Damian Kontny

Fast Online Control based on Homotopies for Systems subject to Time-Varying Constraints

kassel university press

Damian Kontny

Fast Online Control based on Homotopies for Systems subject to Time-Varying Constraints

kassel university press

This work has been accepted by Faculty of Electrical Engineering / Computer Science of the University of Kassel as a thesis for acquiring the academic degree of Doktor der Ingenieurwissenschaften (Dr.-Ing.).

Supervisor: Prof. Dr.-Ing Olaf Stursberg
Co-Supervisor: Prof. Dr.-Ing Knut Graichen
Defense day: 19th February 2020

Bibliographic information published by Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at http://dnb.dnb.de.

Zugl.: Kassel, Univ., Diss. 2020
ISBN 978-3-7376-0870-1
DOI: https://dx.doi.org/doi:10.17170/kobra-202008131567

http://kup.uni-kassel.de

Printed in Germany

Contents

Summary

The integration of intelligent, autonomously acting systems into modern society is a rapidly growing field. After robots are established for simple, recurring processes in industry and everyday life, more complex tasks are of interest which require the systems to consider the environment in their decision making process. In the future, intelligent systems will get access to fields like autonomous driving cars, unmanned aerial vehicle (UAV), manufacturing processes, household, or the assistance to people in need of care. The fast calculation of optimal circumventing trajectories is therefore an essential component to be able to integrate intelligent systems into our environment at all.

The ambitious goal of real-time interaction between a human and an autonomous system is challenging. An autonomous system has to react timely on human motion such that a real interaction can be established. Thus, the autonomous system must continuously capture its environment and adapt its solution. If the system additionally determines a solution with respect to a certain optimization criterion, the computation times quickly rise, and real-time capability moves far away.

To solve this problem, the thesis proposes an algorithmic control procedure which determines optimal collision-free trajectories fast. Therefore, a concept which uses homotopy properties in the control procedure is introduced. This allows to determine near-optimal solutions much faster than by commonly used techniques.

At the beginning, an algorithmic procedure is shown for linear systems. It selects an optimized, circumventing trajectory based on the current obstacle location, and adapts its trajectory when the obstacle moves. Since real physical systems always underlie actuator limitations like e.g. motor torques, the provided method also considers input constraints. The developed method is subsequently extended to consider predictions of a moving obstacle. Thus, the procedure can further reduce the costs of the executed trajectory and is able to detect and avoid a collision at an early stage.

Since many real-world system are described by nonlinear dynamics, the introduced method is also extended to nonlinear systems. The effectiveness of the proposed approach is shown in several simulations. Motivated by the results, the developed approach is extended to more complex tasks, like the collision avoidance between geometric bodies, and to multi agent systems that cooperatively determine solution trajectories in real-time, by means of the homotopy properties. Simulation results of a collision avoidance scenario for a robotic manipulator show, that with the proposed technique good results can be obtained in real-time.

Acknowledgments

First of all, I thank my supervisor Prof. Dr.-Ing. Olaf Stursberg for the continuous support of my research, for his motivation and ideas. I am grateful for the freedom that he gave me and his great guidance.

I also would like to thank the complete team of the Institute of Control and System Theory, where the research for this thesis was conducted. The fruitful discussions regarding all questions in control were always helpful. Furthermore, I am grateful for the financial support by the EU through the H2020-project UnCoVerCPS.

Finally, I would like to thank my family and friends for the encouragement and enduring support during the past years.

Part I.

