonlinearly transformed random variable $\underline{\boldsymbol{x}}_k$ can be determined approximately with first-order accuracy.

Remark 3.11 Accuracy of order k means that the Taylor series expansion of the nonlinear function $\underline{f}$ around the desired value is correct up to the k-th order term.

A more detailed introduction to the unscented transformation is given in Appendix D.

In the considered case, equation (3.51) is given by

$$\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{\boldsymbol{x}}_k, \hat{\underline{u}}_k) + \underline{\boldsymbol{w}}_k\,,\ k = 0, \ldots, N-1 \quad ,$$

that is the system equation (3.1), where the control $\hat{\underline{u}}_k$ is applied. The incorporation of the noise term is possible and treated in [XZC06].

To capture higher-order moments of the distribution of the random variable $\underline{\boldsymbol{x}}_k$ and to improve the accuracy of the UT, an additional parameter κ is introduced in [JU02], which ensures higher accuracy in the calculations. The weights and the $p = 2n + 1$ sigma points $\mathcal{X}_i^{(k)}$ around the mean value $\overline{\underline{x}}_k \in \mathcal{P}_k$ are set to

$$\omega_0 = \frac{\kappa}{n+\kappa} \tag{3.52}$$

$$\omega_j = \frac{1}{2(n+\kappa)}\,,\ j = 1, \ldots, p \tag{3.53}$$

$$\begin{aligned}
\mathcal{X}_0^{(k)} &= \overline{\underline{x}}_k \\
\mathcal{X}_i^{(k)} &= \overline{\underline{x}}_k - \left(\sqrt{(n+\kappa)\mathbf{C}_k^x}\right)_i\,,\ i = 1, \ldots, n \\
\mathcal{X}_i^{(k)} &= \overline{\underline{x}}_k + \left(\sqrt{(n+\kappa)\mathbf{C}_k^x}\right)_i\,,\ i = n+1, \ldots, 2n+1 \quad ,
\end{aligned}$$

where $\underline{x}_k \in \mathbb{R}^n$, and $\left(\sqrt{(n+\kappa)\mathbf{C}_k^x}\right)_i$ denotes the i-th column of a matrix square root[7] of $(n+\kappa)\mathbf{C}_k^x$.

In case of a Gaussian probability density function of the random variable $\underline{\boldsymbol{x}}_k$, the parameter κ is optimal for $\kappa = 3 - n$ [JU02]. Then, the weights ω_i for a scalar system are given by

$$\begin{aligned}
\omega_0 &= \frac{2}{3} \\
\omega_1 &= \frac{1}{6} \\
\omega_2 &= \frac{1}{6} \quad .
\end{aligned}$$

7 For $n \geq 2$, there exist infinitely many matrix square roots.

For $\kappa = \frac{1}{2}$, all weights ω_i are equal resulting from (3.52) and (3.53). Therefore, in the considered case, κ is set to $3 - n$ for the first two prediction steps. Since the initial value $\hat{\underline{x}}_0$ is known, its distribution is given by a Dirac function, and all sigma points coincide with $\hat{\underline{x}}_0$. Therefore, the first successor state $\underline{\boldsymbol{x}}_1$ possesses a Gaussian density function, which depends on the noise covariance $\boldsymbol{\Sigma}_w$. With increasing prediction horizon, the density functions of the successor states are not Gaussian in general, and the corresponding probability density functions become more difficult and possibly multimodal. Therefore, the parameter κ is chosen as a kind of homotopy parameter[8] according to

$$\kappa = \begin{cases} 3 - n & \text{for } k = 1, 2 \ , \\ 3 - n - \frac{|3-n|}{N} k + \frac{1}{2} & \text{for } k > 2 \ . \end{cases}$$

for the k-th prediction step of an N-step horizon. κ approaches $\frac{1}{2}$, such that finally all sigma points $\mathcal{X}_i^{(N)}$ are equally weighted to capture more information of the distribution in regions that are not close to the mean value. This choice is a contribution to the possible multi modality of the unknown densities of the successor states $\underline{\boldsymbol{x}}_{k+1}$, $k \geq 2$.

3.2.4 Determination of the Grid Points for the Value Function Interpolation

After the determination of the set $\mathcal{P}_k$ as described in (3.37), a modified set

$$\mathcal{G}_k^{init} = \left\{ \underline{x}_k^{(0)}, \ldots, \underline{x}_k^{(p)} \right\} \tag{3.54}$$

is determined as follows to incorporate important properties of the considered system. For sufficiently large k, one of the grid points is substituted by the desired target point $\underline{c}$ given by the value function in the DP algorithm, that is Algorithm 1.[9] The grid point to be substituted is the nearest neighbor of the target point. This modification assures exact consideration of the desired terminal states. To keep the symmetry of the set of grid points, the symmetric equivalents of the substituted grid point are exchanged, too.

Remark 3.12 It is important to note that the sets $\mathcal{G}_k^{init}$ cover the same parts of the state space as the sets $\mathcal{P}_k$, if no extremal point of $\mathcal{P}_k$ is replaced. Furthermore, the number of points does not change. Based on simulations, the employment of the modified grid $\mathcal{G}_k^{init}$ is assumed to provide better results, since the desired terminal point is included explicitly as a knot of the subsequent spline interpolation.

Example 3.2: Possible restriction of the state space

In Figure 3.4, a possible restriction of the state space is given for a scalar system and a five-step horizon.

Around the estimates of the mean values of the successor states of the known initial value $\hat{x}_0$, the state space is restricted depending on the estimated covariance of the successor states. ■

8 similar to Section 3.2.2

9 Depending on the concrete application, the first parameter k, for which the target point is included, has to be chosen heuristically.

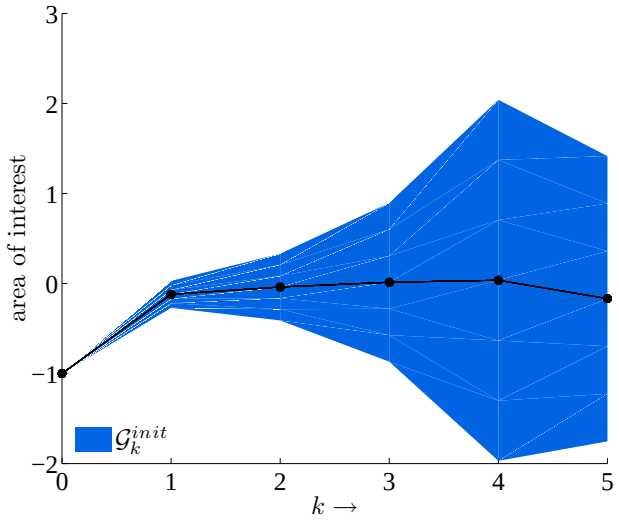


Figure 3.4: Possible restriction of the state space. Around the predicted mean values of the successor states of $\hat{x}_0$, the state space is restricted depending on the predicted covariances of the successor states.

3.2.5 Interpolation of the Value Function by Means of Piecewisely Defined Cubic Splines

In the following, the basic concept is discussed, how to improve the control sequence given by the solution of the nonlinear equation system (3.36) by means of piecewise cubic spline interpolation. Again, dynamic programming is the basis for the algorithm, which is performed within the range of the sets $\mathcal{G}_k^{init},\ k = 0, \ldots, N$.

Motivation to Employ Piecewisely Defined Cubic Splines

The motive for an interpolation is that the approximating function is simpler to compute than the approximated function. The classical choice for an approximation scheme is a single polynomial, sometimes of high degree, to approximate the function. The employment of a single high-degree polynomial creates a global interpolant, that is, it applies to the entire interval. The problem is that a change in one point in the approximation affects the quality of the approximation at all points, even those that are distant from the point of change.

To minimize this behavior, piecewisely defined polynomials of lower degree are used to approximate the original function. The classical examples for such low-degree polynomials are linear, quadratic, and cubic polynomials.

Piecewise polynomials, such as spline functions, are more flexible in following the variations of a function and are also more local in their approximation. With increasing degree, the interpolating functions become less local, that is, changes within one subinterval do not only influence the adjacent subintervals but others, too [FM97].

In this thesis, piecewise interpolation by means of cubic splines is chosen. This approximation scheme minimizes the oscillation behavior of the interpolant [dB78]. Since a common assumption on the value function is the twice continuous differentiability, the interpolating function

should also possess these properties. Therefore, the piecewise interpolation requires at least polynomials of degree three.

Application of the Spline Interpolation in the Dynamic Programming Algorithm

To apply dynamic programming with value function approximation, the first step is to interpolate the terminal cost (3.2), which is possible, since the terminal cost $J_N(\underline{x}_N)$ is independent of the control variable. Thus, an analytic approximation of $J_{k+1}(\underline{x}_{k+1})$ in the range of $\mathcal{G}_{k+1}^{init}$, which is defined in (3.54), is assumed to be given. An approximate analytic description of $J_k(\underline{x}_k)$ in the range of $\mathcal{G}_k^{init}$ can be obtained by means of spline interpolation. After the determination of the value function $J_k(\underline{x}_k^{(i)})$ at the grid points $\underline{x}_k^{(i)}$, the tuples

$$\left\{\left(\underline{x}_k^{(i)}, J_k(\underline{x}_k^{(i)})\right)\right\}_{i=0}^{p}$$

are employed to provide an approximation $J_k^{spline}(\underline{x}_k)$ of the true function $J_k(\underline{x}_k)$ around $\overline{\underline{x}}_k$, where the spread depends on the covariance $\mathbf{C}_k^x$. Therefore, it remains to derive the values $J_k(\underline{x}_k^{(i)})$ for all $\underline{x}_k^{(i)} \in \mathcal{G}_k^{init}$. The recursive calculation of an approximation of $J_k(\underline{x}_k)$ within the dynamic programming algorithm is described in Algorithm 4, which employs the knowledge of the control sequence $(\hat{u}_0, \ldots, \hat{u}_{N-1})$ for the current horizon window.

1. The algorithm is initialized with the spline interpolation of the terminal cost.
2. For $\underline{x}_k^{(i)} \in \mathcal{G}_k^{init}$ the set $\mathcal{U}_k^{(i)}$ is determined, which contains all $\underline{u}_k^{(i,j)}$ with

   $$\underset{\underline{w}_k}{\mathrm{E}}[\underline{f}(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)}) + \underline{\boldsymbol{w}}_k] = \underline{f}(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)}) = \underline{x}_{k+1}^{(j)}$$

   for fixed $\underline{x}_k^{(i)}$ and all successor states $\underline{x}_{k+1}^{(j)} \in \mathcal{G}_{k+1}^{init}$. This means, a set $\mathcal{U}_k^{(i)}$ of control variables is desired mapping $\underline{x}_k^{(i)} \in \mathcal{G}_k^{init}$ onto each element $\underline{x}_{k+1}^{(j)} \in \mathcal{G}_{k+1}^{init}$ under $\underline{f}$. A scalar example is depicted in Figure 3.5.

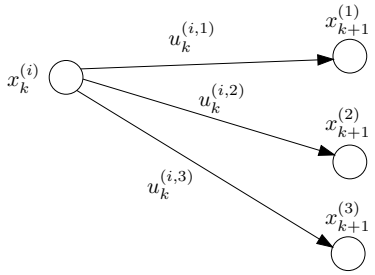


Figure 3.5: Scalar example: determination of the set $\mathcal{U}_k^{(i)}$.

3. The value $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ can be obtained in several ways. On the one hand, numerical integration yields a correct solution. On the other hand, approximate solutions can be employed to circumvent the computational expenses of numerical integration. Therefore, a tradeoff between exactness and calculation time has to be found. One possible candidate

Algorithm 4 Update of the state-feedback control

function VALUE_FUNCTION_INTERPOLATION($\hat{\underline{u}}_0, \ldots, \hat{\underline{u}}_{N-1}, \mathcal{G}_0^{init}, \ldots, \mathcal{G}_N^{init}$)
  $J_N^{spline}(\underline{x}_k) \approx \texttt{spline}\left(\left\{\left(\underline{x}_N^{(i)}, g_N(\underline{x}_N^{(i)})\right)\right\}_{i=0}^{p}\right)$ ▷ initialization
  for $k = N-1$ to 0 **do** ▷ for the current horizon window
    $J_{k+1} := J_{k+1}^{spline}$
    for $i = 0$ to p **do** ▷ for all grid points
      $\mathcal{U}_k^{(i)} := \emptyset$
      for $j = 0$ to p **do**
        $\mathcal{U}_k^{(i)} = \left\{\underline{u}_k^{(i,j)} : \mathrm{E}_{\underline{w}_k}[\underline{f}(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)}) + \underline{\boldsymbol{w}}_k] = \underline{x}_{k+1}^{(j)}\right\} \cup \mathcal{U}_k^{(i)}$
        determine $\underset{\underline{w}_k}{\mathrm{E}}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$
        $V_k(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)}) := g_k(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)}) + \underset{\underline{w}_k}{\mathrm{E}}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ ▷ cost function
      end for
      $V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k) := \texttt{spline}\left(\left\{\left(\underline{u}_k^{(i,j)}, V_k(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)})\right)\right\}_{j=0}^{p}\right)$ ▷ interp. cost function
      $J_k(\underline{x}_k^{(i)}) = \min_{\underline{u}_k} V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k)$ ▷ minimization
      $\underline{u}_k^* = \arg\min_{\underline{u}_k} V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k)$ ▷ optimal state-feedback control
    end for
    $J_k^{spline}(\underline{x}_k) \approx \texttt{spline}\left(\left\{\left(\underline{x}_k^{(i)}, J_k(\underline{x}_k^{(i)})\right)\right\}_{i=0}^{p}\right)$ ▷ interp. value function
  end for
  $\texttt{return}(\underline{u}_0^*)$
end function

for an approximate solution is the unscented transformation [JU96]. In Figure 3.6, the calculation of the value $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ for one successor state is shown graphically.

4. Then, the cost function, that is, the expected cost-to-go from the current state to the terminal state via one specific successor state, is determined by

$$V_k(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)}) = g_k(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)}) + \underset{\underline{w}_k}{\mathrm{E}}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})] \ .$$

   In Figure 3.7, the corresponding scalar example is given. It is important to mention that this cost is not minimized yet.

5. After the computation of the expected cost-to-go for all successor states of $\underline{x}_k^{(i)}$, spline interpolation of $V_k(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)})$ with knots $\underline{u}_k^{(i,j)}$, $j = 0, \ldots, p$, yields an analytic approximation $V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k)$ of the cost function in the range of the controls $\underline{u}_k^{(i,\cdot)}$ applied at $\underline{x}_k^{(i)}$. The graphical equivalent for the scalar example is given in Figure 3.8.

6. Minimization of $V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k)$ with respect to $\underline{u}_k$ yields the minimal expected cost-to-go $J_k(\underline{x}_k^{(i)})$.

7. The optimal state-feedback control for the considered state $\underline{x}_k^{(i)}$ is given by the minimizing control variable of the cost function $V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k)$.

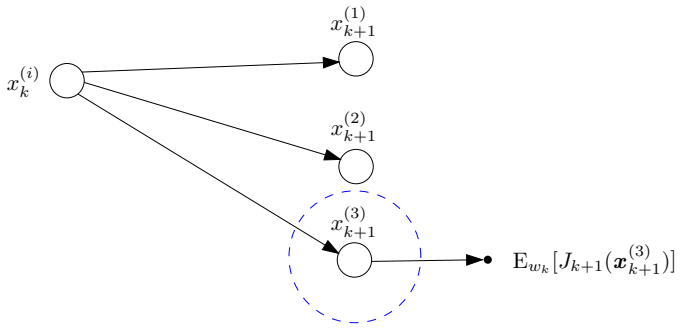


Figure 3.6: Scalar example: computation of $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(3)})]$.

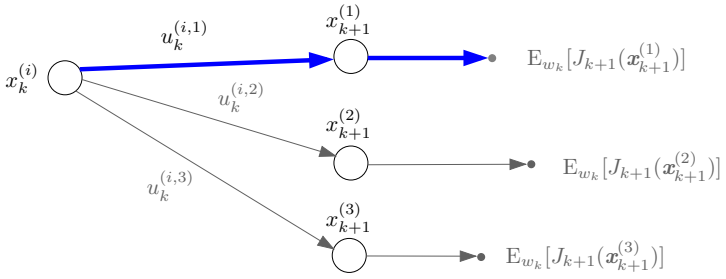


Figure 3.7: Scalar example: computation of the expected cost-to-go $V_k(x_k^{(i)}, u_k^{(i,1)}) = g_k(x_k^{(i)}, u_k^{(i,1)}) + \underset{w_k}{\mathrm{E}}[J_{k+1}(\boldsymbol{x}_{k+1}^{(1)})]$.

8. Finally, an approximate analytic description of J_k is given by interpolating $J_k(\underline{x}_k^{(i)})$ along the knots $\underline{x}_k^{(i)} \in \mathcal{G}_k^{init}$, $i = 0, \ldots, p$. In Figure 3.9, this last step is depicted for the scalar example.

Remark 3.13 It is important to note that in Algorithm 4 the value function (3.3) is not approximated by means of Taylor series expansion according to (3.5) anymore. The new approximation is based on the adaptive discretization of the state space according to (3.37), its modification according to (3.54), and the interpolation of the cost with respect to the control variable in line 12 of Algorithm 4, which yields a continuous-valued control problem to be

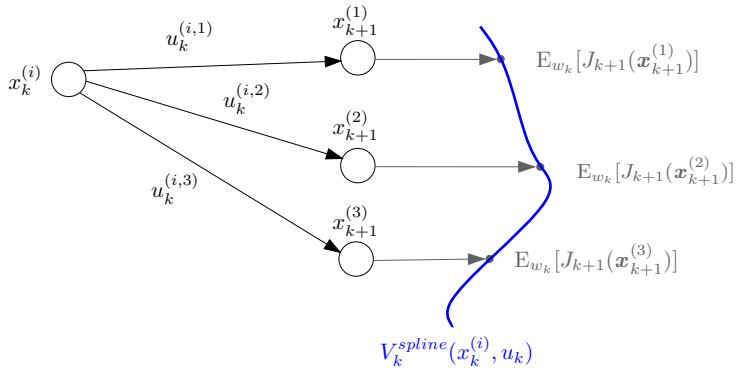


Figure 3.8: Scalar example: interpolation of the cost function with respect to u_k.

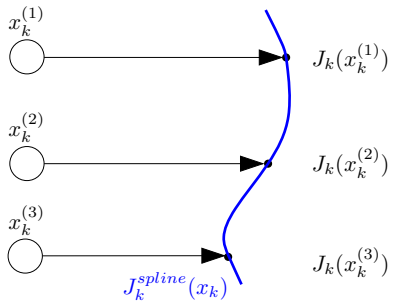


Figure 3.9: Scalar example: interpolation of the value function with respect to x_k.

solved. Furthermore, the spline interpolation along the knots $\underline{x}_k^{(i)}$ in line 16 yields a continuous approximation of the value function within the area of interest.

The resulting updated control vector $\underline{u}_0^*$ for the current horizon window is expected to be closer to the true optimal solution than $\underline{\hat{u}}_0$ given by Algorithm 2. This assumption is based on the fact that the sequence $(\underline{\hat{u}}_0, \ldots, \underline{\hat{u}}_{N-1})$ in the determination of the sets $\mathcal{P}_k$ and the influence of noise according to Remark 3.13 are explicitly incorporated in the calculation of the updated control.

In the following, several steps of Algorithm 4 will be discussed in more detail to motivate the chosen solutions.

Calculation of $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$

In line 9 of Algorithm 4, the calculation of the expectation value $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ is required to apply dynamic programming. Since the state $\underline{x}_k$ is fixed in each calculation step of Algorithm 1, the density function of the successor state $\underline{\boldsymbol{x}}_{k+1}^{(j)}$ only depends on the probability density function of the noise term in (3.1) as explained in the following.

Review: Probability density of the successor state

For a given system function

$$\underline{\boldsymbol{x}}_{k+1} = \underline{f}(\underline{\boldsymbol{x}}_k, \underline{u}_k) + \underline{\boldsymbol{w}}_k \tag{3.55}$$

with probability density functions P_k^x, P_k^w for $\underline{\boldsymbol{x}}_k$, $\underline{\boldsymbol{w}}_k$, respectively, the density P_{k+1}^x of the successor state $\underline{\boldsymbol{x}}_{k+1}$ can be calculated as follows. Regarding (3.55), the conditional density $\mathrm{P}(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k, \underline{w}_k)$ is given by

$$\mathrm{P}(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k, \underline{w}_k) = \delta(\underline{x}_{k+1} - \underline{f}(\underline{x}_k, \underline{u}_k) - \underline{w}_k) \ , \tag{3.56}$$

where $\delta(\,\cdot\,)$ denotes the Dirac function. Exploiting Bayes' Law, the desired function P_{k+1}^{x} can be written as the marginalization

$$\begin{aligned}\mathrm{P}_{k+1}^{x}(\underline{x}_{k+1}) &= \int_{\mathbb{R}^m}\int_{\mathbb{R}^n} \mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k)\,\mathrm{P}_k^{x,u}(\underline{x}_k,\underline{u}_k)\,\mathrm{d}\underline{x}_k\,\mathrm{d}\underline{u}_k \\ &= \int_{\mathbb{R}^m}\int_{\mathbb{R}^n} \mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k)\,\mathrm{P}_k^{x}(\underline{x}_k|\underline{u}_k)\,\mathrm{P}_k^{u}(\underline{u}_k)\,\mathrm{d}\underline{x}_k\,\mathrm{d}\underline{u}_k \ . \end{aligned} \tag{3.57}$$

Since $\underline{\boldsymbol{x}}_k$ and $\underline{\boldsymbol{u}}_k$ are independent random variables, equation (3.57) is simplified to

$$\mathrm{P}_{k+1}^{x}(\underline{x}_{k+1}) = \int_{\mathbb{R}^m}\int_{\mathbb{R}^n} \mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k)\,\mathrm{P}_k^{x}(\underline{x}_k)\,\mathrm{P}_k^{u}(\underline{u}_k)\,\mathrm{d}\underline{x}_k\,\mathrm{d}\underline{u}_k \ ,$$

where $\mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k)$ is given by the marginalization

$$\begin{aligned}\mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k) &= \int_{\mathbb{R}^n} \mathrm{P}(\underline{x}_{k+1},\underline{w}_k|\underline{x}_k,\underline{u}_k)\,\mathrm{d}\underline{w}_k \\ &= \int_{\mathbb{R}^n} \mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k,\underline{w}_k)\,\mathrm{P}_k^{w}(\underline{w}_k)\,\mathrm{d}\underline{w}_k \ , \end{aligned} \tag{3.58}$$

where again Bayes' law has been applied. Insertion of (3.56) into (3.58) yields

$$\mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k) = \int_{\mathbb{R}^n} \delta(\underline{x}_{k+1} - \underline{f}(\underline{x}_k,\underline{u}_k) - \underline{w}_k)\,\mathrm{P}_k^{w}(\underline{w}_k)\,\mathrm{d}\underline{w}_k \ .$$

Then, this equation can be simplified through the properties of the Dirac function, resulting in

$$\mathrm{P}(\underline{x}_{k+1}|\underline{x}_k,\underline{u}_k) = \mathrm{P}_k^{w}(\underline{x}_{k+1} - \underline{f}(\underline{x}_k,\underline{u}_k)) \ .$$

Hence, the density of the successor state of a one-step prediction depends on the noise density [HS05], since

$$\mathrm{P}_{k+1}^{x}(\underline{x}_{k+1}) = \int_{\mathbb{R}^m}\int_{\mathbb{R}^n} \mathrm{P}_k^{w}(\underline{x}_{k+1} - \underline{f}(\underline{x}_k,\underline{u}_k))\,\mathrm{P}_k^{x}(\underline{x}_k)\,\mathrm{P}_k^{u}(\underline{u}_k)\,\mathrm{d}\underline{x}_k\,\mathrm{d}\underline{u}_k \ . \tag{3.59}$$

In the considered case, the state is directly accessible, that is, the state is known with certainty. Moreover, the control variable is treated as a deterministic variable. Therefore, the functions P_k^x and P_k^u in (3.59) can be replaced with Dirac functions, such that the integrals vanish.

According to line 9 of Algorithm 4, the DP algorithm employs a one-step prediction from the state $\underline{x}_k^{(i)}$ to the successor state $\underline{\boldsymbol{x}}_{k+1}^{(j)}$ to determine the expected cost-to-go

$$V_k(\underline{x}_k^{(i)},\underline{u}_k^{(i,j)}) = g_k(\underline{x}_k^{(i)},\underline{u}_k^{(i,j)}) + \mathop{\mathrm{E}}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})] \ .$$

Therefore, the value $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ is a nonlinear transformation of the random variable $\underline{\boldsymbol{x}}_{k+1}^{(j)}$ by means of the function J_{k+1}, where the density of $\underline{\boldsymbol{x}}_{k+1}^{(j)}$ is determined by the density of $\underline{\boldsymbol{w}}_k$.

Review: Nonlinear transformation of random variables — expectation value

Let $\underline{\boldsymbol{x}}$ be a random variable with a given probability density P^x. Moreover, a nonlinear transformation of $\underline{\boldsymbol{x}}$ is given by

$$\underline{\boldsymbol{y}} = \underline{f}(\underline{\boldsymbol{x}}) \ .$$

Then, the expected value of $\underline{\boldsymbol{y}}$ is given by

$$\mathrm{E}[\underline{\boldsymbol{y}}] = \int_{\mathbb{R}^n} \underline{f}(\underline{x}) \, \mathrm{P}^x(\underline{x}) \, \mathrm{d}\underline{x} \ .$$

The expectation value $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ can be exactly determined by means of numerical integration, that is,

$$\mathop{\mathrm{E}}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})] = \int_{\mathbb{R}^n} J_{k+1}(\underline{x}_{k+1}) \, \mathrm{P}_{k+1}^x(\underline{x}_{k+1}) \, \mathrm{d}\underline{x}_{k+1} \ .$$

This calculation is computationally demanding, why approximate solutions are often employed. This can be done by means of the unscented transformation (UT), which yields estimates of mean and covariance of nonlinearly transformed random variables. The UT is superior to the solutions provided by the extended Kalman filter [JU97, vdMDdFW00, JUDW00, JU04, XZC06, CHL05]. Nevertheless, the unscented transformation is not employed in this case because of two reasons. Firstly, the accuracy of the whole algorithm shall not suffer from additional errors resulting from the calculation of the considered expectation value, which complicates the analysis of the algorithm. Secondly, the proposed algorithm is compared to other solutions, which do not determine the value $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ by means of the UT. The differences in the resulting value functions originating from both ways of computing $\mathrm{E}_{\underline{w}_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ are described in Appendix D by means of an example.

Comments on the Interpolation

As described in Section 2.2.6, dynamic programming is computationally not tractable for continuous-valued problems in general. Moreover, analytical solutions to the optimal control problem cannot be found. Therefore, approximations are inevitable. The considered approximation of the value function by means of spline interpolation is one possible approximation. Depending on the smoothness of the value function, the number of grid points has to be chosen appropriately. Since the value function is unknown in general, heuristics are employed. Another parameter to be selected carefully is the placement of the grid points. It seems to be useful to choose a higher concentration in the center area of the considered part of the state space. In this region, the interpolant is assumed to be close to the true, but still unknown, value function. The chosen parameters to determine the grid are given in Appendix F.

The value function[10] is employed in the dynamic programming part of Algorithm 4 in line 10 to determine the cost-to-go for the considered state. This cost function is minimized in the next step of the algorithm. Higher-order polynomials suffer from non-closed solutions to the minimization problem, even if only the necessary condition

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = 0$$

is evaluated. This is due to the fact that there is no general closed-form algorithm to determine the roots of a polynomial of degree five or higher [Bos06]. Therefore, cubic functions seem to be a suitable choice for the piecewise polynomial interpolation of the value function.

Minimization

The calculation of the cost function $V_k(\underline{x}_k^{(i)}, \underline{u}_k^{(i,j)})$ in line 10 of Algorithm 4, which is not minimized yet, yields the expected cost-to-go from the state $\underline{x}_k^{(i)}$ via one specific terminal state to the terminal time. The subsequent interpolation with respect to the control variables in line 12 yields a twice continuously differentiable function in $\underline{u}_k$ to be minimized. Because of the piecewise cubic spline interpolation, the general minimization problem can be solved analytically. For each spline piece, the absolute minimum can be obtained by evaluating the necessary minimum condition

$$\frac{\partial V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k)}{\partial \underline{u}_k} = \underline{0}^{\mathrm{T}} \quad . \tag{3.60}$$

Since (3.60) is at most quadratic, the solution can be obtained easily. A second pair of candidates are the borders of the interval, in which the cubic spline interpolates the value function. Comparison of the evaluation of the function $V_k^{spline}(\underline{x}_k^{(i)}, \underline{u}_k)$ for all candidates yields the minimum of the considered spline piece. Execution of this procedure for all intervals and final comparison of the minima of all spline pieces yields the desired global minimum and the function $J_k(\underline{x}_k^{(i)})$.

10 better: its interpolant

CHAPTER 4

Simulation Results

In this chapter, the algorithms introduced in Chapter 3 are evaluated by means of two scalar example systems. Moreover, an alternative calculation to numerical integration to determine $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$ in Algorithm 4 is discussed. Implementation details are given in Appendix F.

4.1 Considered Systems and Setting

In the following, two different scalar systems are considered, which are given by

$$\boldsymbol{x}_{k+1} = \sin(q\,\boldsymbol{x}_k) + u_k + \boldsymbol{w}_k \ , \tag{4.1}$$

$$\boldsymbol{x}_{k+1} = \sqrt{2}\,\sin\Big(\boldsymbol{x}_k + \frac{\pi}{4}\Big) + \frac{\boldsymbol{x}_k}{2} - 1 + u_k + \boldsymbol{w}_k \ , \tag{4.2}$$

where $x_k,\, u_k,\, w_k \in \mathbb{R}$ and $q = \frac{3\pi}{4}$. For the first considered system (4.1), the simulations are performed for starting values $\hat{x}_0 \in \mathcal{X}_1 := \{-1, -0.8, \ldots, 1\}$ and for noise with standard deviations $\sigma \in \mathcal{S}_1 := \{0.1, 0.2, 0.3\}$. The simulations of the second system (4.2) use $\hat{x}_0 \in \mathcal{X}_2 := \{-10, -8, \ldots, 10\}$ and $\sigma \in \mathcal{S}_2 := \{1, 3, 5\}$.

To apply model predictive control, the decision-making horizon window is set to $N = 5$ steps. That is, the optimal state-feedback control is determined for a horizon of five time steps. The simulations have been carried out for ten time steps.

4.2 Application of the Minimum Principle

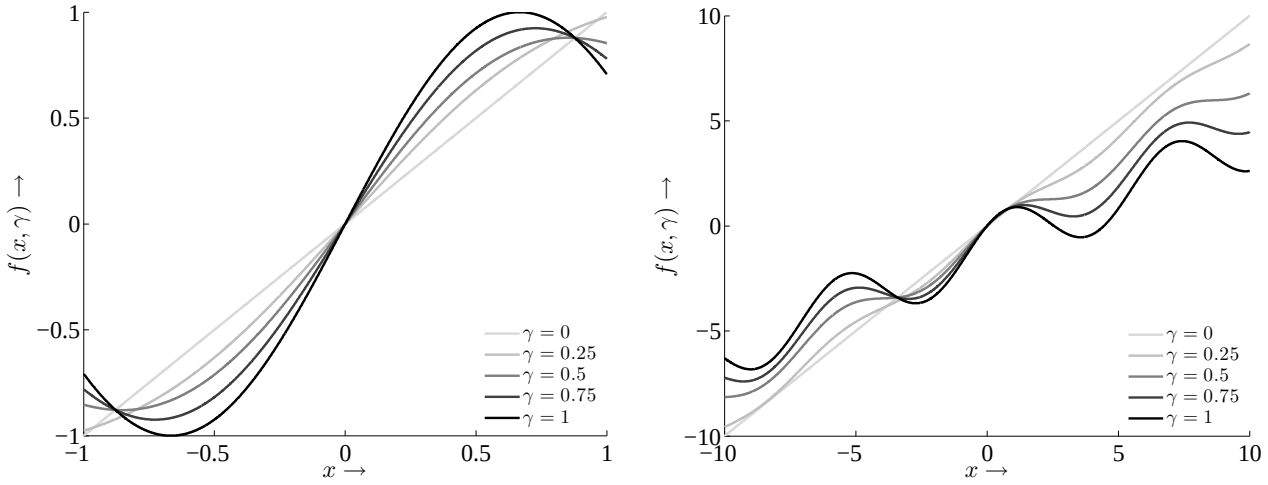
In this section, the control sequence resulting from Algorithm 2 is analyzed. To apply the continuation process of Section 3.2.2 to solve the nonlinear equation system (3.36), the parameterized systems are given by

$$\boldsymbol{x}_{k+1}(\gamma) = \gamma\,\sin(q\,\boldsymbol{x}_k(\gamma)) + (1-\gamma)\,\boldsymbol{x}_k(\gamma) + u_k(\gamma) + \boldsymbol{w}_k \tag{4.3}$$

and

$$\boldsymbol{x}_{k+1}(\gamma) = \gamma\left(\sqrt{2}\,\sin\Big(\boldsymbol{x}_k(\gamma) + \frac{\pi}{4}\Big) + \frac{\boldsymbol{x}_k(\gamma)}{2} - 1\right) + (1-\gamma)\,\boldsymbol{x}_k(\gamma) + u_k(\gamma) + \boldsymbol{w}_k \ , \tag{4.4}$$

respectively. In Figure 4.1(a) and Figure 4.1(b), the transformations of the linear system $f(x,0) = x$ into the desired nonlinear functions $f(x,1)$ of the systems (4.1) and (4.2) are depicted.



(a) $f(x,\gamma) = \gamma\left(\sin\left(\frac{3\pi}{4}x\right)\right) + (1-\gamma)\,x.$

(b) $f(x,\gamma) = \gamma\left(\sqrt{2}\,\sin\left(x+\frac{\pi}{4}\right) + \frac{x}{2} - 1\right) + (1-\gamma)\,x.$

Figure 4.1: Considered functions depending on x and a homotopy parameter γ. The functions depend on a homotopy parameter γ, which transforms the linear system $f(x,0) = x$ into the desired nonlinear systems $f(x,1)$.

The solution of each homotopy step is employed as the initial guess of the solution of the next homotopy step. The system state is propagated by means of (3.7).

The value functions for the parameterized systems (4.3) and (4.4) are given by

$$\begin{aligned} J_N(x_N) &= \frac{1}{2}(x_N - c)^2 \ , \\ J_k(x_k,\gamma) &= \frac{1}{2}\left((x_k - c)^2 + a\,(u_k^*(\gamma))^2\right) + \underset{w_k}{\mathrm{E}}\left[J_{k+1}\left(\boldsymbol{x}_{k+1}(\gamma)\right)\right] \ , \end{aligned} \tag{4.5}$$

where $a = 2$. The desired target state $c = 0$ is an unstable equilibrium point for both considered systems.

With

$$h_{k+1} := \frac{\mathrm{d}^2 J_{k+1}}{\mathrm{d}x_{k+1}^2} \ ,$$

the approximated value functions according to (3.5) are given by

$$\begin{aligned} J_N(x_N) &= (x_N - c)^2 \ , \\ J_k(x_k,\gamma) &= \frac{1}{2}\left((x_k - c)^2 + a\,(u_k^*(\gamma))^2\right) + J_{k+1}\left(x_{k+1}(\gamma)\right) + \frac{1}{2}\,\sigma^2 h_{k+1}\left(x_{k+1}(\gamma)\right) \ . \end{aligned} \tag{4.6}$$

The costate recursion (3.12) and the Hamilton function (3.16) yield the necessary minimum condition

$$\frac{\partial H_k(x_k^*(\gamma), p_{k+1}(\gamma), u_k^*(\gamma), \boldsymbol{w}_k)}{\partial u_k} = a\,u_k^*(\gamma) + p_{k+1}(\gamma) = 0$$

along the optimal state and control trajectories and, therefore, an analytical solution

$$u_k^*(\gamma) = -a^{-1}\, p_{k+1}(\gamma) \ , \tag{4.7}$$

which is a candidate for the optimal state-feedback control. Therefore, (4.7) can be employed to verify the numerical solution of the algorithm.

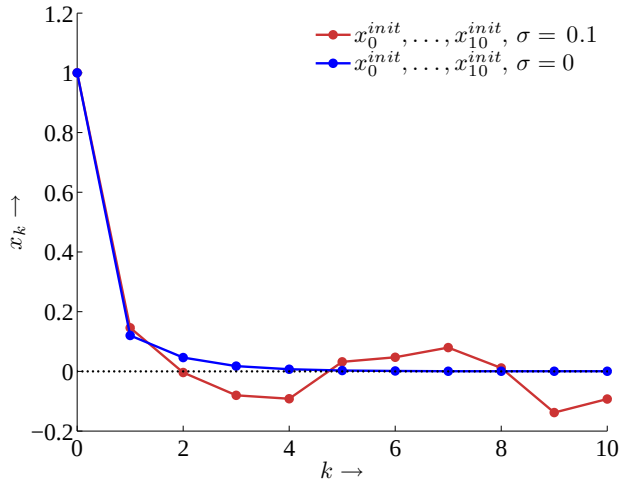
Remark 4.1 In contrast to an algorithm, which does not employ the continuation process, the simulations of the considered systems according to Algorithm 2 almost always converged and provided correct results with respect to (4.7). Hence, the additional expenses arising from the continuation are justified.

Notation. Since the results of Algorithm 2 serve as initialization of Algorithm 4, in the following, the states and the control variables are denoted by x_k^{init} and u_k^{init}, respectively. Moreover, $J_{\sigma=i}^{init}$ denotes the value function approximated by (4.6) for $\gamma = 1$ and $k = 0$. The system is affected by noise with standard deviation σ.

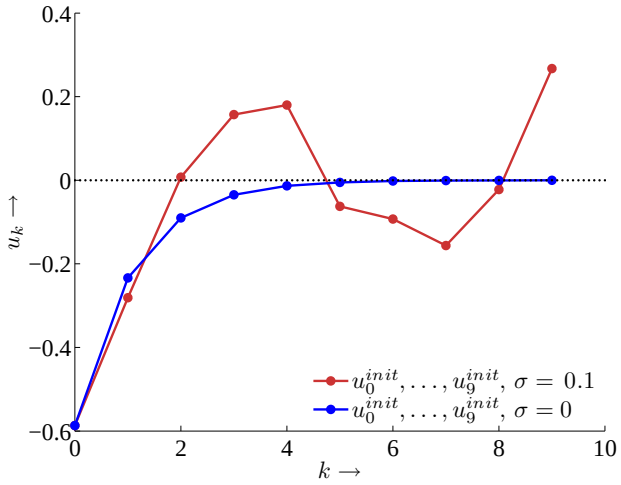
4.2.1 Noise Influence

In case of the first considered system, (4.1) reveals that the influence of noise is as strong as the influence of the system input u_k. Moreover, the sine, as the nonlinear part of the system function, is bounded and attains values within the interval $[-1, 1]$. Thus, even the influence of noise with standard deviation $\sigma = 0.1$ can be regarded as a relatively strong influence on the considered system. This fact is stressed by Figure 4.2 for one example simulation with $\hat{x}_0 = 1$, where the deviations of the state and control trajectories can be seen easily. In Figure 4.2(a), the deviations of two state trajectories are depicted. The differences reveal the strength of the noise influence on system (4.1). In case of the deterministic system, that is $\sigma = 0$, the state trajectory quickly converges toward the target point $c = 0$. In case of the noise-affected system with noise standard deviation $\sigma = 0.1$, the state trajectory oscillates around zero. The trajectories of the corresponding state-feedback controls in Figure 4.2(b) behave similarly. In case of the noise-affected system, the state-feedback control tries to lead the system state toward the target point, which explains the peaks of the trajectory, when the system state significantly deviates from the target point.

To emphasize the influence of the noise on system (4.2), in Figure 4.3 an example state trajectory and the corresponding control trajectory are depicted. Even for the smallest simulated noise influence, that is $\sigma = 1$, and the initial value $\hat{x}_0 = 10$, the differences between the trajectories are again obvious and due to the noise term. Therefore, both simulated systems suffer from relatively strong noise disturbances, which motivates the consideration of noise in the design of the controller.

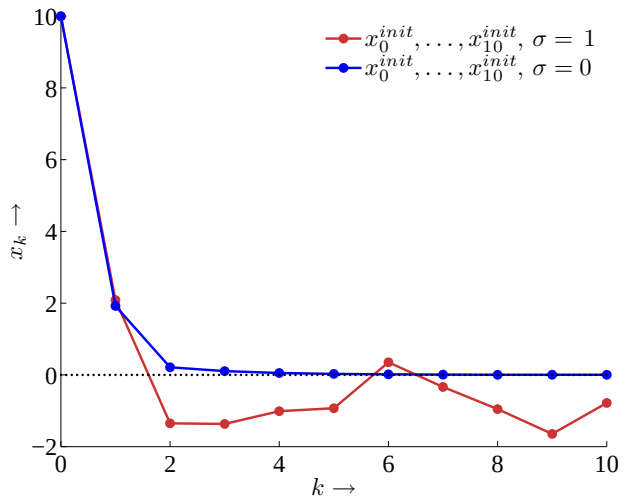


(a) Example state trajectories. Even in the case of small noise influence, the deviations between the noise-affected trajectory and the trajectory of the deterministic system can be seen easily.

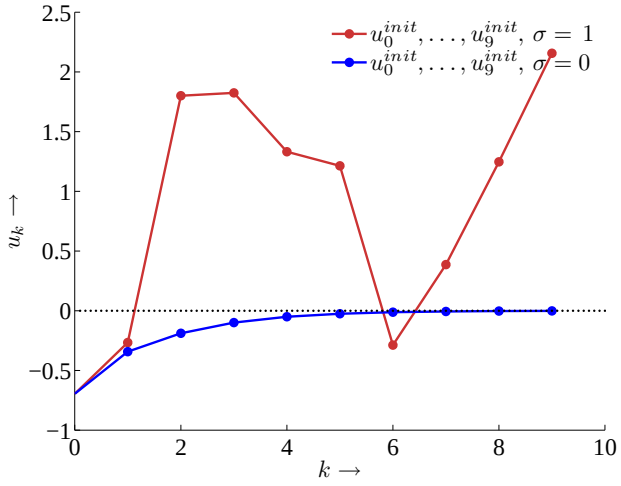


(b) Example control trajectories. Due to the noise influence, both trajectories differ significantly. The state-feedback control at each time step is chosen to lead the system state toward the desired target point.

Figure 4.2: Example state and control trajectories for $\sigma = 0$ and $\sigma = 0.1$ for system (4.1). The system state is shifted significantly due to the relatively strong noise influence on the system. The corresponding system input is chosen to lead the state toward the target point.



(a) Example state trajectory for system (4.2). Even for $\sigma = 1$, which is the smallest considered noise standard deviation, the noise influence on the system can be seen easily.



(b) Example control trajectories for system (4.2). To modify the state trajectory in Figure 4.3(a), the control decision has to be adapted. The significant differences between one example control trajectory of the system influenced by noise with standard deviation $\sigma = 1$ and the deterministic control sequence can be seen easily.

Figure 4.3: Example state and control trajectories for system (4.2). Comparing the trajectories for the deterministic system, that is $\sigma = 0$, and the system affected by noise with standard deviation $\sigma = 1$, the differences become obvious immediately. Even for this noise influence, the deviations are significant.

4.2.2 Monte-Carlo Simulation

For $\sigma > 0$, the arising cost of the simulation changes with each run. A Monte-Carlo simulation provides an approximate upper bound $J_{\sigma=i}^{MC\_init}$ of the true value function depending on the noise standard deviation $\sigma = i$ by calculating the arithmetic mean of all arisen cost of M simulations starting from $\hat{x}_0 \in \mathcal{X}_j$. That is, for $\hat{x}_0 \in \mathcal{X}_j$ and $\sigma \in \mathcal{S}_j$, $j = 1, 2$,

$$J_{\sigma=i}^{MC\_init}(\hat{x}_0) := \frac{1}{M} \sum_{t=1}^{M} J_{\sigma=i}^{\mathrm{sim}_t}(\hat{x}_0) \ , \tag{4.8}$$

where the real state and the corresponding control sequences are known after the simulation and inserted into (3.3). Since each simulation depends on the value function to be minimized, the Monte-Carlo estimate also depends on this value function. That is, the aim to obtain the approximated value function (4.6) leads to $J_{\sigma=i}^{MC\_init}$ in (4.8).

Remark 4.2 After multiple runs, the result of the Monte-Carlo simulation is assumed to provide a sufficiently good estimate $J_{\sigma=i}^{MC\_init}$ of the true value function (under deterministic control). This assumption is based on the uniqueness of the solution to the LQ control problem and the employment of the continuation process, which keeps the solution in the minimum.

Importance of Value Function Approximation

In some practical applications, only the knowledge of the true value function is desired, instead of the optimal control leading to the value function. Then, a sufficiently good approximation of the value function, for example by means of (3.5), may serve as an estimate of the true value function. Since a Monte-Carlo estimate requires a multitude of simulations, the approximator of the value function is much easier to obtain, since only one calculation is needed. When the influence of the neglected higher-order derivatives in the Taylor series expansion of the value function increases with increasing standard deviation of the noise term, the error of the proposed approximation also increases in cases, where these derivatives do not vanish. For a fixed terminal time and finite-horizon control[1], the quality of the value function approximation (4.6) for system (4.1) is discussed in detail in [DOW+06] .

4.2.3 Quality of the Initializing Controller

Since the controller, which employs the results of Algorithm 2, is in principle equivalent to a controller for a deterministic system, it is not reasonably applicable to many classes of noise-affected systems in general. In case of systems, when even small noise effects causes relatively strong influence, for example system (4.1), the application may be justified. Otherwise, the approximation of the value function up to second-order derivatives cannot be justified, if the higher-order derivatives of the value function do not vanish.

1 not MPC

Comparison with Dynamic Programming on a Static Grid

To compare the solution of the proposed initializing algorithm with a standard solution, a dynamic programming approach on a static grid has been chosen.

In this offline-computation approach, the state space as well as the control space is discretized. The discretized time-invariant state space heuristically covers the maximum range of the grid $\mathcal{G}_k^{init}$. This range has been determined heuristically after multiple simulations with the proposed algorithm. Moreover, the set of possible control variables has also been discretized, such that it is possible to cover the range of the whole grid from each grid point. For both systems (4.1) and (4.2), sets of 250 states and 300 controls have been chosen. Detailed information about the discretization of system (4.1) and system (4.2) is given in Tables 4.1 and 4.2, respectively.

Table 4.1: Discretization intervals and distance of the grid points of the state and control space in case of system (4.1) for different noise influences.

	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$
discretized interval (state)	$[-1, 1]$	$[-1.1, 1]$	$[-1.8, 1.7]$
distance of two grid points (state)	0.0080	0.0084	0.0140
discretized interval (control)	$[-2, 2]$	$[-2.1, 2]$	$[-2.8000, 2.7000]$
distance of two grid points (control)	0.0134	0.0137	0.0184

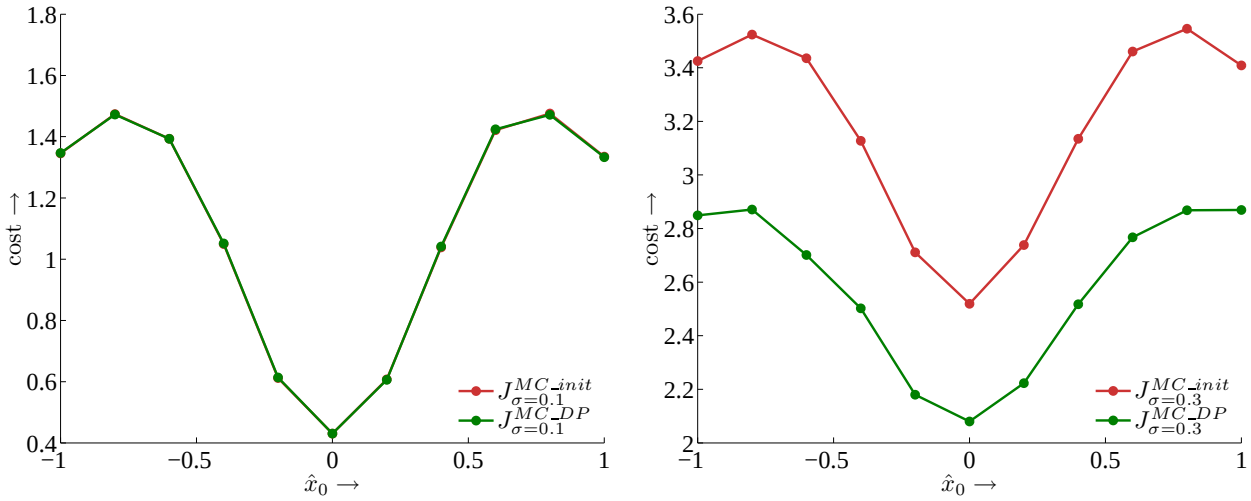
Table 4.2: Discretization intervals and distance of the grid points of the state and control space in case of system (4.2) for different noise influences.

	$\sigma = 1$	$\sigma = 3$	$\sigma = 5$
discretized interval (state)	$[-10, 10]$	$[-13, 13]$	$[-23, 20]$
distance of two grid points (state)	0.0800	0.1040	0.1720
discretized interval (control)	$[-11, 11]$	$[-13.9999, 13.9999]$	$[-23.9999, 21]$
distance of two grid points (control)	0.0733	0.0933	0.1499

To analyze the quality of the controller employing the initializing algorithm, the Monte-Carlo estimate $J_{\sigma=i}^{MC\_DP}$ of the original value function (4.5) for $\gamma = 1$ and system (4.1) is compared to $J_{\sigma=i}^{MC\_init}$ for $i \in \{0.1, 0.3\}$ as depicted in Figure 4.4.

Remark 4.3 Since the value function represents the minimal expected cost, smaller values represent lower cost, which is desirable.

In case of noise standard deviation of $\sigma = 0.1$, only slight differences between both Monte-Carlo estimates can be seen as shown in Figure 4.4(a). A suitable approximation by means of the considered dynamic programming approach is assumed, due to the small distance of the elements of the discretized sets as revealed by Table 4.1. Therefore, the quality of the solution provided by the proposed algorithm is sufficiently good. In case of noise standard deviation



(a) For $\sigma = 0.1$ only minimal differences can be seen.

(b) The differences between the DP-controller and the controller using the initializing algorithm are significant. The quality of the DP-controller significantly outperforms the quality of the other controller.