Introduction and Theoretical Background

1. Introduction

In this thesis, control algorithms for non-convex point-to-point optimization problems of dynamical systems are developed. Point-to-point optimization problems appear in many situations, and thus represent a basic problem in control theory. The task of such an optimization problem is to bring the dynamical system from an initial state to a target state optimally regarding to a given performance function. A practical example is the movement of a robotic manipulator as typically used in industrial processes. The task of the robot may be to grasp a workpiece and bring it to a new point, where, e.g. further processing steps take place. The dynamic behavior of the system is usually described by mathematical equations, or in particular, by ordinary differential equations (ODE). These equations describe the change of the system states under some external inputs, e.g. the power of the actuator drives of the robot manipulator. Such ODEs can be embedded into the optimization process for trajectory planning. Objectives for the optimization can be different criteria like, the minimization of the euclidean distance from the initial to the target state, the time needed for the transition, or the energy consumption of the actuators. A combination of different criteria is also possible. Such an optimization allows the dynamical system to contribute to maximize the process flow and finally the profit. The same applies, of course, to other practical examples like unmanned areal vehicles (UAVs) transporting packages in urban environment, or self-driving cars that carry human passengers to their destinations.

If such systems are to act independently and intelligently, they must pay attention to their environment and adapt to it. For manufacturing processes there is a great interest in human robot cooperation: One goal is, that human workers and robotic manipulators co-exist in a common workspace without spatial separation, in order to combine the advantages of human flexibility and robotic precision. From the robotic point of view, a human worker can be seen as a moving obstacle which may block the robot in its motion towards the target point. In this case, one solution may be to stop the system and wait until a safe continuation of the trajectory is possible. Obviously, the disadvantage of such an approach is, that the process is forced to stop for a certain period of time. The consequences are a drastic increase of the costs. A more efficient solution is of course, to determine a new trajectory which is free of collisions with respect to the current and future obstacle positions.

From the mathematical perspective, trajectory planning with moving obstacles can be formulated as an optimization problem in a time-varying, non-convex search space, where non-convexity of the search space can arise from the partial occupancy of the space due to the obstacle. A frequently used method for solving such collision

avoidance problems is Model Predictive Control (MPC). This method determines future control actions by online numerical optimization. The advantage of this method is that constraints like the collision avoidance constraints, actuator limitations, and the system dynamics, can be considered during the optimization. The method aims at minimizing a predefined cost function. The idea behind MPC is to use the system and obstacle dynamics for the prediction of the system trajectory under control. The first input is then applied to the system, and the procedure is repeated. While a solution of the considered class of optimization problems is possible for applications with slow dynamics, the situation is more difficult for fast dynamic systems. There, a solution has to be determined more quickly. Consider the human-robot example: While the human can move his extremities very fast, the robot also relies on techniques that determine circumventing trajectories fast.

Today, the solution of such non-convex optimization problems is time consuming and prevents MPC from being used in many time-critical applications. This is due to several reasons. The non-convexity either results in a mixed integer optimization problem (MIP), using discrete and continuous variables, or in a nonlinear formulation of the problem (NLP). Another challenge is the large number of optimization variables which quickly arises when using an MPC formulation. When using MPC, a time discretization of the inputs and the states is usually made. Thus, the number of the variables and constraints strongly depends on the prediction horizon. This leads to the fact that often only small horizons can be used, or the discretization time has to be increased. However such compromises have further disadvantages. A simple reduction of the prediction horizon causes the system to run into a local minimum, which means that the system can get stuck in front of the obstacle. Large discretization times on the other hand, can result in solution trajectories that are free of collisions at the discretization times, but can be in collision for times between this time discretization. Thus, the problem of determining quickly optimized trajectories in the presence of obstacles is still a challenging problem which requires further research efforts.

The challenge in this thesis is thus to maintain purposeful prediction horizons, while reducing the computation time and still obtaining close to optimal solutions. Instead of optimizing over every single point in time, as it is the case in MPC, the idea of this thesis is to optimize over complete trajectories. From an interpolation between predetermined trajectories, an algorithmic procedure efficiently selects an optimized and collision-free trajectory.