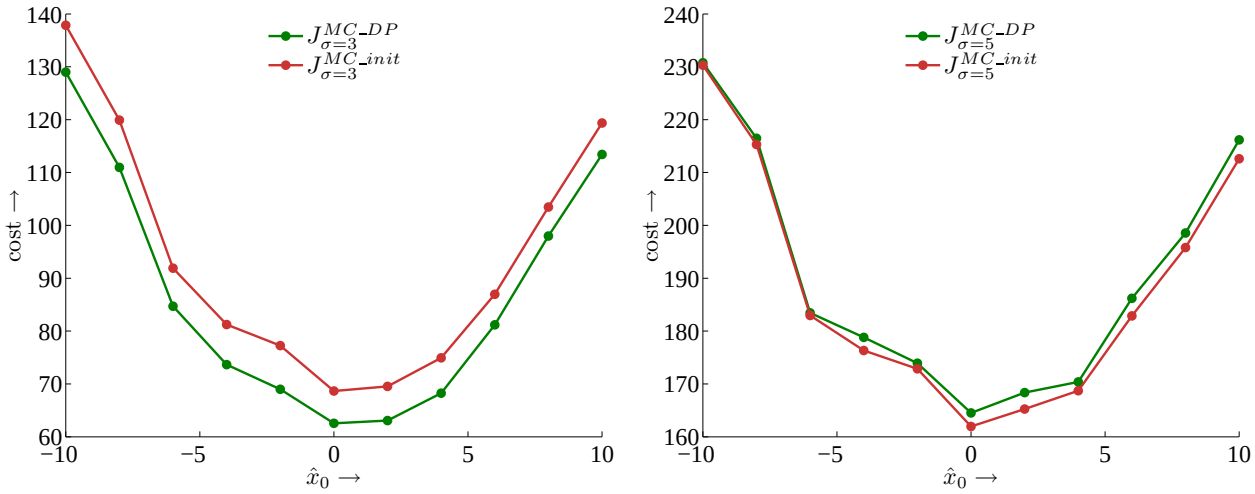
Figure 4.4: Monte-Carlo estimates of the value function resulting from the discrete DP algorithm and the proposed algorithm for system (4.1).

$\sigma = 0.3$, the differences between both Monte-Carlo estimates become significant. This is due to the fact that the error in the determination of the control sequence by means of Algorithm 2 increases with increasing standard deviation. Therefore, the controller based on DP on a static grid significantly outperforms the initializing controller.

In case of system (4.2), the Monte-Carlo estimates $J^{MC\_init}_{\sigma=i}$ and $J^{MC\_DP}_{\sigma=i}$ of the corresponding value functions are compared in Figure 4.5 for $i \in \{3, 5\}$. Similar to system (4.1), the DP algorithm on the static grid provides better results for system (4.2) in case of $\sigma = 3$ as depicted in Figure 4.5(a). Depending on the size of the considered region of the state space, the distance between the grid points increases as described in Table 4.2. Therefore, the quality of the static grid based DP controller decreases. In case of $\sigma = 5$, even the initializing algorithm provides superior results to the discrete DP algorithm, which is revealed by Figure 4.5(b).

Thus, the accuracy of the dynamic programming algorithm in case of the second considered system is not close to the true, but unknown, solution. Due to the required amount of memory, it was not possible to choose more grid points and, therefore, to obtain a finer discretization of the state space. For implementation details of this algorithm, it is referred to Appendix F.3.

Remark 4.4 As described in the previous part, the results of the controller employing Algorithm 2 to obtain the desired optimal control sequence, provide similar results to another controller in case of small noise influence. This controller is based on dynamic programming on a static grid. With increasing noise influence, the grid based DP algorithm outperforms the Algorithm 2, if the length of the discretization intervals is sufficiently small. Otherwise, the quality of the grid based controller decays rapidly. Therefore, the considered variant of



(a) The controller based on DP on the static grid provides better results than the initializing controller with $\sigma = 3$ for system (4.2).

(b) In case of $\sigma = 5$, the range of the restricted state space as well as the length of the discretization intervals increases. Therefore, the quality of the controller based on DP on the static grid significantly decreases. Even the initializing controller outperforms the controller based on DP on a static grid.

Figure 4.5: Monte-Carlo estimates of the value function resulting from the DP algorithm on a static grid and the proposed algorithm for system (4.2). With increasing length of the discretization intervals, the quality of the controller based on the DP algorithm on a static grid significantly decreases.

dynamic programming is not applicable in general. Thus, another controller is provided, which employs the results of Algorithm 2 as initialization to derive an improved state-feedback control by means of Algorithm 4.

4.3 Dynamic Programming Based on Spline Interpolation of the Value Function

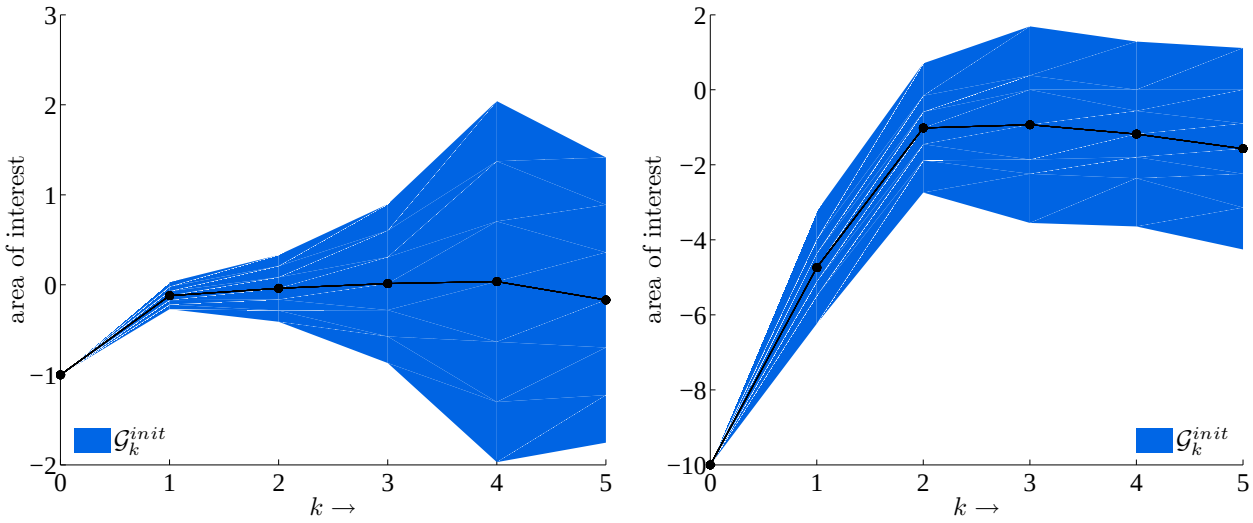
As described in Section 3.1, Algorithm 2 yields a control sequence $(\hat{u}_0, \ldots, \hat{u}_{N-1})$, which solves the nonlinear equation system (3.36). Equation (3.36) satisfies the necessary minimum condition (3.19) for the approximated value function (3.5) for all time steps of the decision-making horizon. There, the value function is approximated by Taylor series expansion of up to second-order derivatives. Employing that sequence as an initial guess, Algorithm 4 determines an updating solution to the considered optimal control problem of the original stochastic system (3.1) and the original value function

$$\begin{aligned} J_N(x_N) &= \frac{1}{2}(x_N - c)^2 \\ J_k(x_k) &= \frac{1}{2}\left((x_k - c)^2 + a\,(u_k^*)^2\right) + \underset{w_k}{\mathrm{E}}\left[J_{k+1}\left(\boldsymbol{x}_{k+1}\right)\right] \quad , \end{aligned} \tag{4.9}$$

which is not approximated by means of Taylor series expansion anymore. The new approximation scheme employs piecewise cubic spline interpolation in the state variable x_k as well as in the control variable u_k within a restricted region of the state space. In the considered scalar case, the sets $\mathcal{P}_k$, $k = 0, \ldots, N$, are given by

$$\mathcal{P}_k := \left\{ \overline{x}_k - \frac{3}{2}\sigma_k^x,\ \overline{x}_k - \frac{3}{4}\sigma_k^x,\ \overline{x}_k - \frac{3}{8}\sigma_k^x,\ \overline{x}_k,\ \overline{x}_k + \frac{3}{8}\sigma_k^x,\ \overline{x}_k + \frac{3}{4}\sigma_k^x,\ \overline{x}_k + \frac{3}{2}\sigma_k^x \right\} , \tag{4.10}$$

where $\overline{x}_k$ denotes the predicted mean of the successor state of $\hat{x}_0$. The corresponding covariance is given by σ_k^x.



(a) Area of interest for system (4.1), where $\hat{x}_0 = -1$, $\sigma = 0.1$.

(b) Area of interest for system (4.2), where $\hat{x}_0 = -10$, $\sigma = 1$

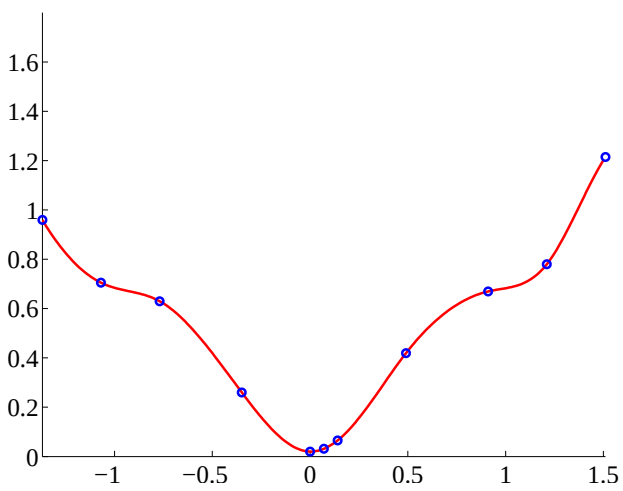
Figure 4.6: Restriction of the state space for both considered systems. Around the sequence of mean values, obtained by the employment of the initial guess of the control sequence and the application of the unscented transformation, the state space is restricted. The spread of this restriction depends on the covariance of each successor state $\boldsymbol{x}_k$ of the initial state $\hat{x}_0$.

The modified restriction $\mathcal{G}_k^{init}$ of the state space according to (3.54) is depicted in Figure 4.6. Figure 4.6(a) depicts the area of interest for system (4.1), and Figure 4.6(b) displays the corresponding restriction of the state space for system (4.2).

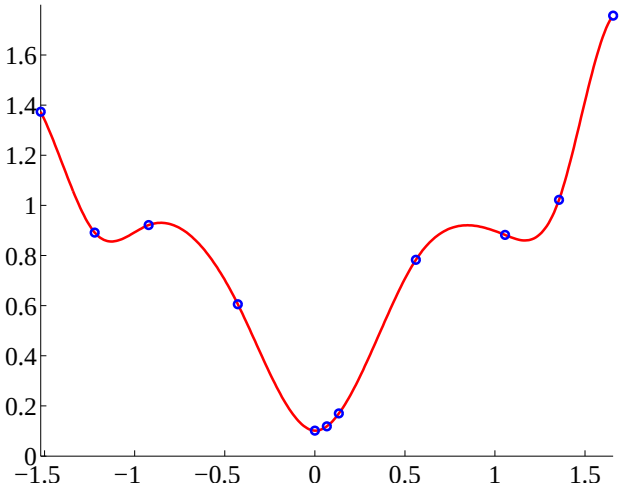
To stabilize the spline interpolation of the value function, two additional dummy points at the extremal points of $\mathcal{G}_k^{init}$ are introduced, which are *not* employed in the optimization. Details on this extension of the grid are given in Appendix F.2.2.

Therefore, in the considered scalar case, the value function is interpolated on a grid, consisting only of 11 points, where only the 7 points of (4.10) are used to find the desired optimal control.

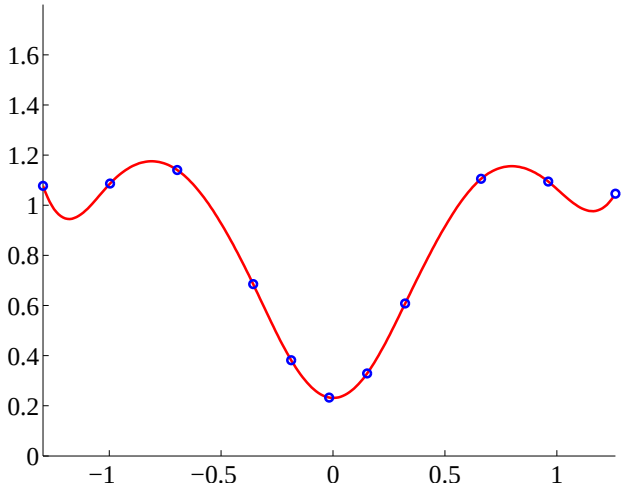
Examples for spline interpolations of the value function (4.9) of system (4.1) within the restricted region of the state space are depicted in Figure 4.7 for different time steps of the decision-making horizon.



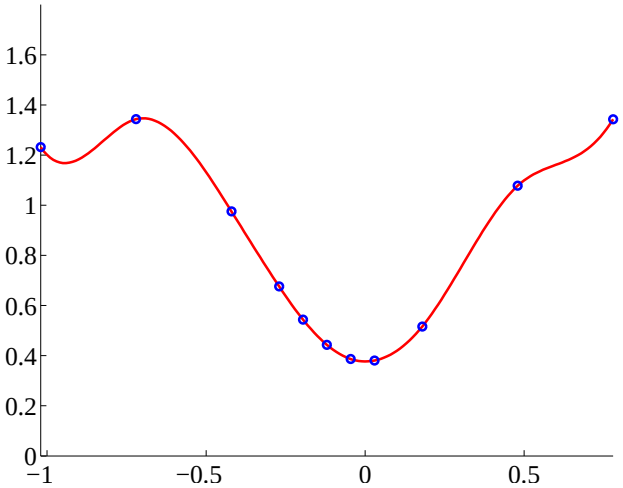
(a) $J_5(x_5)$ in the area of interest, decision-making horizon: 1 step.



(b) $J_4(x_4)$ in the area of interest, decision-making horizon: 2 steps.



(c) $J_3(x_3)$ in the area of interest, decision-making horizon: 3 steps.



(d) $J_2(x_2)$ in the area of interest, decision-making horizon: 4 steps.

Figure 4.7: Spline interpolation of the value function J_k in the area of interest for $\hat{x}_0 = -1$ and $\sigma = 0.2$. The changes of the value function can be seen for different time steps k within a five-step horizon window. With increasing prediction horizon, the expected minimal cost-to-go increases.

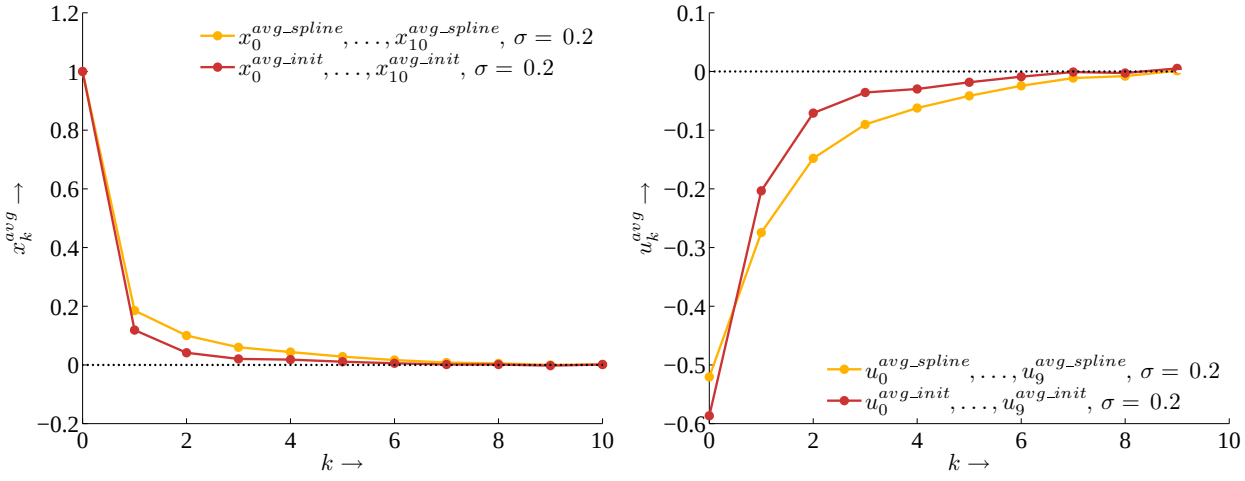
In Figures 4.7(a)–4.7(b), the set $\mathcal{G}_k^{init}$ differs from $\mathcal{P}_k$. The modification according to (3.54) is applied for $k > 3$. Therefore, the cluster of knots around zero can be seen in both figures.

4.3.1 Improvement of the Initial Solution

To evince the improvement of the updating algorithm, compared to the initial solution given by Algorithm 2, the systems (4.1) and (4.2) have been simulated. In each simulation, both the grid based controller and the controller employing Algorithm 2 suffer from exactly the same noise vector. Therefore, a comparison of both controllers is justified, even in case of few simulations.

State and Control Trajectories

In the following, several state and control trajectories are discussed resulting from the application of the proposed controllers. The corresponding algorithms are given by Algorithm 2 and its improvement Algorithm 4.



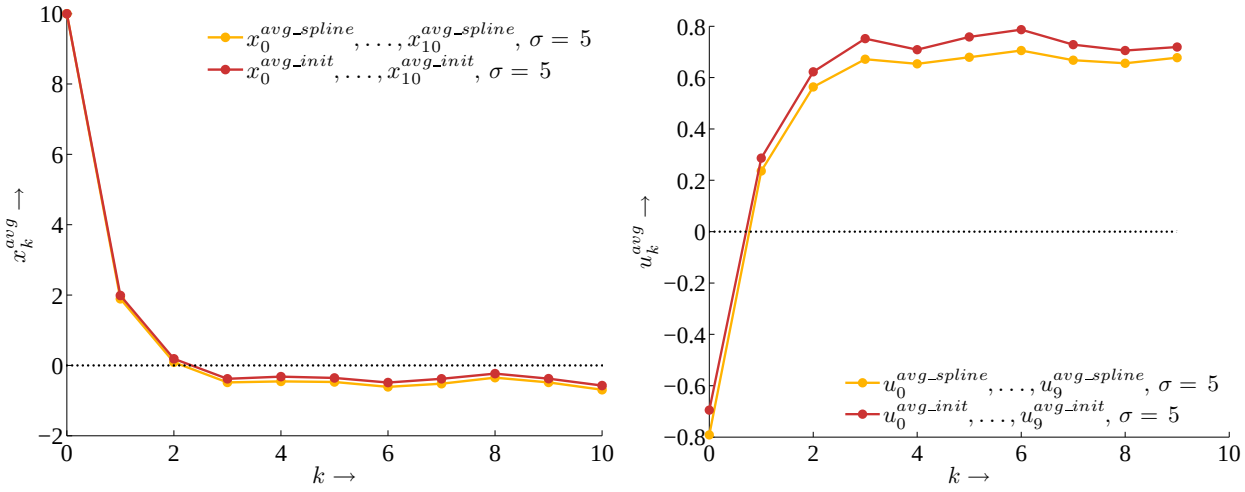
(a) Average state trajectories for system (4.1). On average, the controller employing only the initial solution leads the system state faster toward the desired goal point than the controller, which additionally employs the improved solution.

(b) Average control trajectories for system (4.1). Since the state trajectories converge toward zero, the trajectories of the corresponding controls converge toward zero, too.

Figure 4.8: Average state and control trajectories of system (4.1) for the initializing solution and the proposed additional improvement for $\sigma = 0.2$ and $\hat{x}_0 = 1$.

In Figure 4.8, the average state and the corresponding control trajectories for system (4.1) with $\sigma = 0.2$ and $\hat{x}_0 = -1$ are depicted after 2000 simulations. Both considered controllers lead the state toward the desired unstable equilibrium point $c = 0$. The controller, which employs only Algorithm 2, leads the system state faster toward zero on average, since it does not consider the noise influence explicitly. Due to the original cost function to be minimized, the proposed new controller incorporates the noise influence in its determination of the optimal state-feedback

control. Hence, the results are more conservative, that is, the state-feedback and the state do not converge toward zero as fast as the state-feedback resulting from Algorithm 2. This behavior is depicted in Figure 4.8(b).



(a) Average state trajectories for system (4.2). Depending on the system function, the cost function, and the noise influence on the system, the average state trajectories do not converge toward the desired state $c = 0$.

(b) Average control trajectories for system (4.2). The average control trajectories for ten time steps also converge toward a value $\neq 0$.

Figure 4.9: Average state and control trajectories of system (4.2) for the initial solution and the proposed improvement for $\sigma = 5$ and $\hat{x}_0 = 10$.

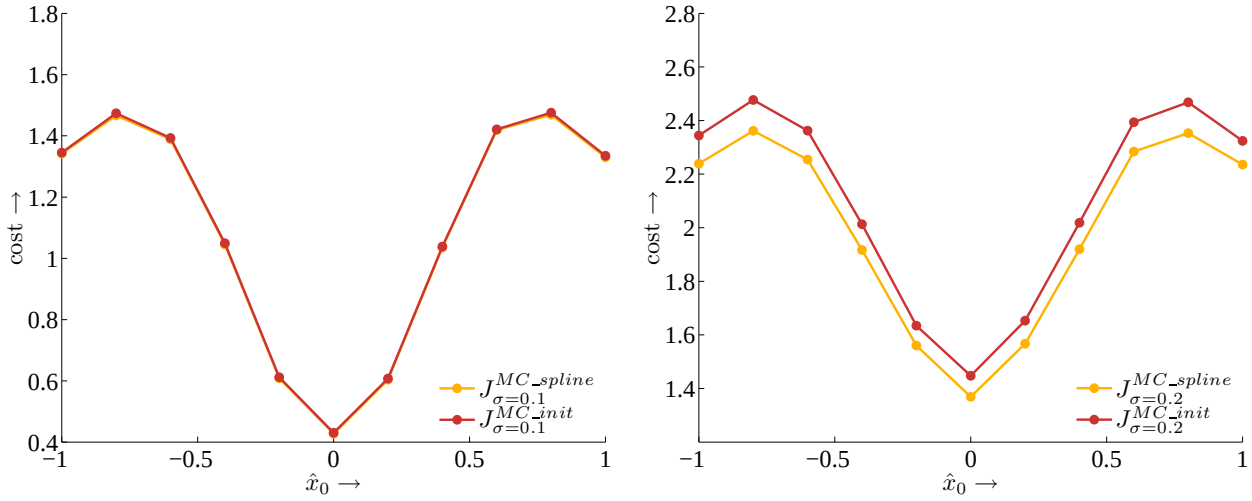
For system (4.2), Figure 4.9 reveals the deviations of the average trajectories of the controller employing only the initial solution (Algorithm 2) and the controller, which additionally employs the proposed improvement (Algorithm 4). Since the initial solution does not explicitly consider the noise influence on the system, the resulting state sequences are closer to the desired target state $c = 0$, compared to the more complex controller, which additionally employs Algorithm 4. However, the simple structured controller requires larger inputs u_k to obtain this result. Depending on the structure of the system function (4.2) and the weights of the parameters of the value function (4.9), the states do not converge toward the equilibrium point $c = 0$, which is shown in Figure 4.9(a). Therefore, the controls do not converge toward zero neither, which is depicted in Figure 4.9(b).

Remark 4.5 With a weighting factor $a = 0.01$ in (4.9), both the average states and the average state-feedback controls of system (4.2) converge toward zero.

Value Functions

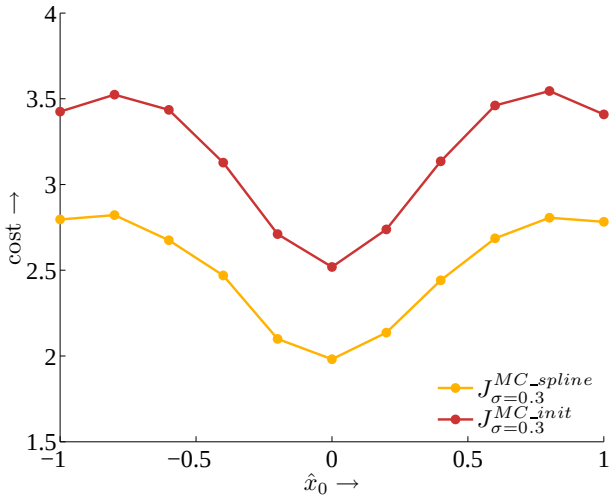
According to Section 4.2, Monte-Carlo estimates of the minimized value functions $J_{\sigma=i}^{init}$ and $J_{\sigma=0}^{spline}$ are compared, where $J_{\sigma=0}^{spline}$ denotes the value function of the controller, which is based on dynamic programming with spline interpolation of the value function (Algorithm 4).

In Figure 4.10, the Monte-Carlo estimates $J_{\sigma=i}^{MC\_init}$ and $J_{\sigma=i}^{MC\_spline}$ for the first considered system (4.1) with $i \in \mathcal{S}_1$ and $\hat{x}_0 \in \mathcal{X}_1$ are presented after 2000 simulations. In case of noise



(a) Monte-Carlo estimates of $J_{\sigma=0.1}^{init}$ and $J_{\sigma=0.1}^{spline}$.

(b) Monte-Carlo estimates of $J_{\sigma=0.2}^{init}$ and $J_{\sigma=0.2}^{spline}$.

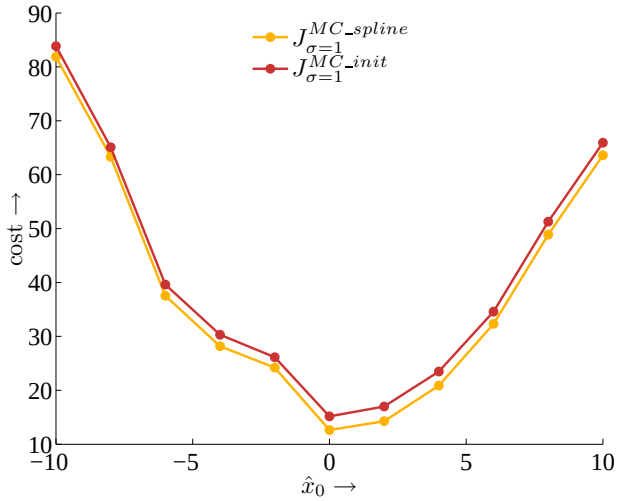


(c) Monte-Carlo estimates of $J_{\sigma=0.3}^{init}$ and $J_{\sigma=0.3}^{spline}$.

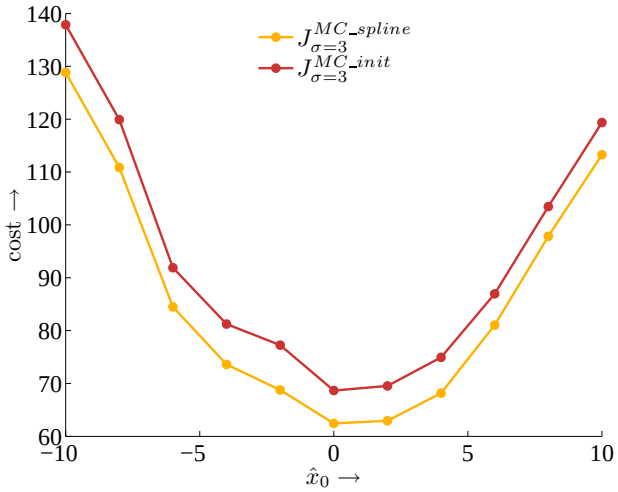
Figure 4.10: Monte-Carlo estimates of the value functions $J_{\sigma=i}^{init}$ and $J_{\sigma=i}^{spline}$ of system (4.1) for increasing noise influence. The stronger the noise affects the system, the better results are presented by the new algorithm compared to the initial solution.

influence with standard deviation $\sigma = 0.1$, the differences are small, since the value function approximation by means of Taylor series expansion up to second-order derivatives is almost correct. Therefore, Algorithm 2 provides satisfactory results, stressed by Figure 4.10(a). Nevertheless, the differences become more and more significant, when the noise influence increases. This fact is revealed by Figures 4.10(b) and 4.10(c). Moreover, it can be seen that the offset between $J_{\sigma=i}^{MC\_init}$ and $J_{\sigma=i}^{MC\_spline}$ is not constant. This property is due to the fact that Algorithm 4 explicitly incorporates the noise influence.

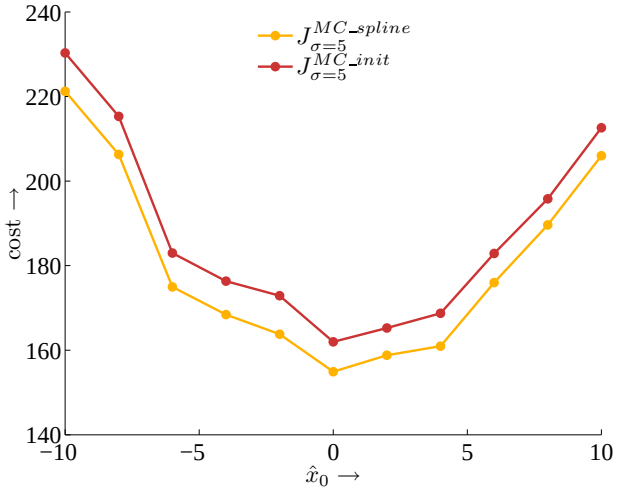
In case of the second considered system (4.2), the corresponding Monte-Carlo estimates are depicted in Figure 4.11, where $\hat{x}_0 \in \mathcal{X}_2$ and $\sigma \in \mathcal{S}_2$. Similar results to before are revealed.



(a) Even for $\sigma = 1$, a slight improvement of the initial solution can be seen.



(b) With increasing noise influence, the difference between both Monte-Carlo estimates becomes more significant.



(c) For $\sigma = 5$ the proposed update algorithm still yields much better results than the initial solution.

Figure 4.11: Monte-Carlo estimates of the value functions $J^{init}_{\sigma=i}$ and $J^{spline}_{\sigma=i}$ of system (4.2) for $i \in \mathcal{S}_2$ and $\hat{x}_0 \in \mathcal{X}_2$. The improvement of the updating algorithm can be seen easily. The differences become more obvious with increasing noise disturbances.

With increasing noise influence, the improvement of the initial solution described in Algorithm 4 becomes more and more significant, which is stressed by Figures 4.11(a)–4.11(c).

From the previous results, it can be seen that the additional employment of Algorithm 4, in fact, improves the solution provided by Algorithm 2. Moreover, this improvement is not restricted to one specific setting of one special system.

4.4 Replacement of Numerical Integration with the Unscented Transformation

As mentioned in Section 3.2.5, there are several ways to determine the value $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$ in line 9 of Algorithm 4. In this thesis, numerical integration is used to avoid additional inaccuracies due to another approximation method. The employment of the UT seems to be a promising approximation for the considered case, since the density of the successor state is Gaussian. This point is important, since the parameters of the UT are well-known in case of Gaussian distributions. In the following, several differences between the implementation of the unscented transformation and the numerical integration are discussed.

On the one hand, the computation of the value $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$ by means of the UT is computationally less demanding than numerical integration. On the other hand, further approximation errors are introduced, in contrast to numerical integration.

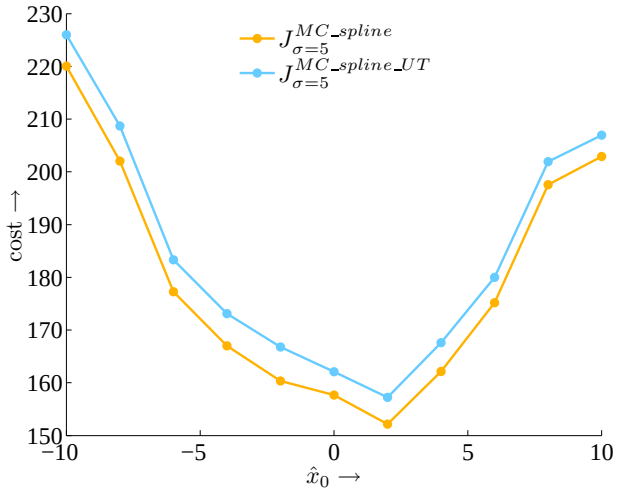


Figure 4.12: Monte-Carlo estimates of the value function, where the numerical integration is replaced with the UT in system (4.2) for $\sigma = 5$. If the UT replaces the numerical integration to determine $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$, the quality of the controller decreases significantly.

In Figure 4.12, two Monte-Carlo estimates of the true value function of system (4.2) are depicted. These estimates depend on the chosen controller, where noise with standard deviation $\sigma = 5$ affects the system. As expected, the function of the Monte-Carlo estimate $J_{\sigma=5}^{MC\_spline}$ is less than the Monte-Carlo estimate $J_{\sigma=5}^{MC\_spline\_UT}$, which employs the UT instead of numerical integration. Hence, exact computation of the value $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$ is an important point of the proposed algorithm.

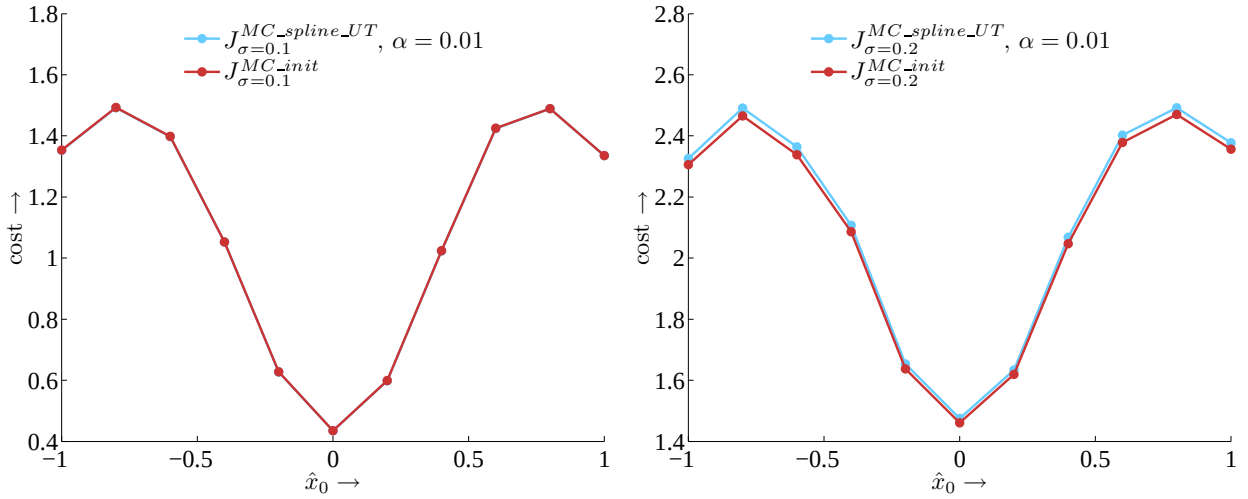
Remark 4.6 It is important to mention that Figure 4.12 represents simulation results, which are independent of the other results in this thesis. This is the reason, why the Monte-Carlo estimates of $J_{\sigma=5}^{spline}$ may look different compared to previous results.

Review: Scaled unscented transformation

A modification of the standard UT is given by the scaled UT, which is introduced in [Jul02]. The scaled version of the unscented transformation modifies a given sigma point $\mathcal{X}_i^{(k)} \in \mathcal{X}^{(k)}$, such that it is moved toward or away from the mean value $\overline{x}_k = \mathcal{X}_0^{(k)}$ according to

$$\tilde{\mathcal{X}}_i^{(k)} = \mathcal{X}_0^{(k)} + \alpha(\mathcal{X}_i^{(k)} - \mathcal{X}_0^{(k)}) \ . \tag{4.11}$$

The corresponding weights have to be adjusted appropriately. For further details on the scaled UT, it is referred to Appendix D.



(a) For $\alpha = 0.01$ the Monte-Carlo estimates for both the initial solution and the algorithm employing the scaled UT almost coincide.

(b) The difference between the initial solution and the algorithm with the scaled UT becomes more obvious. Unexpectedly, the value function of the initial solution is smaller.

Figure 4.13: Monte-Carlo estimates of the value functions of system (4.1) of two controllers, where the first one employs the initial solution. The second one additionally employs Algorithm 4, where the scaled unscented transformation replaces the numerical integration to determine $\mathrm{E}_{w_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$. In the considered case, the scaled unscented transformation fails to predict the expected value $\mathrm{E}_{w_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ sufficiently well, when the scaling parameter is set to $\alpha = 0.01$.

The scaled UT with a scaling parameter

$$\alpha = 0.01$$

to determine $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$ is analyzed in the following. For system (4.1), the Monte-Carlo estimates $J_{\sigma=i}^{MC\_spline}$ and $J_{\sigma=i}^{MC\_init}$ with $i \in \{0.1, 0.2\}$ are depicted in Figure 4.13. It can be seen in Figure 4.13(a) that the Monte-Carlo estimates of the controller, which only applies the minimum principle, and the controller, which additionally employs the proposed improvement with the scaled UT, almost coincide. This fact does not surprise, since the absolute noise influence is small with $\sigma = 0.1$. The differences become more obvious, when the standard deviation of the noise is increased to $\sigma = 0.2$. The corresponding functions are shown in Figure 4.13(b). Unexpectedly, the assumed better controller yields worse results than the

simple controller. This behavior can be explained by the employment of the scaled UT with $\alpha = 0.01$ to determine $\mathrm{E}_{w_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ in Algorithm 4, where higher-order information gets lost. This loss of higher-order information is due to the small value of α, even if the prediction of the mean contains a second-order bias correction term.

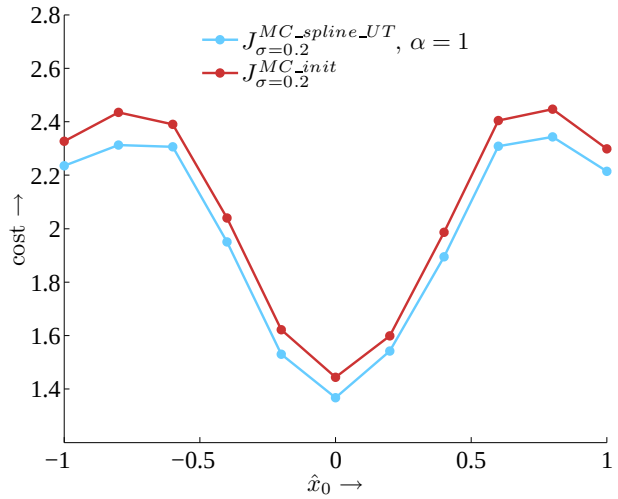


Figure 4.14: Monte-Carlo estimates of the value functions of system (4.1) of two controllers employing the initial algorithm and the improved algorithm with the *un*scaled UT. With the unscaled UT, the new algorithm, in fact, improves the initial solution.

In case of

$$\alpha = 1 \ ,$$

that is, the employment of the unscaled UT, the Monte-Carlo estimates of value functions for $\sigma = 0.2$ are given in Figure 4.14. The accuracy of the unscented transformation is improved, due to the larger value of $\alpha = 1$. Therefore, the expected improvement of the initial solution is obtained. That is, the Monte-Carlo estimate of the value function $J_{\sigma=0.2}^{init}$ is greater than the Monte-Carlo estimate of $J_{\sigma=0.2}^{spline\_UT}$. As indicated by previous simulation results, this property rests upon the superiority of the results, when Algorithm 4 is additionally employed.

Nevertheless, the unscented transformation has not been employed to calculate $\mathrm{E}_{w_k}[J_{k+1}(\underline{\boldsymbol{x}}_{k+1}^{(j)})]$ in Algorithm 4 to avoid additional inaccuracies in the determination of the optimal state-feedback control. Moreover, the solution, with which the new algorithm is compared in the next section, does not employ the unscented transformation at this computation step either.

4.5 Comparison with Other Solutions

In this section, the proposed new algorithm (initialization by Algorithm 2 and subsequent employment of Algorithm 4) is compared to other possible solutions to the considered optimal control problem.

4.5.1 Dynamic Programming on an Adaptive Grid with Few Grid Points

The proposed new algorithm is compared to a grid based dynamic programming approach in the following. This approach employs the same adaptive grid $\mathcal{G}_k^{init},\ k = 0, \ldots, N$, as Algorithm 4 to perform dynamic programming. That is, the considered grid consists of 7 grid points. To determine this grid, Algorithm 2 is employed at each time step to return the initializing control sequence, which solves the nonlinear equation system (3.36). Since the chosen DP algorithm also depends on the adaptive changes of the state space restriction, this algorithm is an online algorithm, too.

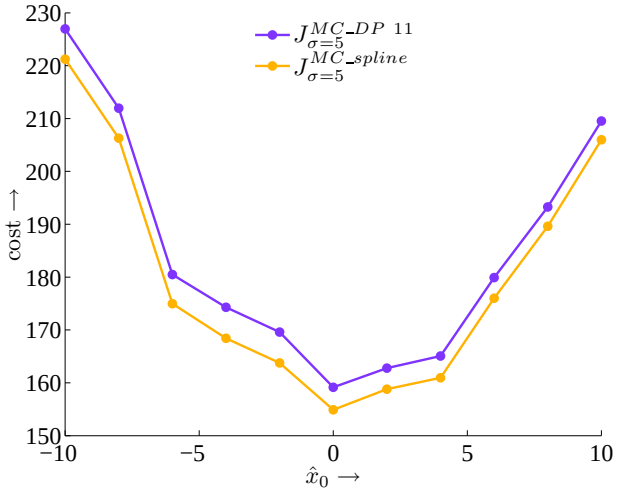
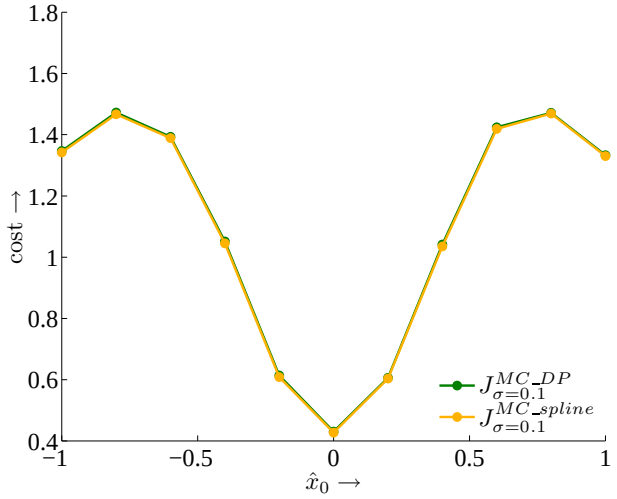


Figure 4.15: Comparison of the Monte-Carlo estimates $J_{\sigma=5}^{MC\_spline}$ and $J_{\sigma=5}^{MC\_DP\ 11}$ for system (4.2). The impact through the spline interpolation is obvious.

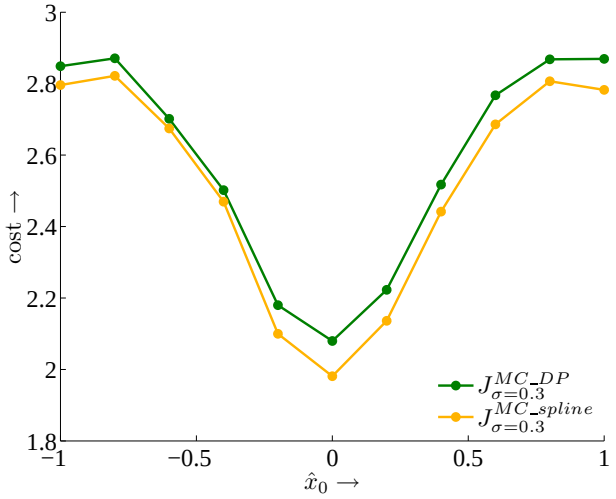
The difference between both algorithms is that this grid based dynamic programming does not interpolate the value function. Therefore, the integration to obtain the value $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$ is replaced with a summation. The consideration of this solution demonstrates that the employment of spline interpolation is beneficial. The Monte-Carlo estimates of the value functions $J_{\sigma=5}^{spline}$ and $J_{\sigma=5}^{DP\ 11}$ for system (4.2) are compared in Figure 4.15. Again, all considered controllers suffer from the same noise vector in each simulation, which explains the symmetry of the peaks in the Monte-Carlo estimates of the corresponding value functions. It can be seen easily that the spline interpolation, in fact, yields better results than the adaptive grid based dynamic programming algorithm.

4.5.2 Dynamic Programming on a Static Grid with Many Grid Points

In the following, the proposed algorithm is compared to the offline dynamic programming algorithm on the static grid introduced in Section 4.2.3. As already mentioned, the quality of the DP algorithm strongly depends on the granularity of the discretized state and control spaces. In Figure 4.16, the Monte-Carlo estimates of the value functions $J_{\sigma=i}^{DP}, J_{\sigma=i}^{spline},\ i = 0.1, 0.3$, of system (4.1) are compared, which are obtained by application of both considered controllers.



(a) For $\sigma = 0.1$ the Monte-Carlo estimates of both controllers almost coincide.



(b) For $\sigma = 0.3$ the controller, which employs Algorithm 4, yields better results than the controller based on DP on a static grid.

Figure 4.16: Comparison of the new controller with a controller, which employs a static grid based DP algorithm, for system (4.1).

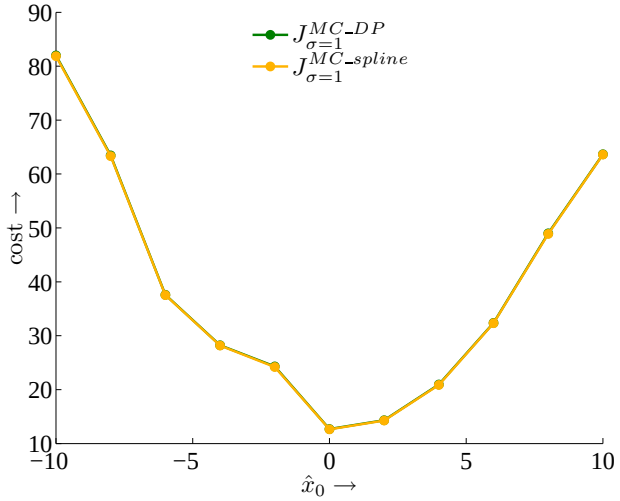
In case of small noise influence, the differences between both considered controllers are small as indicated by Figure 4.16(a). Due to the small discretization intervals given in Table 4.1, the DP algorithm on the static grid is assumed to yield satisfactory results. Nevertheless, the new controller provides at least results of same quality. With increasing noise influence and, therefore, larger discretization intervals, the quality of the results of the DP algorithm on the static grid decays and is significantly outperformed by the results of the new controller, stressed by Figure 4.16(b) for $\sigma = 0.3$.

In Figure 4.17, the Monte-Carlo estimates of the value functions for system (4.2) are plotted. In case of small noise influence, the differences between both controllers are small as depicted in Figure 4.17(a) for $\sigma = 1$. With increasing σ, the considered region of the state and control spaces becomes larger. Therefore, the offline dynamic programming algorithm yields worse results than the proposed new controller, which is shown in Figure 4.17(b) for $\sigma = 5$.

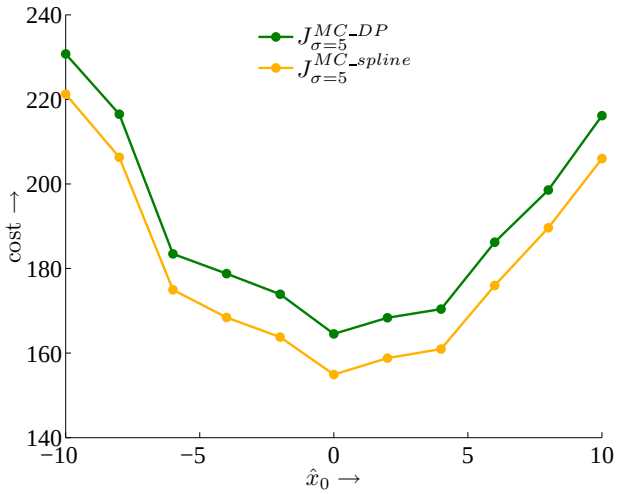
Taking everything into account, the proposed new algorithm and the DP algorithm on a static grid yield results of the same quality in case of small noise influence and small discretization intervals. With increasing noise standard deviation, the new controller outperforms the static grid based DP controller significantly. Moreover, the proposed method is able to adapt the spread of the grid, depending on the current measured state.

4.5.3 Dynamic Programming on an Adaptive Grid with Many Grid Points

The combination of the adaptive grid based controller with few grid points from Section 4.5.1 and the controller based on dynamic programming on a static grid with many grid points from



(a) For small noise influence, the quality of both considered controllers is similar.



(b) For $\sigma = 5$, the controller employing Algorithm 4 yields better results than the DP based controller.

Figure 4.17: Comparison of the new controller with a controller, which employs a static grid based DP approach, for system (4.2).

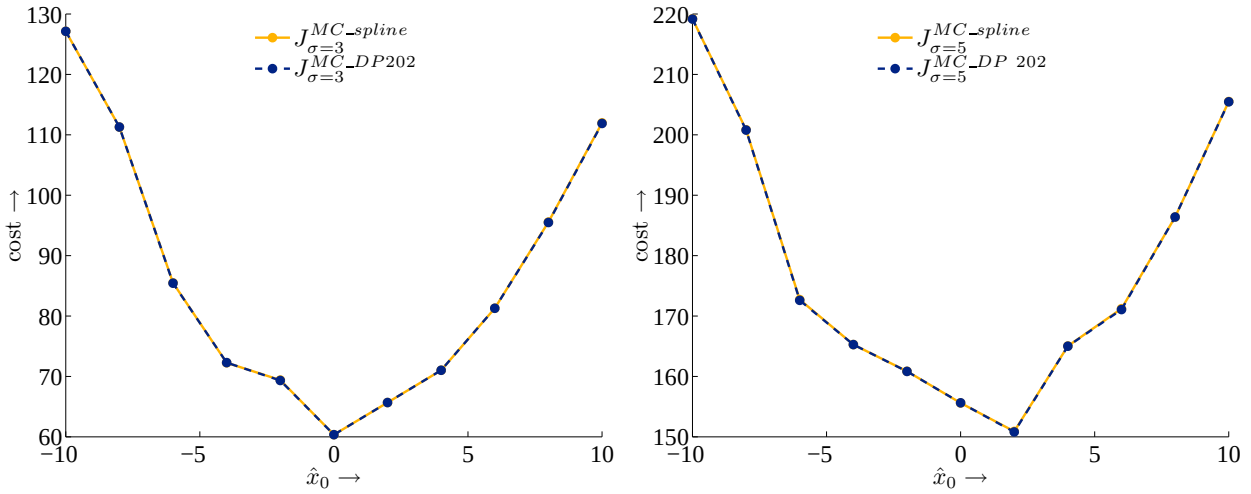
Section 4.5.2 results in a controller, which performs dynamic programming on an adaptive grid with many grid points. To determine the adaptive grid at each time step, Algorithm 2 is employed to obtain the control sequence, which solves the nonlinear equation system (3.36), such that the restricted region of the state space can be determined. This controller is expected to yield the best results of all mentioned controllers, since it combines the advantages of the adaptive grid and a fine discretization. Therefore, the proposed approach to optimal control of nonlinear stochastic systems is compared to this algorithm in the following.

The range of the adaptive grid equals the range of $\mathcal{G}_k^{init}$, $k = 0, \ldots, N$, which is used by Algorithm 4. In contrast to spline interpolation on a grid with few points, the state space as well as the control space is discretized with 202 grid points. Since the range of the grid adapts with each time step and depends on the current system state, this algorithm is also an online algorithm.

The Monte-Carlo estimates $J^{MC\_spline}_{\sigma=i}$ and $J^{MC\_DP202}_{\sigma=i}$ are displayed for $i = 3, 5$ in Figure 4.18. The figure reveals that there is almost no difference in the considered Monte-Carlo estimates after 500 simulations. Independent of the noise influence, both compared algorithms yield almost the same results for all noise influences and both considered systems. This property is exemplarily revealed by Figure 4.18 for system (4.2) and $\sigma = 3, 5$.

4.6 Computational Effort

In this section, the computational effort of the initializing solution, the proposed solution based on spline approximation in the DP algorithm, the DP algorithm on an adaptive grid with few



(a) $J^{MC\_spline}_{\sigma=3}$ and $J^{MC\_DP202}_{\sigma=3}$ almost coincide.

(b) Even in case of $\sigma = 5$, both controllers yield state-feedback controls of similar quality, which cannot be distinguished easily.