Predetermined trajectories are consequently only valid as long as the target state does not change. However, real world problems are faced with the challenge not only to reach a fixed target state, but to track a moving target state. Consider a robot that has to pick up an workpiece which is randomly placed on a conveyor belt, while moving in a shared workspace with a human worker. The thesis provides an online procedure which quickly adapts to the moving target and updates its optimized solution trajectory to obtain a collision-free trajectory.

Besides the problem of a moving target, the importance of an efficient online

process becomes particularly clear when it comes to nonlinear system dynamics. In many cases, the physical description of the system behavior leads to a set of nonlinear equations. In the robotic manipulator example, the individual links are positioned according to geometric conditions. Often such dynamics is globally approximated by linear systems. However, the resulting linearization errors do not permit precise statements about the feasibility and quality of the solution. Thus, the consideration of nonlinear system classes is often unavoidable and therefore the approach in this thesis is extended to nonlinear dynamics.

In the determination of collision-free trajectories it is often the case, that the obstacle is regarded as a geometric body, but the dynamical system itself is regarded as a point mass in space. This approximation is usually done to simplify the problem for the optimization step. The thesis shows how the developed control procedure is efficiently extended to the case of dealing with the geometric body of the obstacle, and the geometry of the dynamical system, simultaneously.

Finally, this thesis extends the results to multi agent systems which underlie a certain degree connection, and are able to communicate between each other. Such systems can already be found in industrial assembling processes where several robots work together to accomplish tasks. Nowadays, such systems strictly execute predefined tasks. Environmental changes like the presence of a human moving in this area or unexpected motions from a single subsystem can not be efficiently considered by all subsystems in real-time. However, future systems will plan their movements itself to cooperatively work on a common task and to consider environmental changes. Thus, the determination of optimized trajectories in real time becomes more and more important for interconnected multi agent systems. Considering that communication networks between these systems can be established today with low costs, the development of efficient control procedures for distributed systems is a current field of interest to which the developed approach in this thesis also contributes.

In summary, this thesis introduces methods for the determination of optimized, collision-free trajectories in real-time, by using homotopy properties. The developed method can be applied to different variants of the problem, and to different system classes. Thus, a holistic new approach is presented.

Outline of the Dissertation

This thesis provides a literature review of approaches for path and trajectory planning Chapter 2, followed by a description of the contribution of this work.

Chapter 3 introduces necessary mathematical notation. This includes a brief introduction into matrix calculations and transformations, as well as the principles of model predictive control, serving as a starting point for the developments.

Chapter 4 Introduces the principle on which the developed method is based on, namely the property of homotopy. Then, this chapter focuses on a point-to-point optimization problem for linear discrete-time dynamics in a non-convex search space.

Based on homotopic trajectories an approach is introduced which transfers the problem from a standard state-space optimization problem into a homotopy space optimization problem.

Chapter 5 enhances the developed procedure from Chapter 4 to consider input constraints already in the controller synthesis. It is also shown that the number of predetermined trajectories compared to the dimension of the system dynamics influences the controller synthesis. Essentially, three relevant situations that influence the controller synthesis can occur: Having less, more, or the same number of predetermined trajectories compared to the system dimension. The importance of these cases is discussed.

Compared to the previous chapters, Chapter 6 considers a special characteristics of the problem in which the dynamical system can no longer be controlled in one step. The main result is an control scheme which additionally considers predicted information of the obstacle movement. Finally an algorithmic procedure, named the *Homotopic Control Algorithm*, `HCA`, is presented. The algorithm determines a collision-free trajectory based on a tree-search procedure.

In Chapter 7, nonlinear system dynamics and a moving target state are considered. This chapter uses online linearization and a heuristic approach for online generation of predetermined trajectories. The developed procedure is applied to a robotic manipulator dynamics with two revolute joints, operating in the planar space.

Chapter 8 connects to the example of controlling a nonlinear robot manipulator by further considering the complete body geometry for the obstacle avoidance process such, that e.g. a collision between a link of the robot and the obstacle can also be excluded. To achieve this, the *Homotopic Control Algorithm* is extended by an particle approach.