Figure 4.18: Comparison of the Monte-Carlo estimates of the value function for system (4.2) with $\sigma = 3, 5$ for the new algorithm and an algorithm, which is based on DP on an adaptive grid with many grid points. In both cases, there is no obvious difference.

points, and the DP algorithm on an adaptive grid with many points are compared. All these algorithms are online algorithms and, therefore, not comparable to the offline algorithm on the static grid.

The computational time has been evaluated for system (4.1), where model predictive control has been carried out for ten steps. The decision-making horizon of the controller was set to five steps. Since all solutions employ the initial solution at least to initialize a subsequent algorithm, the execution time for Algorithm 2 (*init*) is set to 1 time unit. The relative execution times are given in Table 4.3.

Table 4.3: Relative execution time for different algorithms.

init	*spline*	*DP* 11	*DP* 202
1	2.51	1.04	31.72

The proposed dynamic programming algorithm with spline interpolation of the value function on an adaptive grid (*spline*) computes about 2.5 times of the execution time of *init*. The DP algorithm on the same adaptive grid without spline interpolation (*DP* 11) and the initial solution have similar computation times. The computationally most expensive algorithm is the DP algorithm on an adaptive grid with many (202) grid points and no spline interpolation (*DP* 202). Because of the demanding computational requirements, this algorithm is not applicable in practice.

4.7 Survey of Applied Methods

In case of system (4.1), all considered methods provide very similar results in case of small noise influence. With increasing noise disturbances, the quality of the results of Algorithm 2 decreases, compared to the solutions of the spline interpolation based DP algorithm and the DP algorithm with many grid points on the adaptive grid. On the one hand, the offline DP algorithm, which employs a static grid, possesses the advantage that only a look-up table is required to determine the optimal state-feedback control. On the other hand, the demanding amount of memory to pre-calculate this table restricts the applicability of this algorithm. If the distance between two elements of the discretized state and control spaces is not sufficiently small, the quality of the provided results is not satisfactory anymore. The employment of the DP algorithm on the adaptive grid with few grid points does not require much computation time and yields good results in case of smaller noise influence on the systems (4.1) and (4.2). With increasing noise, it still yields better results than the initial solution and the offline dynamic programming algorithm. However, the quality of the results is worse compared to the quality provided by the DP algorithm on the adaptive grid with many grid points and the proposed new algorithm.

The simulation results of the previous parts reveal that the proposed new algorithm, which is based on the spline interpolation of the DP value function on an adaptive grid, and the computationally demanding DP algorithm on an adaptive grid with many grid points yield the best results on average for all considered systems and all simulated noise influences. Only slight differences can be seen in the results of both algorithms. Because of the shorter computation time of the new algorithm according to Table 4.3, the new algorithm is strongly preferable.

CHAPTER 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, the optimal finite-horizon control problem for nonlinear systems has been discussed, where the considered system suffers from additive zero-mean noise. An online-computation algorithm has been presented, which consists of two parts.

In the first part, an initial guess of the optimal control sequence has been derived for the nonlinear system, where the value function was approximated by means of Taylor series expansion up to second-order derivatives. Extensions to higher-order approximations are possible, but computationally demanding and requrire the existence of inverse mappings. After the approximation of the value function, the properties of a stochastic version of the Hamilton function have been exploited to be able to apply a stochastic minimum principle. Employing this minimum principle, a two-point boundary-value problem has been derived. The resulting nonlinear equation system has been solved by means of a continuation process to overcome numerical difficulties. The comparison with a dynamic programming algorithm on a static grid revealed that the solutions to the approximated optimal control problem are suitable in case of small noise influence. Moreover, it has been shown that the quality of the DP algorithm strongly depends on the granularity of the state space discretization.

However, both algorithms do not provide satisfactory results in general. Therefore, a second algorithm has been developed to mitigate those effects. This algorithm employed the solution of the first part, that is, the control sequence solving the two-point boundary-value problem, as prior knowledge. Hence, the state space could be restricted to an area of interest, within which the original optimal control problem has been solved approximately. The state variable as well as the control variable was treated as continuous variables. This result has been obtained by interpolating the value function of the dynamic programming algorithm on a grid with few grid points by means of piecewisely defined cubic splines. In contrast to a standard dynamic programming approach on a static grid, the number of grid points has been reduced significantly, which saves much computation time and memory.

The proposed method is an online algorithm, which accounts for the state- and time dependent adaptation of the grid. Moreover, a satisfactory tradeoff between accuracy and computational

effort is given. Even if the curse of dimensionality of dynamic programming is not completely eliminated, its effect is reduced to a small number of grid points.

Simulations for two scalar systems revealed the superiority of the provided algorithm to a dynamic programming solution with an adaptive grid consisting of few grid points and a standard dynamic programming solution on a static grid with many grid points. Moreover, the new algorithm possesses the same quality as a dynamic programming algorithm on an adaptive grid, with many grid points, which is computationally very demanding. Furthermore, the improved algorithm, in fact, performed much better than the initial solution.

5.2 Future Work

Several open questions are worth dealing with in the future. One important point requiring deeper analysis is the derivation of a stochastic two-point boundary-value problem, which has not been found in literature yet. The state propagation by means of the original system function and the employment of the mean values of the successor states to initialize the costate recursion may be a possible approach to obtain this result. The corresponding costate sequence has to be chosen appropriately. Furthermore, generalizations of the stochastic Hamiltonian and the stochastic minimum principle to the case of non-additive noise influence have to be discussed in detail.

The influence of the state space restriction to perform dynamic programming has to be analyzed as well as the number and the position of the grid points. A minimal number of grid points and their optimal placement are desired, such that the interpolation of the value function is of satisfactory accuracy and the computational effort remains acceptable. Depending on the influence of the accuracy of the propagation of means and covariances to determine the state space restriction, the employed unscented transformation has to be optimized or replaced with another method.

Since the currently implemented algorithm employs computationally demanding numerical integration, alternative approaches have to be discussed in the future. On the one hand, the unscented transformation is an interesting candidate for replacing the numerical integration to calculate the mean of a nonlinearly transformed random variable. On the other hand, the examples in Section 4.5 of this thesis motivate alternative approaches, such as proposed in [HBH06], where a closed-form prediction for nonlinear, time-invariant systems is introduced.

In the proposed algorithm, spline interpolation is employed to interpolate the value function of the dynamic programming algorithm. Even if this approximation scheme yields promising results, alternative approaches have to be analyzed in the future to obtain better results for specific applications.

The nonlinear equation system (3.36) resulting from the reformulation of the optimal control problem as a two-point boundary-value problem only contains necessary minimum conditions. Therefore, it is desirable to evaluate, whether the obtained solution represents a local or a global

minimum. Another important point to be dealt with, is the improvement of the numerical stability, which causes some problems when solving the nonlinear equation system. An alternative approach to formulate the TPBVP is given by the shooting method resulting in a different equation system, which can be solved numerically. The (multiple) shooting method has been successfully applied in [SOD06] to improve the numerical stability of the solution to the TPBVP. A promising modification of existing solution methods for nonlinear equation systems can be found in [Bra72], where a predictor-corrector scheme is proposed to reduce the problems of singular Jacobians and, therefore, to enlarge the region of convergence. Furthermore, multiple solutions of the nonlinear equation system can be found. This scheme is applicable, for instance to the Newton method, the continuation method, or the Broyden method.

The extension of the considered input-affine examples to other classes of system inputs is one further point to be discussed in the future.

To apply the proposed methods to some experimental setups, the program code has to be optimized and transferred into another programming language. Apart from that, an extension to the n-dimensional case is necessary. After that, the algorithms of this thesis can be employed and analyzed in real experiments.

APPENDIX A

Mathematical Definitions

A.1 The $\otimes$-Operator

The mapping $\otimes(\mathbb{A}, \underline{w})$ with $\mathbb{A} \in (\mathbb{R}^N)^m$ and $\underline{w} \in \mathbb{R}^N$ is defined as

$$\otimes : \begin{cases} (\mathbb{R}^N)^m \times \mathbb{R}^N \to (\mathbb{R}^N)^{m-1} \\ (a_{i_1 \cdots i_m}, w_l) \mapsto \sum\limits_k a_{i_1 \cdots i_{m-1} k} w_k \quad \forall i_j \in \{1, \ldots, N\},\, j \in \{1, \ldots, m\} \end{cases} .$$

Because of the associative law, the $\otimes$-operator can be applied recursively according to

$$\mathbb{A} \underbrace{\otimes \underline{w} \otimes \cdots \otimes \underline{w}}_{M \text{ times}} =: \mathbb{A} \bigotimes_{i=1}^{M} \underline{w} = (\mathbb{A} \otimes \underline{w}) \bigotimes_{i=1}^{M-1} \underline{w} \ .$$

For $\underline{v}, \underline{w} \in \mathbb{R}^N$ the application of the $\otimes$-operator to both vectors is given by the dyadic product (Kronecker product)

$$\underline{v} \otimes \underline{w} = \underline{v}\, \underline{w}^{\mathrm{T}} \ .$$

Therefore,

$$\mathbb{A} \otimes (\underline{v}\, \underline{w}^{\mathrm{T}}) = \mathbb{A} \otimes (\underline{v} \otimes \underline{w})$$

holds [BSMM01].

A.2 Multi-Dimensional Taylor Series Expansion

For $\underline{f} \in \mathcal{C}^\infty$ with $\underline{f} : \mathbb{R}^n \to \mathbb{R}$, the differential operator is defined as

$$\mathbf{D}^i \underline{f} := \frac{\mathrm{d}^i \underline{f}}{\mathrm{d}\underline{x}^i} \ .$$

Then, the multi-dimensional Taylor series expansion of $\underline{f}$ is given by

$$\underline{f}(\underline{x}_0 + \underline{w}) = \sum_{i=0}^{\infty} \frac{\mathbf{D}^i \underline{f}(\underline{x}_0)}{i!} \bigotimes_{j=0}^{i} \underline{w} \ ,$$

where the properties of the $\otimes$-operator of Appendix A.1 have been exploited.

A.3 Topological Space

Definition A.1 (Topological space) A *topological space* T is a tuple $(X, \mathcal{G})$, where X is a set and $\mathcal{G}$ is a collection of subsets of X satisfying the following axioms.

- $\emptyset \in \mathcal{G}$ and $X \in \mathcal{G}$.
- The union of any collection of sets in $\mathcal{G}$ is also in $\mathcal{G}$.
- For $U, V \in \mathcal{G}$ holds $U \cap V \in \mathcal{G}$.

Definition A.2 (Points of a topological space) The elements $\underline{x} \in X$ are called *points* of the topological space T.

Definition A.3 (Topology) The collection $\mathcal{G}$ is a *topology on* X. The sets in $\mathcal{G}$ are the open sets, and their complements in X are the closed sets of the topological space T.

The collection $\mathcal{G}$ of the open sets is closed under arbitrary unions and all finite intersections of sets in $\mathcal{G}$. It can be shown by induction that the intersection of finitely many open sets is open again [Reh05].

A.4 The L^p Spaces

Definition A.4 (The space L^p) For $p \geq 1$, the space $L^p(\Omega)$ consists of those real-valued Lebesgue measurable functions $\underline{f}$ on the set Ω, for which $\|\underline{f}(\underline{x})\|^p$ is Lebesgue integrable, that is,

$$\int_{\Omega} \|\underline{f}(\underline{x})\|^p \, \mathrm{d}\underline{x} < \infty \ .$$

The norm on this space is defined as

$$\|\underline{f}\|_p := \left(\int_{\Omega} \|\underline{f}(\underline{x})\|^p \, \mathrm{d}\underline{x} \right)^{\frac{1}{p}} \ .$$

L^p is a normed linear space, if there is no distinction between functions that are equal almost everywhere, that is, they differ on a set of Lebesgue measure zero (null set) [Lue69].

Definition A.5 (The space L^∞) The space $L^\infty(\Omega)$ is defined as the space of all Lebesgue measurable functions on Ω, which are bounded, except possibly on a null set. Two functions are considered equivalent if they are equal almost everywhere (a.e.).

In other words, $\underline{f} \in L^\infty(\Omega)$ can be regarded as the set

$$\underline{f} = \left\{ \underline{g} : g = f \quad \text{a.e.} \right\} \ .$$

The value $\sup_{\underline{x}\in\Omega} \|\underline{f}(\underline{x})\|$ is different for different functions, which are equivalent to $\underline{f}$. Therefore, the norm of a function $\underline{f} \in L^\infty$ is defined as the *essential supremum* of $\|\underline{f}(\underline{x})\|$, that is,

$$\|\underline{f}\|_\infty := \operatorname{ess\,sup}_{\underline{x}\in\Omega} \|\underline{f}(\underline{x})\| := \inf_{\underline{g}(\underline{x})=\underline{f}(\underline{x}) \text{ a.e.}} \left\{ \sup_{\underline{x}\in\Omega} \underline{g}(\underline{x}) \right\} \quad .$$

Remark A.1 The space L^p, $1 \leq p \leq \infty$ is a Banach space, that is, a complete normed vector space [Lue69].

APPENDIX B

Introduction to Continuation Processes

In many practical applications, iterative processes

$$\underline{x}_{k+1} = \underline{\Phi}(\underline{x}_k)$$

are employed to obtain numerical solutions to a considered problem. Sufficiently well conditioned initial values are required by a majority of these iteration processes. Especially in case of higher-dimensional problems, it is almost impossible to find a good initial value to start the iteration.

Continuation processes represent one possible way to solve this problem. Since a continuation process is employed in this thesis, its basic idea according to [RD83] is introduced in the following.

B.1 Basic Idea

Continuation processes represent a class of methods solving the problems of iterative processes. This is due to the fact that they are global and under some conditions exhaustive, which means that the initial value is not required to be close to the true solution. The main idea of a continuation method can be described as follows. A given problem

$$\underline{G}(\underline{x}) = \underline{0}$$

is embedded into a parameterized family of problems

$$\underline{G}_\gamma(\underline{x}) = \underline{0}\,,\ 0 \leq \gamma \leq 1 \ \ ,$$

where the solution for $\underline{G}_0(\underline{x}) = \underline{0}$ is easy to obtain. The parameterization transforms the problem $\underline{G}_0(\underline{x}) = \underline{0}$ into the original problem

$$\underline{G}_1(\underline{x}) = \underline{0} \quad \Leftrightarrow \quad \underline{G}(\underline{x}) = \underline{0} \ \ ,$$

where the solution to the transformed problem is calculated at each step of the transformation.

B.2 Formal Definition and Construction of Continuation Processes

Definition B.1 (Continuation process) Let two topological spaces[1] X, Y, and a problem $P = (\underline{G}, U)$ be given, where $U \subseteq Y$ and $\underline{G} : X \to Y$. A solution to the problem P is a point $\underline{x} \in X$ with $\underline{G}(\underline{x}) \in U$. Then, a *continuation process* is defined as a continuous mapping $\underline{H} : X \times [0,1] \to Y$ satisfying

1. $\underline{H}(\underline{x}, 1) = \underline{G}(\underline{x})$.
2. There exists at least one $\underline{x}_0 \in X$ with $\underline{H}(\underline{x}_0, 0) \in U$.
3. There exists a continuous curve $\sigma \subseteq X \times [0,1]$ such that $\sigma = (\underline{x}(\gamma), \gamma)$ is a solution to $\underline{H}(\,\cdot\,,\,\cdot\,) \in U$ for all $\gamma \in [0,1]$ with $(\underline{x}(0), 0) = (\underline{x}_0, 0)$.
4. The space $X \times [0,1]$ has a differential structure, in which case the curve $(\underline{x}(\gamma), \gamma)$ is differentiable.

To construct a continuation process $\underline{H}$ for a problem P, the main difficulties arise in satisfying the conditions 3 and 4 of Definition B.1. Examples and more detailed information are given in [RD83].

When the curve σ is obtained, two classes of methods can be employed to follow this curve, that is, discrete and continuous methods.

In case of discrete methods, the unit interval $[0,1]$ is partitioned into finitely many subintervals $[\gamma_i, \gamma_{i+1}]$, $i = 0, \ldots, N-1$. Then, a chain of problems

$$\underline{H}(\underline{x}(\gamma_i), \gamma_i) = 0\,, \quad 0 = \gamma_0 \leq \ldots \leq \gamma_N = 1\ ,$$

is obtained and solved iteratively. Starting from a known solution $\underline{H}(\underline{x}(0), 0)$, $\underline{x}_{i+1}$ is calculated by means of a local iteration scheme, where the solution $\underline{x}_i$ of the previous subinterval is employed as the initial guess of the current interval. The main theoretical problem is to determine conditions on $\underline{H}$, which ensure that such a partition of the unit interval and an iterative process $\underline{\Phi}$ exist, such that $\underline{x}_i$ is in the domain of attraction of $\underline{x}_{i+1} = \underline{\Phi}(\underline{x}_i, \gamma_i)$.

The second method to apply a continuation process is the continuous Davidenko's method. The main idea is to differentiate $\underline{H}(\underline{x}(\gamma), \gamma) = \underline{0}$ with respect to the homotopy parameter γ. Then, Davidenko's differential equation

$$\frac{\mathrm{d}\underline{H}\,(\underline{x}(\gamma), \gamma)}{\mathrm{d}\gamma} = \frac{\partial \underline{H}\,(\underline{x}(\gamma), \gamma)}{\partial \underline{x}} \frac{\mathrm{d}\underline{x}}{\mathrm{d}\underline{\gamma}} + \frac{\partial \underline{H}\,(\underline{x}(\gamma), \gamma)}{\partial \gamma} = \underline{0} \tag{B.1}$$

is obtained. Together with the initial condition $\underline{x}(0) = \underline{x}_0$, this initial value problem can be solved by numerical integration from 0 to 1. The desired solution is obtained for $\gamma = 1$.

In both the discrete and the continuous method, it may happen that the curve σ crosses singularities of $\frac{\partial H}{\partial \underline{x}}$, which can be treated in different ways, for instance by parameterizing the

1 The definition of a topological space is given in appendix A.3.

curve σ appropriately or by selecting a suitable function $\underline{H}\,(\underline{x}(0), 0)$. The advantage of the continuous variant of the curve follower is that it coincides with the integration of an initial value problem. However, a large number of points may be necessary to trace the curve σ from 0 to 1 [RD83].

In this thesis, the discrete variant of the curve follower has been employed. Similar to [TJ79], a continuation method is applied to the optimal control problem for a two-point boundary-value problem, where the process is initialized by a linear problem for $\gamma = 0$. For $\gamma = 1$, the original nonlinear control problem is obtained. In contrast to [TJ79], a discrete-time system is considered, which suffers from the influence of additive noise.

APPENDIX C

Linear Quadratic Control

Referring to [Ber00a] and [Fai98], in this part, the optimal linear state-feedback control for the linear system

$$\underline{\boldsymbol{x}}_{k+1} = \mathbf{A}_k \underline{\boldsymbol{x}}_k + \mathbf{B}_k \underline{u}_k + \underline{\boldsymbol{w}}_k \, , \, k = 0, \ldots, N-1 \ , \tag{C.1}$$

is derived. Equation (C.1) can be rewritten as

$$\underline{\boldsymbol{x}}_{k+1} = \begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{x}}_k \\ \underline{u}_k \end{bmatrix} + \underline{\boldsymbol{w}}_k \ , \tag{C.2}$$

where the state is given by $\underline{x}_k \in \mathbb{R}^n$, and $\underline{u}_k \in \mathbb{R}^m$ denotes the control variable. $\mathbf{A}_k$ and $\mathbf{B}_k$ are given matrices of appropriate dimensions. The disturbances $\underline{\boldsymbol{w}}_k$ are independent zero-mean random vectors with given probability distributions that depend neither on $\underline{x}_k$ nor on $\underline{u}_k$. In the following, the quadratic cost function

$$V(\underline{x}_0) = \mathop{\mathrm{E}}_{\substack{\underline{\boldsymbol{w}}_k \\ k=0,\ldots,N-1}} \left[\underline{\boldsymbol{x}}_N^{\mathrm{T}} \mathbf{Q}_N \underline{\boldsymbol{x}}_N + \sum_{k=0}^{N-1} \underline{\boldsymbol{x}}_k^{\mathrm{T}} \mathbf{Q}_k \underline{\boldsymbol{x}}_k + \underline{u}_k^{\mathrm{T}} \mathbf{R}_k \underline{u}_k \right] \tag{C.3}$$

is considered, which minimizes the output of the system (C.2). The matrices $\mathbf{Q}_k$ are assumed to be symmetric and positive semidefinite, the matrices $\mathbf{R}_k$ are assumed to be symmetric and positive definite. Employing (C.2), the quadratic cost of $\underline{\boldsymbol{x}}_{k+1}$ is given by

$$\begin{aligned} \underline{\boldsymbol{x}}_{k+1}^{\mathrm{T}} \mathbf{Q}_{k+1} \underline{\boldsymbol{x}}_{k+1} &= \left(\begin{bmatrix} \underline{x}_k^{\mathrm{T}} & \underline{u}_k^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{A}_k^{\mathrm{T}} \\ \mathbf{B}_k^{\mathrm{T}} \end{bmatrix} + \underline{\boldsymbol{w}}_k^{\mathrm{T}} \right) \mathbf{Q}_{k+1} \left(\begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \begin{bmatrix} \underline{x}_k \\ \underline{u}_k \end{bmatrix} + \underline{\boldsymbol{w}}_k \right) \\ &= \left(\begin{bmatrix} \underline{x}_k^{\mathrm{T}} & \underline{u}_k^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \\ \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \end{bmatrix} \begin{bmatrix} \underline{x}_k \\ \underline{u}_k \end{bmatrix} \right) \\ &\quad + \left(\underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \begin{bmatrix} \mathbf{A}_k & \mathbf{B}_k \end{bmatrix} \begin{bmatrix} \underline{x}_k \\ \underline{u}_k \end{bmatrix} \right) + \left(\begin{bmatrix} \underline{x}_k^{\mathrm{T}} & \underline{u}_k^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{A}_k^{\mathrm{T}} \\ \mathbf{B}_k^{\mathrm{T}} \end{bmatrix} \mathbf{Q}_{k+1} \underline{\boldsymbol{w}}_k \right) \\ &\quad + \underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \underline{\boldsymbol{w}}_k \ . \end{aligned} \tag{C.4}$$

Calculation of the expectation value of (C.4) yields

$$\mathop{\mathrm{E}}_{\underline{\boldsymbol{w}}_k} \left[\underline{\boldsymbol{x}}_{k+1}^{\mathrm{T}} \mathbf{Q}_{k+1} \underline{\boldsymbol{x}}_{k+1} \right] = \left(\begin{bmatrix} \underline{x}_k^{\mathrm{T}} & \underline{u}_k^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \\ \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \end{bmatrix} \begin{bmatrix} \underline{x}_k \\ \underline{u}_k \end{bmatrix} \right) + \mathop{\mathrm{E}}_{\underline{\boldsymbol{w}}_k} \left[\underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \underline{\boldsymbol{w}}_k \right] \tag{C.5}$$

because of the zero-mean of $\underline{\boldsymbol{w}}_k$. Applying dynamic programming to minimize (C.3), the recursively defined value functions for each time step from N to 0 are given by

$$\begin{aligned} J_N(\underline{x}_N) &= \underline{x}_N^{\mathrm{T}} \mathbf{Q}_N \underline{x}_N \\ J_k(\underline{x}_k) &= \min_{\underline{u}_k} \left(\underline{x}_k^{\mathrm{T}} \mathbf{Q}_k \underline{x}_k + \underline{u}_k^{\mathrm{T}} \mathbf{R}_k \underline{u}_k + \mathop{\mathrm{E}}_{\underline{w}_k} \left[\underline{\boldsymbol{x}}_{k+1}^{\mathrm{T}} \mathbf{Q}_{k+1} \underline{\boldsymbol{x}}_{k+1} \right] \right) \quad . \end{aligned} \tag{C.6}$$

Insertion of (C.5) into (C.6) leads to the value function

$$\begin{aligned} J_k(\underline{x}_k) &= \min_{\underline{u}_k} \left(\underline{x}_k^{\mathrm{T}} \mathbf{Q}_k \underline{x}_k + \underline{u}_k^{\mathrm{T}} \mathbf{R}_k \underline{u}_k + \begin{bmatrix} \underline{x}_k^{\mathrm{T}} & \underline{u}_k^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \\ \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \end{bmatrix} \begin{bmatrix} \underline{x}_k \\ \underline{u}_k \end{bmatrix} \right) \\ &\quad + \mathop{\mathrm{E}}_{\underline{w}_k} \left[\underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \underline{\boldsymbol{w}}_k \right] \\ &= \min_{\underline{u}_k} \left(\begin{bmatrix} \underline{x}_k^{\mathrm{T}} & \underline{u}_k^{\mathrm{T}} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{Q}_k + \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{A}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \\ \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k & \mathbf{R}_k + \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \end{bmatrix}}_{=: \tilde{\mathbf{A}}_k} \begin{bmatrix} \underline{x}_k \\ \underline{u}_k \end{bmatrix} \right) + \mathop{\mathrm{E}}_{\underline{w}_k} \left[\underline{\boldsymbol{w}}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \underline{\boldsymbol{w}}_k \right] \quad . \end{aligned} \tag{C.7}$$

To satisfy the necessary minimum condition, the roots of the partial derivative of (C.7) with respect to $\underline{u}_k$ have to be found, that is,

$$\begin{aligned} &\begin{bmatrix} \underline{0}^{\mathrm{T}} & \underline{1}^{\mathrm{T}} \end{bmatrix} \tilde{\mathbf{A}}_{\mathbf{k}} \begin{bmatrix} \underline{x}_k \\ \underline{u}_k \end{bmatrix} + \begin{bmatrix} \underline{x}_k^{\mathrm{T}} & \underline{u}_k^{\mathrm{T}} \end{bmatrix} \tilde{\mathbf{A}}_{\mathbf{k}} \begin{bmatrix} \underline{0} \\ \underline{1} \end{bmatrix} \\ &= 2 \left(\mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k \underline{x}_k + \left(\mathbf{R}_k + \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \right) \underline{u}_k \right)^{\mathrm{T}} \\ &\stackrel{!}{=} \underline{0}^{\mathrm{T}} \quad . \end{aligned} \tag{C.8}$$

With this minimum condition, the optimal solution $\underline{u}_k^*$ is given by

$$\underline{u}_k^* = - \left(\mathbf{R}_k + \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{B}_k \right)^{-1} \mathbf{B}_k^{\mathrm{T}} \mathbf{Q}_{k+1} \mathbf{A}_k \underline{x}_k \quad . \tag{C.9}$$

Although this equation already yields the optimal control, an interesting property of the optimal feedback gain can be seen, when going to the last time step, that is, the first step of the dynamic programming algorithm. Therefore, let $k = N - 1$. Then, from (C.9) it follows that

$$\underline{u}_{N-1}^* = - \left(\mathbf{R}_{N-1} + \mathbf{B}_{N-1}^{\mathrm{T}} \mathbf{Q}_N \mathbf{B}_{N-1} \right)^{-1} \mathbf{B}_{N-1}^{\mathrm{T}} \mathbf{Q}_N \mathbf{A}_{N-1} \underline{x}_{N-1} \quad . \tag{C.10}$$

Substitution of (C.10) into (C.6) for $k = N - 1$ yields

$$J_{N-1}(\underline{x}_{N-1}) = \underline{x}_{N-1}^{\mathrm{T}} \mathbf{K}_{N-1} \underline{x}_{N-1} + \mathop{\mathrm{E}}_{\underline{w}_N} \left[\underline{\boldsymbol{w}}_N^{\mathrm{T}} \mathbf{Q}_N \underline{\boldsymbol{w}}_N \right] \quad ,$$

that is,

$$\underline{x}_{N-1}^{\mathrm{T}} \mathbf{K}_{N-1} \underline{x}_{N-1} = \min_{\underline{u}_{N-1}} \left(\begin{bmatrix} \underline{x}_{N-1}^{\mathrm{T}} & \underline{u}_{N-1}^{\mathrm{T}} \end{bmatrix} \tilde{\mathbf{A}}_{N-1} \begin{bmatrix} \underline{x}_{N-1} \\ \underline{u}_{N-1} \end{bmatrix} \right) \quad . \tag{C.11}$$

The matrix $\mathbf{K}_{N-1}$ is given by

$$\mathbf{K}_{N-1} = \mathbf{A}_{N-1}^{\mathrm{T}} \left(\mathbf{Q}_N - \mathbf{Q}_N \mathbf{B}_{N-1} \left(\mathbf{B}_{N-1}^{\mathrm{T}} \mathbf{Q}_N \mathbf{B}_{N-1} + \mathbf{R}_{N-1} \right)^{-1} \mathbf{B}_{N-1}^{\mathrm{T}} \mathbf{Q}_N \right) \mathbf{A}_{N-1} + \mathbf{Q}_{N-1} \quad . \tag{C.12}$$

Moreover, $\mathbf{K}_{N-1}$ is symmetric and positive semidefinite, which follows from (C.11) and the assumptions on $\mathbf{Q}_{N-1}, \mathbf{Q}_N$, and $\mathbf{R}_{N-1}$. Therefore, J_{N-1} is positive semidefinite and quadratic[1]. Hence, a recursion

$$\begin{aligned}
\mathbf{K}_N &= \mathbf{Q}_N \ , \\
\mathbf{K}_k &= \mathbf{A}_k^{\mathrm{T}} \left(\mathbf{K}_{k+1} - \mathbf{K}_{k+1}\mathbf{B}_k \left(\mathbf{B}_k^{\mathrm{T}}\mathbf{K}_{k+1}\mathbf{B}_k + \mathbf{R}_k \right)^{-1} \mathbf{B}_k^{\mathrm{T}}\mathbf{K}_{k+1} \right) \mathbf{A}_k + \mathbf{Q}_k \ , \ k = N-1, \ldots, 0 \ ,
\end{aligned} \tag{C.13}$$

$$\begin{aligned}
\mathbf{L}_k &= \left(\mathbf{B}_k^{\mathrm{T}}\mathbf{K}_{k+1}\mathbf{B}_k + \mathbf{R}_k \right)^{-1} \mathbf{B}_k^{\mathrm{T}}\mathbf{K}_{k+1}\mathbf{A}_k \ , \ k = N-1, \ldots, 0 \ , \\
\underline{u}_k^* &= \underline{\mu}_k(\underline{x}_k) = -\mathbf{L}_k(\underline{x}_k) \ , \ k = N-1, \ldots, 0 \ ,
\end{aligned} \tag{C.14}$$

can be defined to determine the optimal state-feedback control law for $k = N-1, \ldots, 0$ Then, the value of the value function just depends on the initial state, the noise term $\underline{\boldsymbol{w}}_k$, and the matrices $\mathbf{K}_k$. Thus,

$$J(\underline{x}_0) = \underline{x}_0^{\mathrm{T}}\mathbf{K}_0\underline{x}_0 + \sum_{k=0}^{N-1} \underset{\underline{w}_k}{\mathrm{E}} \left[\underline{\boldsymbol{w}}_k^{\mathrm{T}}\mathbf{K}_{k+1}\underline{\boldsymbol{w}}_k \right] \ .$$

Remark C.1 Equation (C.13) is called the *discrete-time algebraic Riccati equation*. If $\mathbf{K}_k > \mathbf{0}$ for all symmetric positive definite matrices $\mathbf{Q}_k > \mathbf{0}$, $\mathbf{K}_k$ is unique, asymptotically stabilizing, and solves

$$\mathbf{A}_k\mathbf{K}_{k+1}\mathbf{A}_k^{\mathrm{T}} - \mathbf{K}_k + \mathbf{Q}_k = \mathbf{0} \ . \tag{C.15}$$

Equation (C.15) is called the *discrete-time Lyapunov equation*.

1 This property does not hold for nonlinear systems.

APPENDIX D

Unscented Transformation

D.1 Motivation

To treat nonlinear transformations of random variables, sophisticated analysis is inevitable. Unfortunately, exact analytic solutions are impossible to obtain in general. One approximation method to obtain a moment-based description of the density of a nonlinearly transformed random variable

$$\underline{\boldsymbol{y}} = \underline{f}(\underline{\boldsymbol{x}}) \tag{D.1}$$

is the unscented transformation (UT) introduced in [JU96]. Instead of approximating the nonlinear function $\underline{f}$, the density of the random variable $\underline{\boldsymbol{x}}$ is approximated with a fixed number of parameters. The main idea to obtain estimates of the mean and the covariance of $\underline{\boldsymbol{y}}$ and several properties of this algorithm are discussed in the following, where only the prediction step is considered, since the filter step of the UT is not employed in this thesis.

Convention. Without loss of generality, in the following, it is assumed that the mean value of the random variables to be transformed is zero.

D.2 Basic Idea

A set $\mathcal{X}$ of $p+1$ sampling points $\mathcal{X}_i$ of the original density of the random variable $\underline{\boldsymbol{x}}$ and corresponding weights ω_i is generated, which possesses the same mean, covariance, and possibly higher-order moments as the original density of $\underline{\boldsymbol{x}}$. Then, each sampling point $\mathcal{X}_i$ of the set $\mathcal{X}$ is nonlinearly transformed onto a set

$$\mathcal{Y} := \{\mathcal{Y}_i : \mathcal{Y}_i = \underline{f}(\mathcal{X}_i),\, \omega_i\,,\ i = 0, \ldots, p\}$$

by means of (D.1) as illustrated by Figure D.1. Mean and covariance of $\mathcal{Y}$ can be regarded as an estimate of the true moments of the random variable $\underline{\boldsymbol{y}}$ in (D.1).

In contrast to particle filters, the small set of the sampling points to be transformed, the so called *sigma points*, is chosen deterministically, such that mean and covariance of the random

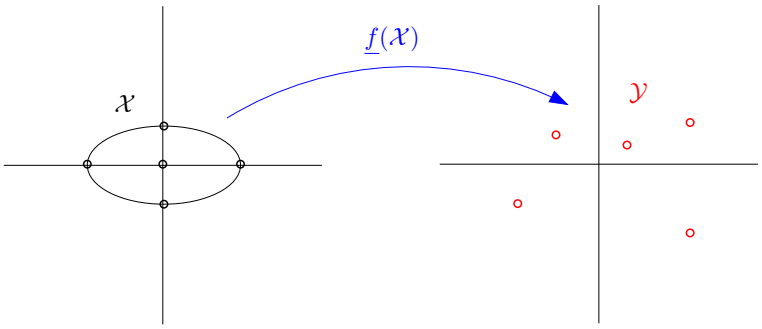


Figure D.1: Principle of the unscented transformation. A specific set of sampling points is nonlinearly transformed. The moments of the resulting set of transformed points approximate the true moments of the nonlinearly transformed random variable.

variable $\underline{\boldsymbol{x}}$ are matched. Moreover, the set of sigma points is weighted, where negative weights are possible.[1] To provide an unbiased estimate, the weights must satisfy

$$\sum_{i=0}^{p} \omega_i = 1 \ . \tag{D.2}$$

The estimates of mean and covariance of $\underline{\boldsymbol{y}} = \underline{f}(\underline{\boldsymbol{x}})$ are obtained as described in Algorithm 5. After the pointwise transformation of all $p+1$ points $\mathcal{X}_i$, the mean and the covariance of the nonlinearly transformed random variable $\underline{\boldsymbol{y}}$ can be calculated approximately.

Algorithm 5 Unscented transformation

1: **function** MEAN_AND_COVARIANCE_PREDICTION$(\mathcal{X}_0, \ldots, \mathcal{X}_p, \omega_0, \ldots, \omega_p, \underline{f})$
2: **for** $i = 0$ to p **do**
3: $\mathcal{Y}_i = \underline{f}(\mathcal{X}_i)$ ▷ propagation of the sigma points
4: **end for**
5: ▷ mean prediction

$$\overline{\underline{y}} = \mathrm{E}[\underline{\boldsymbol{y}}] \approx \mathrm{E}[\mathcal{Y}] = \sum_{i=0}^{p} \omega_i \mathcal{Y}_i \tag{D.3}$$

6: ▷ covariance prediction

$$\boldsymbol{\Sigma}_y \approx \sum_{i=0}^{p} \omega_i \left(\mathcal{Y}_i - \mathrm{E}[\mathcal{Y}]\right) \left(\mathcal{Y}_i - \mathrm{E}[\mathcal{Y}]\right)^{\mathrm{T}} \tag{D.4}$$

7: **end function**

In case of additive noise in the transformation (D.1), that is,

$$\underline{\boldsymbol{y}} = \underline{f}(\underline{\boldsymbol{x}}) + \underline{\boldsymbol{w}} \ ,$$

1 Therefore, the samplings do *not* define a discretized density.

an additional term has to be incorporated in the calculation of the covariance matrix [XZC06], which leads to

$$\mathbf{\Sigma}_y \approx \sum_{i=0}^{p} \omega_i \left(\mathcal{Y}_i - \mathrm{E}[\mathcal{Y}]\right)\left(\mathcal{Y}_i - \mathrm{E}[\mathcal{Y}]\right)^{\mathrm{T}} + \mathbf{\Sigma}_w \ .$$

D.3 Accuracy

According to [JU96] and [JU04], this section analyzes the accuracy of the estimates of means and covariances of a nonlinearly transformed random variable by means of the unscented transformation.

D.3.1 Mean Value

The mean value of $\underline{\boldsymbol{y}}$ in (D.1) can be written as

$$\overline{\underline{y}} = \underset{\underline{\delta}}{\mathrm{E}}\left[\underline{f}(\overline{\underline{x}} + \underline{\boldsymbol{\delta}})\right] \ , \tag{D.5}$$

since the random variable $\underline{\boldsymbol{x}}$ can be split into a deterministic part, that is $\overline{\underline{x}} = \underset{\underline{\delta}}{\mathrm{E}}[\underline{\boldsymbol{x}}]$, and a stochastic part $\underline{\boldsymbol{\delta}}$ with mean $\underline{0}$.[2] According to Appendix A.2, Taylor series expansion of (D.5) around the nominal value $\overline{\underline{x}}$ results in

$$\overline{\underline{y}} = \underline{f}(\overline{\underline{x}}) + \underset{\underline{\delta}}{\mathrm{E}}\left[\sum_{i=1}^{\infty} \frac{1}{i!} \frac{\mathrm{d}\underline{f}(\overline{\underline{x}})}{\mathrm{d}\underline{x}} \bigotimes_{j=1}^{i} \underline{\boldsymbol{\delta}}\right] = \underline{f}(\overline{\underline{x}}) + \sum_{i=1}^{\infty} \underset{\underline{\delta}}{\mathrm{E}}\left[\frac{1}{i!} \frac{\mathrm{d}\underline{f}(\overline{\underline{x}})}{\mathrm{d}\underline{x}} \bigotimes_{j=1}^{i} \underline{\boldsymbol{\delta}}\right] \ . \tag{D.6}$$

With the definition

$$\mathbf{D}^i_{\delta}\underline{f}(\overline{\underline{x}}) := \frac{\mathrm{d}\underline{f}(\overline{\underline{x}})}{\mathrm{d}\underline{x}} \bigotimes_{j=1}^{i} \underline{\boldsymbol{\delta}} \ ,$$

the expectation value in the sum of (D.6) can be rewritten as

$$\underset{\underline{\delta}}{\mathrm{E}}\left[\frac{\mathbf{D}^i_{\delta}\underline{f}(\overline{\underline{x}})}{i!}\right] = \frac{1}{i!} \underset{\underline{\delta}}{\mathrm{E}}\left[\left(\sum_{j=1}^{n} \boldsymbol{\delta}_j \frac{\partial \underline{f}(\overline{\underline{x}})}{\partial x_j}\right)^i\right] \ , \tag{D.7}$$

where $\boldsymbol{\delta}_j$ and x_j denote the components of the vectors $\underline{\boldsymbol{\delta}}$ and $\underline{x}$, respectively.

Theorem D.1 *If the first k moments of the random variable $\underline{\boldsymbol{\delta}}$ and the first k derivatives of the transformation $\underline{f}$ are known, the UT-approximation of $\overline{\underline{y}}$ by means of*

$$\overline{\underline{y}} \approx \sum_{i=0}^{2p} \omega_i \underline{f}\left(\mathcal{X}_i\right) \tag{D.8}$$

yields correct results up to the k-th order term of (D.6).

Proof. The proof of Theorem D.1 is given in [JU96]. □

2 Without loss of generality, the mean value of $\underline{\boldsymbol{\delta}}$ is set to $\underline{0}$.

Remark D.1 The standard UT-algorithm considers only mean value and covariance to be captured by the set $\mathcal{X}$. But if higher-order moments of $\underline{\boldsymbol{x}}$ are known, this knowledge can be employed to capture higher-order moments. Therefore, the expectation value of $\underline{\boldsymbol{x}}$ is at least accurate, but not restricted to second-order terms of the Taylor series expansion of the nonlinear transformation of $\underline{\boldsymbol{x}}$, which is revealed by (D.6). In case of symmetric densities of $\underline{\boldsymbol{x}}$, all terms in (D.6) vanish, which contain odd-order moments.

Example D.1: Influence of the knowledge of derivatives and moments in the UT for $n = 2$
For $n = 2$, the inner term of (D.7) can be reformulated by means of the Binomial theorem [BSMM01] as

$$\left(\sum_{j=1}^{2} \boldsymbol{\delta}_j \frac{\partial \underline{f}(\overline{x})}{\partial x_j}\right)^i = \sum_{k=0}^{i} \binom{i}{k} \boldsymbol{\delta}_1^k \left(\frac{\partial \underline{f}(\overline{x})}{\partial x_1}\right)^k \boldsymbol{\delta}_2^{i-k} \left(\frac{\partial \underline{f}(\overline{x})}{\partial x_2}\right)^{i-k} \quad , \tag{D.9}$$

which is a polynomial of degree two, in both the partial derivative and the term $\boldsymbol{\delta}_j$. This means, the calculation of (D.7) requires the knowledge of the first i moments of the random variable $\underline{\boldsymbol{\delta}}$ and the first i derivatives of the function $\underline{f}$. ■

The set $\mathcal{X}$ in the UT matches at least the first two moments of the random variable $\underline{\boldsymbol{x}}$. Moreover, the nonlinear transformation $\underline{f}$ is known.

D.3.2 Covariance

The covariance of $\underline{\boldsymbol{y}}$ is defined as

$$\boldsymbol{\Sigma}_y := \mathrm{Cov}(\underline{\boldsymbol{y}}, \underline{\boldsymbol{y}}) := \mathop{\mathrm{E}}_{\underline{\delta}} \left[(\underline{\boldsymbol{y}} - \overline{\underline{y}})(\underline{\boldsymbol{y}} - \overline{\underline{y}})^{\mathrm{T}}\right] \quad , \tag{D.10}$$

where Taylor series expansion of

$$\underline{\boldsymbol{y}} - \overline{\underline{y}} = \underline{f}(\overline{x} + \underline{\boldsymbol{\delta}}) - \mathop{\mathrm{E}}_{\underline{\delta}} \left[\underline{f}(\overline{x} + \underline{\boldsymbol{\delta}})\right] \tag{D.11}$$

yields

$$\mathbf{D}_\delta \underline{f}(\overline{x}) + \frac{\mathbf{D}_\delta^2 \underline{f}(\overline{x})}{2!} + \frac{\mathbf{D}_\delta^3 \underline{f}(\overline{x})}{3!} + \frac{\mathbf{D}_\delta^4 \underline{f}(\overline{x})}{4!} + \cdots - \mathop{\mathrm{E}}_{\underline{\delta}} \left[\frac{\mathbf{D}_\delta^2 \underline{f}(\overline{x})}{2!} + \frac{\mathbf{D}_\delta^4 \underline{f}(\overline{x})}{4!} + \cdots\right] \tag{D.12}$$

in case of a symmetric density of $\underline{\boldsymbol{x}}$. Therefore, (D.10) can be written as

$$\begin{aligned}
\boldsymbol{\Sigma}_y = {} & \mathop{\mathrm{E}}_{\underline{\delta}} \left[\mathbf{D}_\delta \underline{f}\, \underline{f}^{\mathrm{T}} \mathbf{D}_\delta^{\mathrm{T}}\right] + \mathop{\mathrm{E}}_{\underline{\delta}} \left[\frac{\mathbf{D}_\delta \underline{f} \left(\mathbf{D}_\delta^2 \underline{f}\right)^{\mathrm{T}}}{2!} + \frac{\mathbf{D}_\delta^2 \underline{f} \left(\mathbf{D}_\delta \underline{f}\right)^{\mathrm{T}}}{2!}\right] \\
& + \mathop{\mathrm{E}}_{\underline{\delta}} \left[\frac{\mathbf{D}_\delta \underline{f} \left(\mathbf{D}_\delta^3 \underline{f}\right)^{\mathrm{T}}}{3!} + \frac{\mathbf{D}_\delta^2 \underline{f} \left(\mathbf{D}_\delta^2 \underline{f}\right)^{\mathrm{T}}}{2! \cdot 2!} + \frac{\mathbf{D}_\delta^3 \underline{f} \left(\mathbf{D}_\delta \underline{f}\right)^{\mathrm{T}}}{3!}\right] \\
& - \mathop{\mathrm{E}}_{\underline{\delta}} \left[\frac{\mathbf{D}_\delta^2 \underline{f}}{2}\right] \mathop{\mathrm{E}}_{\underline{\delta}} \left[\frac{\mathbf{D}_\delta^2 \underline{f}}{2}\right]^{\mathrm{T}} + \cdots \quad .
\end{aligned} \tag{D.13}$$

Theorem D.2 *If the first $2k$ moments of the random variable $\underline{\boldsymbol{\delta}}$ and the first $2k$ derivatives of the function $\underline{f}$ are known, the UT-approximation of the covariance $\boldsymbol{\Sigma}_y$ by means of*

$$\boldsymbol{\Sigma}_y \approx \sum_{i=0}^{p} \omega_i \left(\underline{\overline{y}} - \mathcal{Y}_i\right) \left(\underline{\overline{y}} - \mathcal{Y}_i\right)^{\mathrm{T}} \tag{D.14}$$

yields correct results up to the k-th order term of (D.13).

Proof. The proof of Theorem D.2 can be found in [JU96]. □

Remark D.2 For any set of sigma points satisfying (D.2)–(D.4), the transformed mean and covariance are calculated correctly to the second-order terms of (D.13), [JU04].

D.4 Incorporation of Knowledge of Higher-Order Moments

Knowledge of higher-order moments can be partially incorporated into the choice of the sigma point set. Since the maintenance of the full density of a random variable is in general intractable, a sufficiently good and tractable approximation P^x is assumed, which captures the most significant features of the true density, for instance a Gaussian mixture density. In this case, P^x can be used as a basis for a good approximate solution to the nonlinear transformation problem. Since the sigma point set and the corresponding weights only have to satisfy (D.2)–(D.4), there exist free parameters, which can be employed to capture higher-order information partially by minimizing the error of these moments subject to (D.2)–(D.4). The more sigma points are chosen, the more information can be captured. To satisfy (D.2)–(D.4), at least $n+1$ affinely independent sigma points are required for $\underline{x} \in \mathbb{R}^n$ [JU02]. This simplex possesses some degrees of freedom, which can be exploited to minimize the average skew (third moment) of the density. Since no information about the (a)symmetry of the density is maintained by the mean and covariance, the density can be skewed in any direction. Therefore, the average error is minimized if the density is assumed to be symmetric. With increasing dimension, the influence of higher-order moments becomes more significant than in lower dimensions. Instead of employing a set of $n+1$ sigma points, an extended symmetric set of $2n+1$ sigma points is proposed in [JU04] to minimize the worst case error due to the skew of the density of $\underline{\boldsymbol{x}}$. According to [JU04], this symmetric set of $p+1 = 2n+1$ sigma points with corresponding weights for $\underline{x} \in \mathbb{R}^n$ is chosen as

$$\begin{aligned}
\mathcal{X}_0 &= \underline{\overline{x}} \ , \\
\omega_0 &= \omega_0 \text{ (free parameter)} \ , \\
\mathcal{X}_i &= \mathcal{X}_0 + \left(\sqrt{\frac{n}{1-\omega_0}\boldsymbol{\Sigma}_x}\right)_i \ , \\
\omega_i &= \frac{1-\omega_0}{2n} \ ,
\end{aligned}$$

$$\mathcal{X}_{i+n} = \mathcal{X}_0 - \left(\sqrt{\frac{n}{1-\omega_0}\boldsymbol{\Sigma}_x}\right)_i \ ,$$
$$\omega_{i+n} = \frac{1-\omega_0}{2n} \ ,$$

where $i = 1, \ldots, n$. $\left(\sqrt{\frac{n}{1-\omega_0}\boldsymbol{\Sigma}_x}\right)_i$ denotes the i-th column of a matrix square root of $\frac{n}{1-\omega_0}\boldsymbol{\Sigma}_x$. The free parameter ω_0 can be chosen, such that information about higher-order moments of the random variable $\underline{\boldsymbol{x}}$ is included [JU04]. For instance, with

$$\omega_0 = \frac{\kappa}{n+\kappa} \, , \ \kappa \in \mathbb{R} \ ,$$

the parameter κ scales the third- and higher-order terms of the sigma point set. In case of a Gaussian density, even some of the fourth-order terms are captured for $n + \kappa = 3$ [JU02].

With increasing dimension, the radius of the sigma point set increases as well, such that non-local effects are sampled. Furthermore, the influence of higher-order terms in (D.6) and (D.13) becomes more significant than in lower dimensions. A scaling algorithm treats this problem by eliminating higher-order, orientation-dependent effects, whereby the second-order accuracy of the UT is maintained [Jul02]. This scaling parameter adjusts the distance between the sigma points. A given sigma point $\mathcal{X}_i^{(k)}$ is moved toward or away from the mean value $\underline{\overline{x}}_k = \mathcal{X}_0^{(k)}$ according to

$$\tilde{\mathcal{X}}_i^{(k)} = \mathcal{X}_0^{(k)} + \alpha(\mathcal{X}_i^{(k)} - \mathcal{X}_0^{(k)}) \ .$$

For $\alpha = 1$, the original point $\mathcal{X}_i^{(k)}$ is obtained. The corresponding weights have to be modified to satisfy the preconditions of the UT and are given by

$$\tilde{\omega}_j = \begin{cases} \frac{\omega_0+\alpha^2-1}{\alpha^2} & \text{for } j = 0 \ , \\ \frac{\omega_j}{\alpha^2} & \text{for } j \neq 0 \ . \end{cases} \tag{D.15}$$

Remark D.3 The scaled UT keeps the accuracy of the estimated values up to second-order terms in (D.6) and (D.13), [Jul02]. On the one hand, for small values of α, local features of the density of the considered state can be captured, and the influence of higher-order terms can be reduced. On the other hand, global features of the density cannot be incorporated in the calculation. Higher-order terms in the Taylor series expansions (D.6) and (D.13) almost vanish, if the scaling parameter is small. In case of known higher-order moments, this knowledge cannot be incorporated in the prediction of mean and covariance, which may lead to worse results compared to the unscaled variant of the UT. This behavior is demonstrated in Section 4.5 by means of a scalar example. Therefore, a tradeoff between these properties has to be found depending on the specific application.