Chapter 9 considers the case in which multiple robotic manipulators operate in a shared workspace. In the context of this problem, a distributed, cooperative control approach is developed. Each robot plans its trajectory with its local `HCA`, while considering the predicted motions of the other agents via a communication network.

Chapter 10 concludes this thesis and proposes some future research directions in the topic of homotopic control.

2. Literature Review

The aim of this chapter is to provide an overview of relevant literature in the field of motion planning for obstacle avoidance. As already mentioned, this thesis focuses on the control of collision avoidance problems as they arise for instance in the field of human robot interaction. Hence, the task is to find a trajectory that moves the system from the start to the goal configuration, while not intersecting with any obstacle.

Generally, when planning a collision-free motion between e.g., a robotic manipulator and an obstacle, the check against collisions is done in a space where both objects are described simultaneously. This can be done in the workspace or the configuration space (C-space). The workspace is defined as the euclidean space where the obstacle is located. The advantage of planning in the workspace is, that the collision avoidance constraints are directly available due to the obstacle geometry. On the other side, the dynamic system must be described in the workspace. However, the position of a robotic manipulator is described by the angles of each link. These angles are referred to as the configurations. Planning in the C-space means, that the system remains in its original space, but the obstacle must be transferred into it. Since most of the planning methods work in the C-space, the construction and obstacle transformation into it is first discussed in this chapter. After that, relevant planning methods are introduced.

Basically one can distinguish between a trajectory and a path. A trajectory is a sequence of movements that can be followed by the dynamical system. On the other hand, a path is a geometrically defined route, which may not be executable by the dynamics. If for example, a path sharply bends, the dynamical system is only able to follow this bend in a wider curve. Within the framework of computing trajectories, one can separate the problem into a path planning problem, followed by specifying how this path should be passed in terms of the time behavior or a trajectory planning problem. In the path planning, the time behavior can be done by either defining velocity profiles on this path or by delegating this task to a lower level controller. One problem of such a split approach is the question, whether the obtained path is dynamically feasible, and to still guarantee a collision free trajectory. On the other hand, the solution of the problem can also be determined directly as a time parametrized function, hence a trajectory. However, such direct formulations often suffer from being computationally expensive.

For a good overview, the reader is referred to the works [66] and [69]. With respect to self driving vehicles, a survey of the relevant methods is given in [92] and [35]. From the perspective of robot path planning, a general overview is given in

[33] and [20]. The following section contains a detailed literature review on existing approaches.

2.1. Configuration Space

A planning problem can mean to drive a robotic manipulator from its initial position to a target position, while avoiding collision with an obstacle that may block the desired path in the workspace. This planning problem can also be defined in another space, the C-space. Under presence of obstacles, the C-space can be separated into a free space and a space occupied by the obstacle. If one thinks of an obstacle, e.g. with the geometry of a ball and a robotic manipulator, collisions between the ball and the robot can not only occur between the robot end effector and the ball, but also between the robot links and the ball. The mapping of the obstacle in to the C-space thus provides the complete information on free and occupied configurations. A free configuration thus positions the robot in the workspace such that no collision with the obstacle occurs. Thus, the representation of the obstacle avoidance problem into the C-space is a key concept for many planning algorithms. As already introduced in the survey paper [124], C-space maps can be categorized in three main subgroups: boundary representation methods, division and classification methods, and hybrid methods.

A boundary representation method is demonstrated in [78], which uses the Minkowski sum between non-rotating planar robots, and obstacles, to obtain a grown configuration space of forbidden regions. The additional consideration of obstacle rotations is considered in [15]. The work of [81] analytically determines the boundaries of obstacles in the C-space for robotic manipulators. The method determines when disjoint workspace obstacles intersect in the C-space and uses this topological information for the identification of free spaces. The boundary representation of C-space obstacles for manipulators is further handled in [47]. There, the boundary equations are derived from intersections between a collection of triangular patches (representing the obstacles) and line segments. Nevertheless, such geometry-based methods are usually limited to low-dimensional C-space because of the analytical complexity.

Division and classification methods discretize the C-space into a number of cells, for which each cell is classified as either safe, prohibited (fully contained in the obstacle), or in contact (partially contained in the obstacle). A realization of such mappings is shown in [1], by storing the cell information in lookup-up-tables. The amount of memory is reduced by using symmetry properties of the robot in the workspace. The work of [126] stores C-space maps, which are indexed by cells of the workspace. Thus a complete mapping is made offline even for changes like obstacle or robot movements. Instead of rastering the C-space for identification of the collision free space, [120] presents the dual problem. That is, only C-space parts are mapped which are guaranteed to be free of collision due to geometric

criteria, while some configurations that are free of collision are not captured. The determination of the C-space as a convolution between the robot and the obstacle by a fast Fourier transform (FFT), is shown in [54], and [105]. In [7], the occupied C-space is determined by the Minkowski sum, which is based on the idea of reduced convolution.

In [14], an approach is shown, which identifies simple elements in the Cartesian space that can be rapidly transformed into the C-space. Complex shapes are then described by multiple transformations of such simpler primitives. An efficient transformation of these primitives is realized by a hybrid mapping algorithm. This means a combination of analytic considerations and C-space discretization.

The main problem of C-space mapping are the non-convex shapes that result when representing the obstacle in the C-space. In addition, movements of the obstacle in the workspace lead to unpredictable changes of its shape in the C-space. Thus, approximations of the geometries or restrictions in the allowable locations of the obstacle and the dynamic system are made. This often leads to the fact that only simple and low dimensional problems can be considered. Nevertheless, C-space maps are essential for many planning algorithms and are often used in combination with other approaches.

2.2. Planning Methods

In the past decades, a variety of methods were developed that determine feasible paths or trajectories in different ways. These methods can be divided into the following categories:

- Roadmaps
- Artificial Potential Field Method
- Sampling-Based Methods
- Optimization Based Methods
- Homotopy Methods in Optimization

The following section introduces these methods.

2.2.1. Roadmaps

The term roadmap describes the discretization of the free configuration space into a finite set of nodes and edges. A node represents a configuration of the system, while an edge describes the transition between two nodes. Nodes can only be connected, if the path between them is free. Thus, the task is to first build a finite graph, i.e. a roadmap of free configurations, followed by the search of a free path in this

constructed graph. Well-known roadmaps are visibility graphs, Voronoi diagrams, and cell decomposition methods. These methods have in common that they rely on a geometric representation of the obstacle in the configuration space. Commonly, the obstacles are given as polygons or polyhedra.

In visibility graphs, see [79], nodes correspond to vertices of the obstacles. If the connecting line segment between any two nodes is in the free space, or corresponds to an edge of the obstacle, the nodes are connected. These types of roadmaps are commonly used to find the shortest Euclidean path between a start, and a goal configuration. In [46], the concept of vertex circles (V-circle) is introduced to decrease the computation time for reconstructing the graph, when the obstacle changes. The algorithm identifies the nodes of the convex hull of the obstacle, and reduces the construction of the visibility graph to this nodes. The path is then constructed between tangent points of the V-circles and if the obstacle size changes, the only re-computed part is the tangent line between the V-circles.

A Voronoi diagram, see [4], is a partition of the space into sub-regions, which are based on distributed points in the space. Each point represents the center of a Voronoi cell. A Voronoi cell consists of all points that are closer to the corresponding center, than to any other. Generalized Voronoi diagrams (GVD) are used for space partitioning in presence of geometric obstacles. Since exact computations of GVDs are limited to small special cases, the main methods are based on approximation techniques which raster the configuration space [41, 112]. A path planning algorithm for a mobile robot navigating in a cluttered environment is shown in [32]. The author combines a Voronoi diagram method with a fast marching method, to determine shortest paths that are close to the obstacles.