D.5 Properties, Limitations, and Extensions

Compared with the extended Kalman filter, the unscented transformation yields good approximations of the mean and covariance of a transformed random variable $\underline{\boldsymbol{y}} = \underline{f}(\underline{\boldsymbol{x}})$ based on the

corresponding values of $\underline{\boldsymbol{x}}$ [JU97, vdMDdFW00, JUDW00, JU04, XZC06, CHL05]. Compared to the extended Kalman filter, the results are more accurate, since the EKF just linearizes the function $\underline{f}$ and does not include information of higher-order derivatives. Therefore, the Taylor series expansions (D.6) and (D.13) are truncated after the first-order term, when the EKF is applied. With additional points in the sigma point set, the unscented transformation can capture and propagate higher-order information of the density [LBS02]. Then, higher-order errors in the approximation of the transformed density can be minimized. One further advantage of the unscented transformation is that the existence of inverse Jacobians is not necessary to be guaranteed. Another point is that the unscented transformation can treat arbitrarily distributed random variables and and arbitrary nonlinear transformations, in principle. The computational efforts of the UT and the EKF are of the same order [JU04]. In contrast to particle filters, the set of sigma points is small and deterministically chosen as described in Section D.2. Especially for (but not restricted to) Gaussian random variables, the unscented transformation is a promising approach and superior to the commonly used extended Kalman filter [vdMW04].

Unfortunately, a moment-based description of a non-Gaussian density does not give any hints about the shape of the density or the number of its modes, that is, whether the function is unimodal (like the Gauss function) or multimodal. Therefore, it will help to approximate the density of a non-Gaussian random variable $\underline{\boldsymbol{x}}$ by means of Gaussian mixtures, that is, the density is approximated by a convex combination of Gaussian functions. This approximation can be performed for arbitrary densities, since the Gaussian mixture is a universal approximator [HS05]. Then, the UT can be applied to each component, and the results are weighted to obtain the desired values, that is mean and covariance, of the transformed random variable. The main problem is to obtain the desired parameters of each Gaussian components to approximate the original function sufficiently well. A sophisticated framework to approximate densities is introduced in [HBR03]. The main idea is to solve the problem for a linear function. After that, a continuation process with a continuous curve follower is employed to derive a system of ordinary differential equations to be solved. The solution minimizes the Kullback-Leibler divergence between the original unknown function and its approximation by means of Gaussian mixtures.

APPENDIX E

Splines

E.1 n-Dimensional Cubic Spline

An n-dimensional interpolating cubic spline

$$\underline{S}(\underline{x}) = \underline{S}(x_1, \ldots, x_n)$$

is the extension of the one-dimensional case using a tensor product approach [BSMM01].

$$\underline{S}(\underline{x}) = \underline{S}_{i_1 \ldots i_n}(\underline{x}) = \sum_{k_1=0}^{3} \cdots \sum_{k_n=0}^{3} a_{i_1 \ldots i_n k_1 \ldots k_n} (x_1 - \lambda_{i_1})^{k_1} \cdots (x_n - \lambda_{i_n})^{k_n} \ , \tag{E.1}$$

where $\underline{\lambda} = (\lambda_{i_1}, \ldots, \lambda_{i_n})^{\mathrm{T}}$ denotes the coordinate vector of each knot. The extension of the properties (3.38)–(3.41) of the one-dimensional case to the n-dimensional case is straightforward. Considering linearly independent systems of functions $\underline{g}_{i_1}(\underline{x}), \ldots, \underline{g}_{i_n}(\underline{x})$, (E.1) can be rewritten as

$$\underline{S}(\underline{x}) = \sum_{k_1=0}^{3} \cdots \sum_{k_n=0}^{3} a_{i_1 \ldots i_n} \underline{g}_{i_1}(\underline{x}) \cdots \underline{g}_{i_n}(\underline{x}) \ . \tag{E.2}$$

Employing the $\otimes$-operator introduced in Appendix A.1, (E.2) can be written as

$$\underline{S}(\underline{x}) = \mathbb{A} \otimes \underline{g}_{i_1}(\underline{x}) \otimes \cdots \otimes \underline{g}_{i_n}(\underline{x}) \ ,$$

where $\mathbb{A} \in (\mathbb{R}^4)^n$ denotes the tensor of the coefficients from (E.2).

E.2 Example of the Error Bound of a Scalar Cubic Spline Interpolant

In Subsection 3.1.6, an error bound for a cubic spline interpolant has been derived. This error bound is calculated on the basis of system (4.1) in the following.

With

$$f(x_k, u_k) = \sin(q\, x_k) + u_k \tag{E.3}$$

and

$$g_N(x_N) = (x_k - c)^2 \tag{E.4}$$

$$g_k(x_k, u_k^*) = \frac{1}{2}(x_k - c)^2 + \frac{1}{2}\, a\, (u_k^*)^2 \ , \tag{E.5}$$

where $c = 0$ and $a = 2$, it follows that

$$\begin{aligned}
\frac{\partial^2 g_k}{\partial x_k^2} &= 1 \\
\frac{\partial f}{\partial x_k} &= q\, \cos(q\, x_k) \\
\frac{\partial^2 f}{\partial x_k^2} &= -q^2\, \sin(q\, x_k) \\
\frac{\mathrm{d}J_{k+1}^{spline}}{\mathrm{d}x_{k+1}} &= 3a_i(x_{k+1} - \lambda_i)^2 + 2b_i(x_{k+1} - \lambda_i) + c_i \\
\frac{\mathrm{d}^2 J_{k+1}^{spline}}{\mathrm{d}x_{k+1}^2} &= 6a_i(x_{k+1} - \lambda_i) + 2b_i
\end{aligned}$$

and

$$\begin{aligned}
\left\| \frac{\partial^2 J_k}{\partial x_k^2} \right\|_\infty \leq & \big\| 1 + (6a_i(x_{k+1} - \lambda_i) + 2b_i)\ q^2 \cos(q\, x_k)^2 \\
& - \left(3a_i(x_{k+1} - \lambda_i)^2 + 2b_i(x_{k+1} - \lambda_i) + c_i\right)\ q^2\ \sin(q\, x_k) \big\|_\infty \\
= & \big\| 1 + q^2 \left[(6a_i(x_{k+1} - \lambda_i) + 2b_i) \cos(q\, x_k)^2 \right. \\
& \left. - \left(3a_i(x_{k+1} - \lambda_i)^2 + 2b_i(x_{k+1} - \lambda_i) + c_i\right) \sin(q\, x_k) \right] \big\|_\infty
\end{aligned}$$

Considering the maximal values of sine and cosine, it follows that

$$\left\| \frac{\partial^2 J_k}{\partial x_k^2} \right\|_\infty \leq \left\| 1 + q^2 \left[6a_i(x_{k+1} - \lambda_i) + 2b_i - 3a_i(x_{k+1} - \lambda_i)^2 + 2b_i(x_{k+1} - \lambda_i) + c_i \right] \right\|_\infty \ .$$

Due to the piecewise definition of the cubic spline, the maximal interval length is defined as

$$\|x_{k+1} - \lambda_i\|_\infty \leq \max_j \lambda_{j+1} - \lambda_j =: h \ . \tag{E.6}$$

Then, with (E.6) it holds that

$$\left\| \frac{\partial^2 J_k}{\partial x_k^2} \right\|_\infty \leq (1 + q^2) \left\| h^2\, (-3\, a_i) + h\, (6\, a_i - 2\, b_i) + 2\, b_i + c_i \right\|_\infty \ .$$

Employing the triangular inequality for norms, a new upper bound for the second derivative of the value function is given by

$$\left\| \frac{\partial^2 J_k}{\partial x_k^2} \right\|_\infty \leq (1 + q^2) \left(h^2 |3\, a_i| + h\, (|6\, a_i| + |2\, b_i|) + |2\, b_i + c_i| \right) =: C \ . \tag{E.7}$$

Therefore, an upper bound for the error between the value function and its spline approximation is given by

$$\begin{aligned}\|(J_k^{spline} - J_k)^{(0)}\|_\infty &\overset{(3.43)}{\leq} \varepsilon_{20}\|f^{(2)}\|_\infty h^{2-0} + K_2\beta_0\left(2^{1-j} - 2^{1-N+j}\right)\, h \\ &= \frac{9}{8}\,\|J_k^{(2)}\|_\infty\, h^2 + \left(\frac{5}{2}\|J_k^{(2)}\|_\infty + R\right)\frac{\Delta x_j}{4}\left(2^{1-j} - 2^{1-N+j}\right)\, h \\ &\overset{(E.7)}{\leq} \frac{9}{8}\,C\,h^2 + \left(\frac{5}{2}\,C + R\right)\frac{\Delta x_j}{4}\left(2^{1-j} - 2^{1-N+j}\right)\, h\ . \end{aligned} \tag{E.8}$$

The remaining problem is to find an upper bound for the value

$$R = \max\left\{\|J_k^{(2)}(\lambda_0) - (J_k^{spline})^{(2)}(\lambda_0)\|,\, \|J_k^{(2)}(\lambda_N) - (J_k^{spline})^{(2)}(\lambda_N)\|\right\}\ .$$

Because of that, an expression for R will be derived. Moreover, the resulting error bound converges to zero with order four for $h \to 0$. Since

$$R \leq \|(J_k - J_k^{spline})^{(2)}\|_\infty\ , \tag{E.9}$$

(3.43) can be applied to (E.9). Therefore,

$$\begin{aligned} R \leq& \|(J_k - J_k^{spline})^{(2)}\|_\infty \overset{r=2=m}{=} \varepsilon_{22}\|J_k^{(2)}\|_\infty + K_2\beta_2(2^{1-j} - 2^{1-N+j})\,h \\ \overset{(E.7)}{\leq}& 10\,C + \frac{5}{2}\,C\frac{6\,h}{\Delta x_j}\left(2^{1-j} - 2^{1-N+j}\right) + R\frac{6\,h}{\Delta x_j}\left(2^{1-j} - 2^{1-N+j}\right)\ . \end{aligned}$$

Thus, the inequality

$$R\left(1 - \frac{6}{\Delta x_j}\,h\left(2^{1-j} - 2^{1-N+j}\right)\right) \leq C\left(10 + \frac{30}{2\Delta x_j}\,h\left(2^{1-j} - 2^{1-N+j}\right)\right)$$

for R can be derived. Because of that,

$$|R| \leq \left|\frac{C\left(10 + \frac{15}{\Delta x_j}\,h\left(2^{1-j} - 2^{1-N+j}\right)\right)}{1 - \frac{6}{\Delta x_j}\,h\left(2^{1-j} - 2^{1-N+j}\right)}\right| =: D\ . \tag{E.10}$$

With (E.10), (E.8) can be evaluated, and it can be seen that (E.8) is a function, which is quadratic in h, since

$$\Delta x_j \leq h\ .$$

Therefore, the limit of $h \to 0$ yields the result

$$\|(J_k^{spline} - J_k)\|_\infty \in \mathcal{O}(h^2)\,,\, h \to 0\ , \tag{E.11}$$

which coincides with the result in [Hal73]. But in contrast to [Hal73], an approximation error for the considered system is presented, which can be calculated without taking the limit.

APPENDIX F

Implementation Details

In this chapter, the main structure of the program code is described, and specific chosen parameters are mentioned. The algorithms are implemented in MATLAB 7.1 [Mat05]. In case of examples for several steps, the system

$$\boldsymbol{x}_{k+1} = \sin(q\,\boldsymbol{x}_k) + u_k + \boldsymbol{w}_k \ ,$$

which is described in detail in Chapter 4, is employed.

Convention. This chapter is restricted to the scalar case to simplify imagination. Moreover, the systems are simulated for a scalar case.

The procedure simulating the system is implemented in `system_sim.m`. After determining the N-step finite horizon window, MPC is performed until a predefined end time *sim_time*, when the simulation is stopped.

Algorithm 6 System simulation

procedure SYSTEM_SIM($\hat{x}_0$, *horizon*, σ_w, *sim_time*)
  $N = horizon$ ▷ length of finite horizon window for MPC
  for $k = 0$ to *sim_time* **do**
    $state\_init = \hat{x}_k$ ▷ initial state of current simulation step
    ▷ call continuation function
    $u\_init\_sequence(k : k+N-1)$=`continuation`$(state\_init, horizon, stepsize, \sigma_w)$
    ▷ call spline based algorithm
    u_k^*=`update_control_main`$(u\_init\_sequence(k : k+N-1), state\_init, \sigma_w)$
    $\boldsymbol{x}_{k+1} = f(\hat{x}_k, u_k^*) + \boldsymbol{w}_k$ ▷ successor state
  end for
end procedure

The parameter σ_w denotes the standard deviation of the noise affecting the system. Starting from the initial value $\hat{x}_0$, the optimal control sequence of the initial solution is obtained by

means of a continuation process with a discrete curve follower as explained in Algorithm 6. The parameter *stepsize* has to be chosen, such that the minimization algorithm converges, invoked by the `continuation` function. That control sequence is employed in the improving algorithm as described in Algorithm 4 in Section 3.2.5. Finally, the system state x_{k+1} is determined, depending on the current state $\hat{x}_k$, the optimal state-feedback control u_k^*, and the random noise $\boldsymbol{w}_k$. After the time update, the state x_k is directly accessible and needs not to be estimated.

F.1 Application of the Minimum Principle

F.1.1 Continuation Process

In `continuation.m`, the continuation process is implemented, which deforms the linear into the nonlinear system by means of a discrete curve follower, that is, a discrete sequence of homotopy parameters γ_i. The difference between subsequent homotopy parameters is given by the parameter $stepsize = 0.01$, which was chosen in Algorithm 6 (`system_sim.m`). The employment of a continuation process is necessary, since the optimal solution to the nonlinear problem strongly depends on the initial value. If the solution of the previous continuation step serves as the initial guess of the next step, numerical instability of the subsequent minimization algorithm is reduced.

F.1.2 Minimization

The minimization function can be found in `opt_ctrl_iteration2.m`. A quasi-Newton algorithm is used at each step of the transformation of the homotopy to solve the nonlinear equation system (3.49), which depends on the homotopy parameter γ. The algorithm works as described in Algorithm 7. After the initialization in lines 2–3, it is checked, whether the initial solution is already close enough to the desired solution of the nonlinear equation system (3.49) for the current value of γ. If yes, the initial value will be returned to the calling function `continuation.m`. Otherwise, the iteration scheme starts. The Jacobian matrix is approximated by means of finite differences, [SB02], in line 10. After that, this approximation is decomposed by means of the QR decomposition in line 11, such that an update $\Delta\underline{u}$ of $\underline{U}$ can be computed. Here, the MATLAB built-in function `qr.m` of has been employed. The update of the augmented vector $\underline{U}$ is given in line 14. After the determination of the updated vector $\underline{F}$ and the quality check, either the updated solution $\underline{U}$ is returned to the invoking function or the iteration is continued. If the iteration does not converge after $MaxIt$ steps, an error message is displayed, and the last vector $\underline{U}$ is returned. This self-implemented scheme[1] performs sufficiently fast. However, if numerical problems arise, that is, Algorithm 7 does not converge, the almost identical function `mmfsolve.m` by Hanselmann [Han06] is employed, which is slower, but numerically more stable. The accuracy of both functions is similar. The difference between both functions is that

1 `opt_ctrl_iteration2.m`

Algorithm 7 Implemented quasi-Newton method

```
1:  function OPT_CTRL_ITERATION(U*(γ⁻))
2:      tolF = 10^-8                          ▷ threshold for nonlinear equation system
3:      MaxIt = 20                            ▷ stop after MaxIt iterations
4:      F=EQS(U*(γ⁻))                         ▷ build the equation system
5:      if norm(F < tolF) then                ▷ if initial value sufficiently close to solution
6:          return(U*(γ⁻))                    ▷ return initial value
7:      else
8:          U = U*(γ⁻)
9:          for i = 1 to MaxIt do             ▷ start iteration
10:             Jac=Finite_Diff(U, F)         ▷ finite difference approximation of Jacobian
11:             [Q, R]=qr(Jac)                ▷ QR decomposition of Jac
12:             y_tmp = −Q^T F
13:             Δu = R \ y_tmp                ▷ solve R^T Δu = y_tmp
14:             U = u + (Δu)^T                ▷ update of U
15:             F=EQS(U*(γ⁻))                 ▷ build the equation system
16:             if norm(F < tolF) then        ▷ if initial value sufficiently close to solution
17:                 return(U)                 ▷ return updated value
18:             end if
19:         end for
20:         print('not converged')            ▷ error message
21:         return(U)                         ▷ return last value
22:     end if
23: end function
```

`mmfsolve.m` does not employ the MATLAB built-in function `qr.m`, but a more sophisticated QR decomposition.

After the solution of the nonlinear equation system for $\gamma = 1$, the solution to the original nonlinear problem is obtained. This control sequence is returned to the function `system_sim.m`. After that, this initial guess is employed by the second part of Algorithm 6, which can be found in `update_control_main.m`.

F.2 Application of Dynamic Programming Based on Interpolation of the Value Function

F.2.1 Mean- and Covariance Prediction

Employing the open-loop control sequence of the previous section, the mean values and the corresponding covariances of the successor states of x_0 are predicted by means of the unscented transformation, which is explained in more detail in Appendix D.

Parameters of the Unscented Transformation

Distance of the Sigma Points. According to (4.11), the implementation employs a scaling parameter $\alpha = 1$, that is, the unscaled unscented transformation, to determine the sequences of means and covariances of the initial state $\hat{x}_0$ approximately. This parameter is chosen, since all moments and central moments of the transformed Gaussian distribution can be determined recursively [HS05]. Therefore, the weight ω_0 is chosen to capture the kurtosis of the Gaussian to improve the accuracy of the UT as explained in Appendix D.

Weights of the Sigma Points. According to Section 3.2.3, the parameter κ, which ensures higher accuracy of the predicted means and covariances, is chosen as

$$\kappa = \begin{cases} 3 - n & \text{for } k = 1, 2 \ , \\ 3 - n - \frac{|3-n|}{N}\, k + \frac{1}{2} & \text{for } k > 2 \ . \end{cases}$$

for the k-th prediction step of an N-step horizon. Then, κ approaches $\frac{1}{2}$, such that at time step N all sigma points are equally weighted to capture more information of the distribution in the regions, which are not close to the mean value. This choice is a contribution to the unknown shape of the unknown densities of the successor states $\boldsymbol{x}_{k+1}$, $k > 2$.

F.2.2 Definition of the Grid

The file `mean_propagation.m` performs those predictions and returns the corresponding values to `update_control_main.m`. Around the mean values, an adaptive grid is defined. Depending on the variances σ_k^x of the successor states $\boldsymbol{x}_k$ of the initial state $\hat{x}_0$, a state space discretization $\mathcal{P}_k$ of seven points is defined around the mean values $\overline{x}_k$ as

$$\mathcal{P}_k := \left\{ \overline{x}_k - \frac{3}{2}\sigma_k^x,\, \overline{x}_k - \frac{3}{4}\sigma_k^x,\, \overline{x}_k - \frac{3}{8}\sigma_k^x,\, \overline{x}_k,\, \overline{x}_k + \frac{3}{8}\sigma_k^x,\, \overline{x}_k + \frac{3}{4}\sigma_k^x,\, \overline{x}_k + \frac{3}{2}\sigma_k^x \right\} \ , \tag{F.1}$$

which realizes (3.37). The grid $\mathcal{G}_k^{init}$ is a modification of $\mathcal{P}_k$ and determined as follows. Assuming that the desired target point c lies within the area of interest after $k \geq 3$ steps[2], the nearest neighbor of the target point is replaced by c. To keep the symmetry of $\mathcal{P}_k$, the symmetric counterparts of the substituted point are also replaced by the mirrored point c. That is, the points

$$\{\overline{x}_k \pm |\overline{x}_k - c|\}$$

replace their nearest neighbors in $\mathcal{P}_k$. The recently obtained set is denoted by $\mathcal{G}_k^{init}$.

Figure F.1 shows the sequence of mean values and the corresponding area of interest, that is, the area covered by the initial grid $\mathcal{G}_k^{init}$, $k = 0, \ldots, N$, depending on the covariances of the states $\boldsymbol{x}_k$. The goal is to interpolate the value function within this area of interest.

2 heuristically chosen for system (4.1)

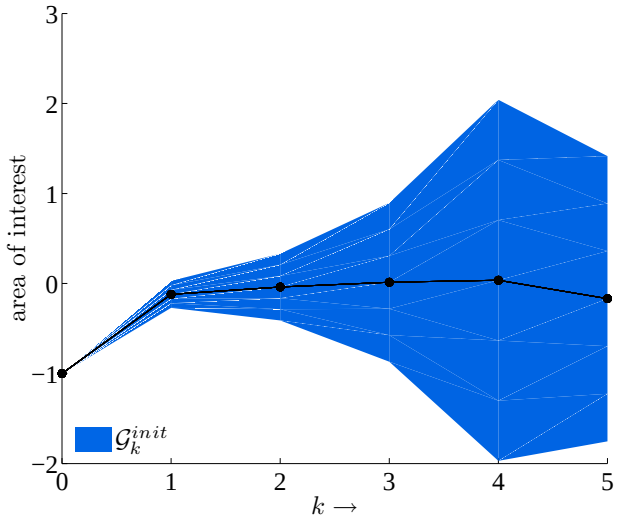


Figure F.1: Grid area around the predicted mean values of the successor states of $\hat{x}_0 = -1$ for $\sigma = 0.1$. The mean values and the corresponding covariances of the successor states $\boldsymbol{x}_k$ of $\hat{x}_0$ are calculated by means of the unscented transformation. Around the mean values, this initial grid is defined, whose spread depends on the covariance of the states.

Introduction of a Buffer to Stabilize the Spline Interpolation

To stabilize the spline interpolation on $\mathcal{G}_k^{init}$, an extended grid $\mathcal{G}_k^{ext}$ is introduced, which contains two additional dummy points at each extremal point of $\mathcal{G}_k^{init}$, that is

$$\mathcal{G}_k^{ext} = \mathcal{G}_k^{init} \cup \left\{ \min(\mathcal{G}_k^{init}) - 3\sigma_w,\ \min(\mathcal{G}_k^{init}) - \frac{3}{2}\sigma_w,\ \max(\mathcal{G}_k^{init}) + \frac{3}{2}\sigma_w,\ \max(\mathcal{G}_k^{init}) + 3\sigma_w \right\} , \tag{F.2}$$

where $\min(\mathcal{G}_k^{init})$ and $\max(\mathcal{G}_k^{init})$ denote the extremal points of $\mathcal{G}_k^{init}$ in the considered scalar case. It is noteworthy that the choice of these dummy points depends on the noise standard deviation σ_w. Altogether, $\mathcal{G}_k^{ext}$ contains 11 grid points in the scalar case. The sets $\mathcal{G}_k^{init}$ and $\mathcal{G}_k^{ext} \setminus \mathcal{G}_k^{init}$ are shown in Figure F.2.

F.2.3 Spline Interpolation of the Value Function

After the determination of $\mathcal{G}_k^{init}$ and $\mathcal{G}_k^{ext}$ for $k = 0, \ldots, N$, the value function

$$J_N(x_N) = (x_N - c)^2$$

is piecewisely interpolated by means of cubic splines in the range of $\mathcal{G}_N^{ext}$, where the MATLAB built-in function `spline.m` has been employed.

For all grid points $x_{k+1}^{(j)} \in \mathcal{G}_{k+1}^{init}$, the value $\mathrm{E}_{w_k}[J_{k+1}(\boldsymbol{x}_{k+1}^{(j)})]$ is determined by means of numerical integration in the range of the extended grid $\mathcal{G}_k^{ext}$, where the MATLAB built-in function `quad.m` has been employed.

Remark F.1 At this point, it is important to emphasize the importance of the extended grid $\mathcal{G}_k^{ext}$, which stabilizes the spline approximation of the value function J_k. If this extended grid is

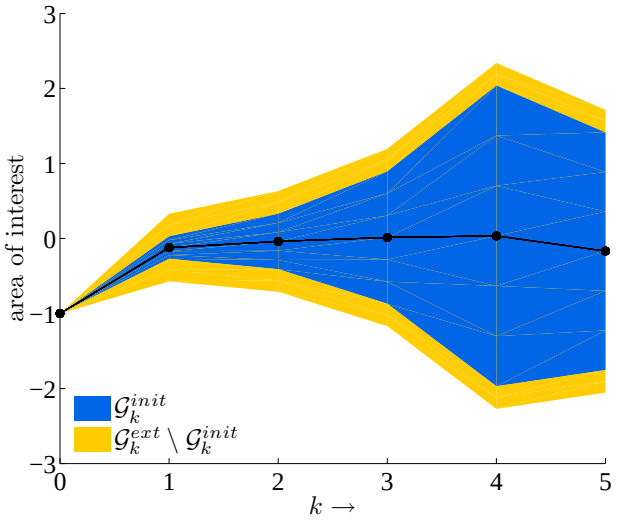


Figure F.2: Extended and initial grid area around the predicted mean values of the successor states of $\hat{x}_0 = -1$ in system (4.1) for $\sigma = 0.1$. To stabilize the spline interpolation on $\mathcal{G}_k^{init}$, a buffer of two additional points around the extremal points of $\mathcal{G}_k^{init}$ is defined, which is given by $\mathcal{G}_k^{ext} \setminus \mathcal{G}_k^{init}$.

not introduced, the calculation of $\mathrm{E}_{w_k}[J_{k+1}^{spline}(\boldsymbol{x}_{k+1}^{(j)})]$ will cause problems at the extremal points of $\mathcal{G}_k^{init}$, since the function has to be extrapolated, which is less accurate than interpolation. The additional dummy points are chosen, such that the distance of the extremal points of $\mathcal{G}_k^{ext}$ and the extremal points of $\mathcal{G}_k^{init}$ is at least $3\sigma_w$. Therefore, the integration in the range of the extended grid is almost correct, due to the fact that 99.7% of the area under the Gauss function are covered.

A set of controls $u_k^{(i,j)}$ is determined, which maps a grid point $x_k^{(i)} \in \mathcal{G}_k^{ext}$ onto the grid points $x_{k+1}^{(j)} \in \mathcal{G}_k^{init}$, where the assumption is employed that the successor state of $x_k^{(i)} \in \mathcal{G}_k^{ext}$ lies in the range of $\mathcal{G}_{k+1}^{init}$, when the optimal state-feedback control is employed. This heuristic assumption is based on simulations. With the recently determined values $u_k^{(i,j)}$, the expected cost-to-go from a state $x_k^{(i)} \in \mathcal{G}_k^{ext}$ via a successor state $x_{k+1}^{(j)} \in \mathcal{G}_{k+1}^{init}$ is obtained by

$$V_k(x_k^{(i)}, u_k^{(i,j)}) := g_k(x_k^{(i)}, u_k^{(i,j)}) + \mathop{\mathrm{E}}_{w_k}[J_{k+1}^{spline}(\boldsymbol{x}_{k+1}^{(j)})] \ .$$

This cost function is restricted to the grid points of $\mathcal{G}_k^{ext}$ and $\mathcal{G}_{k+1}^{init}$. Spline interpolation along the values $u_k^{(i,j)}, j = 1, \ldots, 7$, yields a function $V_k^{spline}(x_k^{(i)}, u_k)$, which is continuous in u_k. Subsequent minimization of this function (with respect to u_k) yields the minimal expected cost-to-go starting from $x_k^{(i)}$, where the possible successor states cover the area of interest, that is $\mathcal{G}_{k+1}^{init}$.

Remark F.2 Due to the spline interpolation with respect to the control variable u_k, the algorithm approximately treats an optimal control problem with continuous control variables.

The minimization of $V_k^{spline}(x_k^{(i)}, u_k)$ with respect to u_k is performed as explained in detail in Section 3.2.5 and yields $J_k(x_k^{(i)})$ for $x_k^{(i)} \in \mathcal{G}_k^{ext}$.

The final spline interpolation of $J_k(x_k^{(i)})$ with knots $x_k^{(i)}$ yields an approximate, continuous (in x_k) description of the value function in the range of $\mathcal{G}_k^{ext}$.

Remark F.3 Due to the spline interpolation with respect to the state variable x_k, the algorithm approximately treats an optimal control problem with continuous states.

In Algorithm 8, the proposed improvement of the initial solution is summarized.

Algorithm 8 Dynamic programming based on interpolation of the value function on an adaptive grid

1: **function** UPDATE_CONTROL_MAIN$(\hat{x}_0, \hat{u}_0, \ldots, \hat{u}_{N-1}, \sigma_w)$
2: `mean_propagation`$(\hat{x}_0, \hat{u}_0, \ldots, \hat{u}_{N-1}, \sigma_w)$ ▷ mean and covariance prediction (UT)
3: **for** $k = 0$ to N **do**
4: determine $\mathcal{G}_k^{init}$ ▷ area of interest
5: determine $\mathcal{G}_k^{ext}$ ▷ buffer around area of interest
6: **end for**
7: ▷ spline interpolation (x) of value function in the area of $\mathcal{G}_N^{ext}$
8: $J_N^{spline}(x_N)$:=`spline`$(\left\{\left(x_N^{(j)}, J_N(x_N^{(j)})\right)\right\}_{j=1}^{11})$
9: **for** $k = N-1$ to 0 **do**
10: **for** $j = 1$ to 7 **do** ▷ for all grid points in $\mathcal{G}_{k+1}^{init}$
11: determine $\underset{w_k}{\mathrm{E}}[J_{k+1}^{spline}(\boldsymbol{x}_{k+1}^{(j)})]$ ▷ expected cost-to-go from $\mathcal{G}_{k+1}^{init}$ (integration)
12: **end for**
13: **for** $i = 1$ to 11 **do** ▷ for all grid points in $\mathcal{G}_k^{ext}$
14: **for** $j = 1$ to 7 **do** ▷ for all grid points in $\mathcal{G}_{k+1}^{init}$
15: determine $u_k^{(i,j)}$ with $x_{k+1}^{(j)} = \underset{w_k}{\mathrm{E}}[f(x_k^{(i)}, u_k^{(i,j)}) + \boldsymbol{w}_k]$ ▷ $u_k^{(i,j)}$ maps $x_k^{(i)}$ onto $x_{k+1}^{(j)}$
16: $V_k(x_k^{(i)}, u_k^{(i,j)}) := g_k(x_k^{(i)}, u_k^{(i,j)}) + \underset{w_k}{\mathrm{E}}[J_{k+1}^{spline}(\boldsymbol{x}_{k+1}^{(j)})]$ ▷ cost function for $x_k^{(i)} \in \mathcal{G}_k^{ext}$
17: **end for**
18: ▷ spline interpolation (u) of cost function in the range of $\mathcal{G}_k^{ext}$
19: $V_k^{spline}(x_k^{(i)}, u_k) :=$ `spline` $(\left\{\left(u_k^{(i,j)}, V_k(x_k^{(i)}, u_k^{(i,j)})\right)\right\}_{j=1}^{7})$
20: $J_k(x_k^{(i)}) = \min\limits_{u_k} V_k^{spline}(x_k^{(i)}, u_k)$ ▷ minimization
21: $u_k^* = \arg\min\limits_{u_k} V_k^{spline}(x_k^{(i)}, u_k)$ ▷ optimal state-feedback control
22: **end for**
23: ▷ spline interpolation (x) of value function in the area of $\mathcal{G}_k^{ext}$
24: $J_k^{spline}(x_k) :=$ `spline` $(\left\{\left(x_k^{(i)}, J_k(x_k^{(i)})\right)\right\}_{i=1}^{11})$
25: **end for**
26: `return`(u_0^*) ▷ return updated control variable to `system_sim.m`
27: **end function**

F.3 Dynamic Programming on a Static Grid

The dynamic programming algorithm on a static grid is implemented as follows. After the determination of the finite sets of 250 system states and 300 control variables, a deterministic,

that is, in this case boolean, transition tensor $\mathbb{T} \in \{0,1\}^{250\times250\times300}$ is determined, where

$$\mathbb{T}(i,j,t) = 1 \quad :\Leftrightarrow \quad x^{(j)} = f(x^{(i)}, u^{(t)}) \ , \tag{F.3}$$

where $x^{(i)}$ denotes a grid point of the time-invariant discretization of the state space, and $u^{(t)}$ denotes a grid point of the time-invariant set of discretized control variables. That is, the first dimension of $\mathbb{T}$ denotes the current state, the second dimension the successor state, and the third dimension the control variable, respectively. The projection onto the x_k, x_{k+1}-plane of the tensor $\mathbb{T}$ is depicted in Figure F.3, where a fixed control is chosen. Due to the structure of

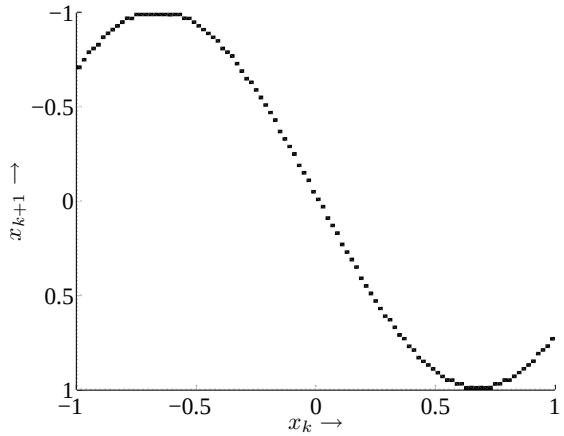


Figure F.3: Transition tensor for system (4.1) with $\sigma = 0.1$ for a fixed control variable. The boolean values reveal the unique successor states of x_k.

the corresponding system function, the boolean matrix looks like the mirrored system function. The binary values uniquely describe the successor states of x_k in the discretized region of the state space. After the determination of $\mathbb{T}$, the noise density, that is, the Gauss function, is discretized within the considered part of the state space according to

$$N(x^{(i)}) := \frac{1}{\sqrt{2\,\pi}\,\sigma} \int_{x^{(i)}-\frac{d}{2}}^{x^{(i)}+\frac{d}{2}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}\,\mathrm{d}x \tag{F.4}$$

for all grid points $x^{(i)}$. The resulting discretized Gaussian is given by a vector containing the values of (F.4), which is shown in Figure F.4. The stochastic transition tensor[3] $\mathbb{T}_w$, which incorporates the noise influence, is given by the convolution of each row of $\mathbb{T}$ with the discretized Gaussian vector, when a control variable $u^{(i)}$ is fixed. That is,

$$\mathbb{T}_w(:,:,i) = \mathbb{T}(:,:,i) * N(x) \ .$$

For a fixed control variable, the discretized transition density $\mathbb{T}_w$ is plotted in Figure F.5. Due to the noise influence, each deterministic successor state defined in (F.3) is replaced by a non-trivial density of successor states. For $\underline{x}_1 \in \mathbb{R}^n$ and $\underline{x}_2 \in \mathbb{R}^m$

$$\underline{x}_1 * \underline{x}_2 \in \mathbb{R}^{n+m-1} \ .$$

3 the discretized transition density

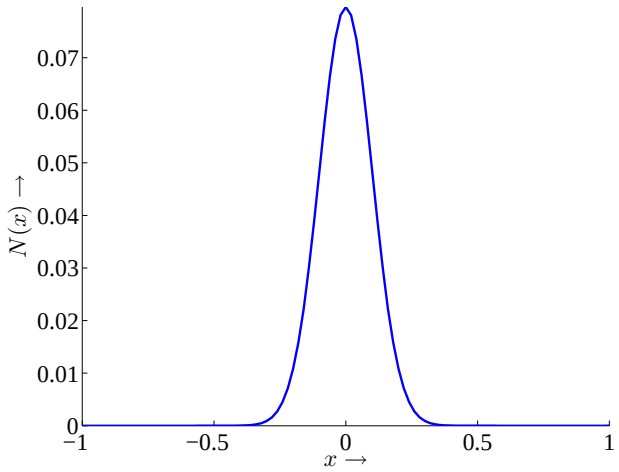


Figure F.4: Discretized Gaussian in case of system (4.1) for $\sigma = 0.1$.

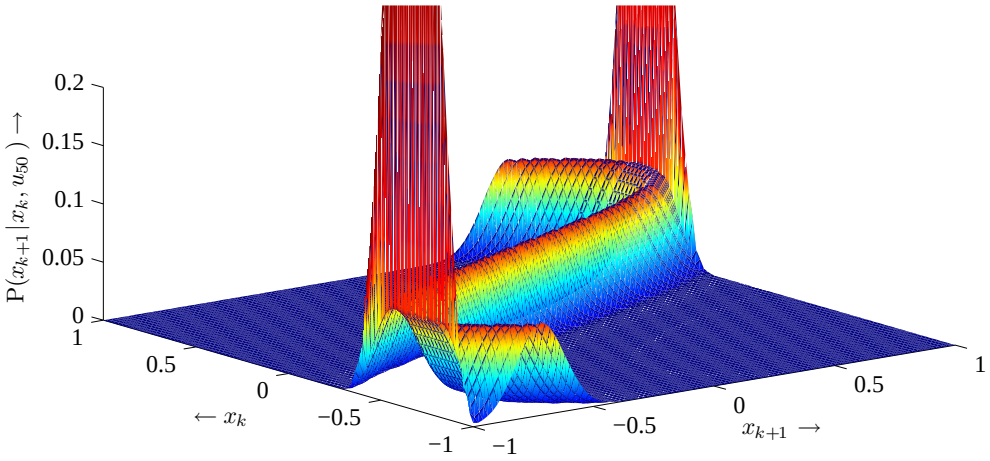


Figure F.5: Discretized transition density for a fixed control variable. Due to the influence of noise, a non-trivial distribution of the successor states of a grid point is given.

Since the resulting vector should possess the same dimensions as the original vector, the borders have to be quantized, which explains the peaks at the borders of the considered region of the state space in Figure F.5.

After these initializing steps, the expected cost-to-go at time step k for the state $x^{(i)}$ and the control variable $u^{(t)}$ is determined as

$$V_k(i,t) = \frac{1}{2}(x^{(i)} - c)^2 + \frac{1}{2}\,a(u^{(t)})^2 + \sum_{j=1}^{250} \mathbb{T}_w(i,j,t) \cdot J_{k+1}(x^{(j)}) \ , \tag{F.5}$$

where the summation weights the expected cost-to-go of the distribution of the successor states. Moreover, it is assumed that J_{k+1} has already been determined by the DP algorithm, that is, Algorithm 1. The value function for $x^{(i)}$ is given by the minimum

$$J_k(x^{(i)}) = \min_t V_k(i,t)$$

of (F.5). The value function within the range of the restricted region of the state space for all time steps is shown in Figure F.6. With increasing prediction horizon, the value function becomes more complex, starting from a quadratic function at time step 5, where the incurring cost is just the terminal cost.

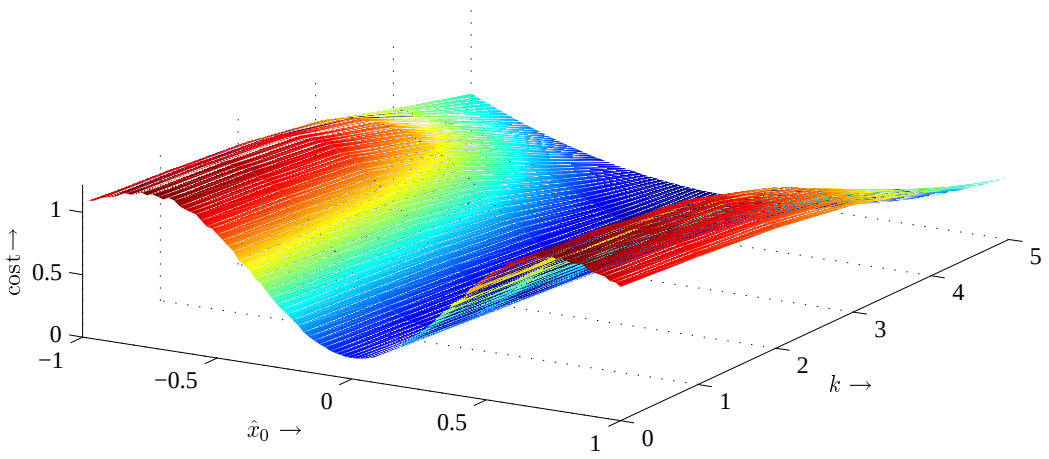


Figure F.6: Minimal expected cost-to-go for all time steps of the decision-making horizon. For system (4.1) with $\sigma = 0.1$, the horizon is set to $N = 5$ steps. The minimal expected cost-to-go for $\hat{x}_0$ is given for $k = 0$.

The minimizing control variable $u^{(t)}$ is stored in a look-up table $\mathbf{U}$. After performing this minimization for all states, this look-up table contains the optimal control state-feedback controls for all states and all time steps. This calculation can be performed offline. During the simulations, only the entries of the look-up table $\mathbf{U}$ are needed. Since MPC is implemented, only the entries for the full horizon length are accessed during the simulations.

Bibliography

[Aub00] Jean-Pierre Aubin. *Applied Functional Analysis*. Pure and Applied Mathematics. John Wiley & Sons, Inc., Scientific, Technical, and Medical Division, 605 Third Avenue, New York, N.Y. 10158-0012, 2nd edition, 2000.

[ÅW97] Karl J. Åström and Björn Wittenmark. *Computer-Controlled Systems — Theory and Design*. Prentice Hall, Upper Saddle, New Jersey, 3rd edition, 1997.

[Ber87] Dimitri P. Bertsekas. *Dynamic Programming — Deterministic and Stochastic Models*. Prentice Hall, Inc., Englewood Cliffs, N.J. 07632, 1987.

[Ber00a] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1 of *Optimization and Computation Series*. Athena Scientific, Belmont, Massachusetts, U.S.A., 2nd edition, 2000.

[Ber00b] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2 of *Optimization and Computation Series*. Athena Scientific, Belmont, Massachusetts, U.S.A., 2nd edition, 2000.

[BM05] Seid Bahlali and Brahim Mezerdi. A General Stochastic Maximum Principle for Singular Control Problems. *Electronic Journal of Probability*, 10(30):988–1004, May 2005.

[Bos06] Siegfried Bosch. *Algebra*. Springer-Verlag, 6th edition, 2006.

[Bra72] Franklin H. Branin, Jr. Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations. *IBM Journal of Research and Development*, 16(5):504–522, September 1972.

[BS96] Dimitri P. Bertsekas and Steven E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Optimization and Neural Computation Series. Athena Scientific, Belmont, Massachusetts, U.S.A., 1996.

[BSMM01] Ilja N. Bronstein, Konstantin A. Semendjaev, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 5th edition, 2001.

[CH94] Abel Cadenillas and Ulrich G. Haussmann. The stochastic maximum principle for a singular control problem. *Stochastics and Stochastic Reports*, 49:211–237, 1994.

[CHL05] Ningzhou Cui, Lang Hong, and Jeffery R. Layne. A comparison of nonlinear filtering approaches with an application to ground target tracking. *Signal Processing*, 85:1469–1492, August 2005.

[dB78] Carl de Boor. *A Practical Guide to Splines*, volume 27 of *Applied Mathematical Sciences*. Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, N.Y. 10010, 1978.

[Dei04] Marc P. Deisenroth. Toward Optimal Control of Nonlinear Systems with Continuous State Spaces. Studienarbeit, Intelligent Sensor-Actuator-Systems Laboratory, Faculty of Informatics, Universität Karlsruhe (TH), November 2004.

[DOW+06] Marc P. Deisenroth, Toshiyuki Ohtsuka, Florian Weissel, Dietrich Brunn, and Uwe D. Hanebeck. Finite-Horizon Optimal State Feedback Control of Nonlinear Stochastic Systems Based on a Minimum Principle. To appear in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2006)*, pages 371–376, Heidelberg, Germany, September 3–6, 2006.

[Fai98] Frederick W. Fairman. *Linear Control Theory: The State Space Approach*. John Wiley & Sons, Ltd., Baffins Lane, Chichester, West Sussex PO19 1UF, England, 1998.

[Fin04] Rolf Findeisen. *Nonlinear Model Predictive Control: A Sampled-Data Feedback Perspective*. PhD thesis, Fakultät Maschinenbau, Universität Stuttgart, December 2004.

[FM97] Delbert D. Franz and Charles S. Melching. Full Equations Utilities (FEQUTL) Model for the Approximation of Hydraulic Characteristics of Open Channels and Control Structures During Unsteady Flow. Water Resources Investigations Report 97-4037, U.S. Geological Survey, 1997.

[Hal73] C. A. Hall. Natural Cubic and Bicubic Spline Interpolation. *SIAM Journal on Numerical Analysis*, 10(6):1055–1060, December 1973.

[Han06] Duane Hanselmann. mmfsolve. http://www.mathworks.com/matlabcentral/fileexchange/, May 2006. Software.

[HBH06] Marco Huber, Dietrich Brunn, and Uwe D. Hanebeck. Closed-Form Prediction of Nonlinear Dynamic Systems by Means of Gaussian Mixture Approximation of the Transition Density. To appear in *IEEE International Conference on*

Multisensor Fusion and Integration for Intelligent Systems (MFI 2006), pages 98–103, Heidelberg, Germany, September 3–6, 2006.

[HBR03] Uwe D. Hanebeck, Kai Briechle, and Andreas Rauh. Progressive Bayes: A New Framework for Nonlinear State Estimation. In *Proceedings of SPIE*, volume 5099, pages 256–267, Orlando, 2003. AeroSense Symposium.

[Hol70] Jack M. Holtzman. *Nonlinear System Theory — A Functional Analysis Approach*. Prentice Hall Network Series. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1970.

[HS05] Uwe D. Hanebeck and Oliver C. Schrempf. Stochastische Informationsverarbeitung. Textbook for Lecture, Chair for Intelligent Sensor-Actuator-Systems, Universität Karlsruhe (TH), 2005.

[JK01] C. Russell Philbrick Jr. and Peter K. Kitanidis. Improved Dynamic Programming Methods for Optimal Control of Lumped-Parameter Stochastic Systems. *Operations Research*, 49(3):398–412, May–June 2001.

[JS94] Kenneth L. Judd and Andrew Solnick. Numerical Dynamic Programming with Shape-Preserving Splines. preliminary and incomplete, 1994.

[JSS+93] Sharon A. Johnson, Jery R. Stedinger, Christine A. Shoemaker, Ying Li, and Jose Alberto Tejada-Guibert. Numerical Solution of Continuous-State Dynamic Programs Using Linear and Spline Interpolation. *Operations Research*, 41(3):484–500, May–June 1993.

[JU96] Simon Julier and Jeffrey K. Uhlmann. A General Method for Approximating Nonlinear Transformations of Probability Distributions. Technical report, Robotics Research Group, Department of Engineering Science, University of Oxford, Oxford, OC1 3PJ United Kingdom, November 1, 1996.

[JU97] Simon J. Julier and Jeffrey K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *Proceedings of AeroSense: 11th Symposium on Aerospace/Defense Sensing, Simulation and Controls*, pages 182–193, Orlando, Florida, 1997.

[JU02] Simon J. Julier and Jeffrey K. Uhlmann. Reduced Sigma Point Filters for the Propagation of Means and Covariances Through Nonlinear Transformations. In *Proceedings of the American Control Conference*, volume 2, pages 887–892, Anchorage, Alaska, U.S.A., May 8–10, 2002.

[JU04] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. In *Proceedings of the IEEE*, volume 92, March 2004.

[Jud98] Kenneth L. Judd. *Numerical Methods in Economics*. The MIT Press, Cambridge, Massachussetts, 1998.

[JUDW00] Simon Julier, Jeffrey Uhlmann, and Hugh F. Durrant-Whyte. A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, March 2000.

[Jul02] Simon J. Julier. The Scaled Unscented Transformation. In *Proceedings of the American Control Conference (ACC)*, volume 6, pages 4555–4559, Anchorage, Alaska, U.S.A., May 8–10, 2002.

[Ker71] D. Kershaw. A Note on Convergence of Interpolatory Cubic Splines. *SIAM Journal on Numerical Analysis*, 8(1):67–74, March 1971.

[LBS02] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. Comment on "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators" [and author's reply]. *IEEE Transactions on Automatic Control*, 47(8):1406–1409, August 2002.

[Lue69] David G. Luenberger. *Optimization by Vector Space Methods*. Series in Decision and Control. John Wiley & Sons, Inc., 1969.

[LW02] S. J. Luo and John B. Walsh. A Stochastic Two-Point Boundary Value Problem. *Electronic Journal of Probability*, 7(12):1–32, 2002.

[Mat05] The MathWorks, Inc. MATLAB 7.1. http://www.mathworks.com/, August 2005. Software.

[NB03] Daniel Nikovski and Matthew Brand. Non-Linear Stochastic Control in Continuous State Spaces by Exact Integration in Bellman's Equations. In *13th International Conference on Automatic Planning & Scheduling (ICAPS '03)*, pages 91–95, Trento, Italy, 2003.

[OF94] Toshiyuki Ohtsuka and Hironori Fujii. Stabilized Continuation Method for Solving Optimal Control Problems. *Journal on Guidance, Control, and Dynamics*, 17:950–957, November 1994.

[Oht00] Toshiyuki Ohtsuka. Continuation/GMRES Method for Fast Algorithm of Nonlinear Receding Horizon Control. In *Proceedings of 39th IEEE Conference on Decision and Control*, pages 766–771, Sydney, Australia, December 2000.

[Pen90] Shige Peng. A General Stochastic Maximum Principle for Optimal Control Problems. *SIAM Journal Control and Optimization*, 28(4):966–979, July 1990.

[PTVF02] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flanning. *Numerical Recipes in C — The Art of Scientific Computing*. The Press Syndicate of the University of Cambridge, The Pitt Building, Trumpington Street, Cambridge CB2 1RP, United Kingdom, 2nd edition, 2002.

[Put05] Martin L. Puterman. *Markov Decision Processes — Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, U.S.A., 2005.

[RD83] Stephen L. Richter and Raymond A. DeCarlo. Continuation Methods: Theory and Applications. In *IEEE Transactions on Automatic Control*, volume AC-28, pages 660–665, June 1983.

[Reh05] Hans Peter Rehm. Topologie. Lecture Notes, Universität Karlsruhe (TH), 2005.

[RRD04] Vicente Rico-Ramirez and Urmila M. Diwekar. Stochastic maximum principle for optimal control under uncertainty. *Computers & Chemical Engineering*, 28(12):2845–2849, November 2004.

[SB02] Josef Stoer and Roland Bulirsch. *Introduction to Numerical Analysis*. Number 12 in Texts in Applied Mathematics. Springer-Verlag, Inc., 175 Fifth Avenue, New York, NY 10010, U.S.A., 3rd edition, 2002.

[Sch83] Larry L. Schumaker. On Shape-Preserving Quadratic Spline Interpolation. *SIAM Journal on Numerical Analysis*, 20(4):854–864, August 1983.

[SOD06] Yuichi Shimizu, Toshiyuki Ohtsuka, and Moritz Diehl. A Real-Time Algorithm for Nonlinear Receding Horizon Control Using Multiple Shooting and Continuation-Krylov Method. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2006)*, pages 766–770, Kyoto, Japan, July 24–28, 2006.

[SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, Cambridge, MA, 2002.

[TJ79] James D. Turner and John L. Junkins. Optimal Large-Angle Single-Axis Rotational Maneuvers of Flexible Spacecraft. In *2nd AIAA/VPI&SU Symposium on Dynamics and Control of Large Flexible Spacecraft*, pages 91–110, Blacksburg, VA, June 21–23, 1979.

[vdMDdFW00] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric Wan. The Unscented Particle Filter. Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, August 16, 2000.

[vdMW04] Rudolph van der Merwe and Eric A. Wan. Sigma-Point Kalman Filters for Integrated Navigation. In *Proceedings of the 60th Annual Meeting of The Institute of Navigation (ION)*, Dayton, OH, June 2004.

[Vid02] Mathukumalli Vidyasagar. *Nonlinear Systems Analysis*. SIAM's Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2nd edition, 2002.

[XZC06] K. Xiong, H. Y. Zhang, and C. W. Chan. Performance evaluation of UKF-based nonlinear filtering. *Automatica*, 42:261–270, 2006.