Roadmaps are also constructed by cell decomposition methods. These can be divided into exact and approximate methods [66]. The exact cell decomposition methods decompose the free space into trapezoidal or triangular cells. Each cell is then numbered and represents a node in the search tree, where neighboring nodes are connected by an edge. The resulting path is a connection of cells between the initial to the goal configuration. A often used method is to pass adjacent cells through the midpoint of the intersection plane. In contrast, approximate cell decomposition method recursively divide the configuration space into smaller cells. This process is done until either a resolution limit is reached, or the cell can be clearly classified as a free cell or a cell that is occupied by the obstacle. In [101], these methods are used for path planning of an unnamed areal vehicle (UAV).

A disadvantage of roadmap construction is that the complexity of the algorithms quickly increases when the dimension of the C-space grows or the number of obstacle faces becomes large. Unfortunately, the latter is often the case since the obstacles can take complex shapes in the C-space after transformation from the workspace. Furthermore it is clear that these path planning approaches do not guarantee for a feasible execution of the path by the dynamic system, since the dynamics is not considered at all: A path is feasible per construction, a trajectory not. Therefore these methods often work hierarchically. The upper level plans the path by one

of the named methods and a lower level, consisting of local controllers, realizes a dynamically feasible motion.

Graph Search Strategies for Roadmaps In the previous section, roadmap approaches were introduced that represent the free C-space in form of a graph. The approaches therefore have in common to search the graph for an appropriate path with a graph search algorithm. A well-known graph search algorithm is the Dijkstra algorithm [118], which determines the shortest path from a starting node to any other node. In manufacturing processes the task often occurs that a robot has to move from an initial position to a final position. Thus, a complete exploration, as it is done with the Dijkstra algorithm, becomes inefficient. Since the final position is known, a guided search of the graph is useful. For such situations, a heuristically guided algorithm like A* [37] achieves better performance compared to an unguided search by avoiding the evaluation of the complete graph. Since the A* algorithm can be updated at every iteration, changes in the environment, e.g. a movement of the obstacle, can be considered. However, during such updates, usually only small parts of the graph change. This would make a complete replanning inefficient. Therefore, D* [108] and D*-Lite [59, 113] were developed, which use path information from the previous iterations during the replanning process. Nevertheless, in real world problems, A* and D* based algorithms, may not finish their computations in time. Therefore, anytime repairing algorithms like the ARA* [74] were developed. They determine an initial solution quickly, and then continuously improve the solution by changing a weight factor of the heuristic, and by using information from the previous iteration. The work of [73] unifies the ARA* with the D* algorithm to refine a path in an anytime fashion and additionally update the graph when changes are observed. Although graphs can be efficiently explored, the main problem still remains the construction of the graph by the C-space mapping and the roadmap generation. Another disadvantage of using such methods is the fact, that predictions of the obstacle movement can not be considered efficiently. Predictions would extend the graph by further dimensions. This would highly increase the graph, leading to a large amount of storage and exploration time. Typically, roadmaps with graph search strategies react to the current situation and do not consider predictions at all.

2.2.2. Planning based on Artificial Potential Fields

The approach developed in [57] generates an attractive potential field (APF) around the goal state such, that the robot configuration, which is considered as a point in the C-space, moves towards the goal state. A repulsive fields around the obstacles repels the movement to prevent collisions. The resulting force, respectively the gradient of the resulting superposition between the attractive and repulsive fields, moves the robot towards a lower potential and therefore to the goal. Besides the advantage of a fast computation of the path, a disadvantage of the method is that

the robot can get trapped in a local minimum. In [130, 94], a simulated annealing approach is integrated into the potential field. This allows the robot to escape from local minima. The work in [104] proposes a repelling potential that reduces oscillations when the obstacle is close to the target and shows the effectiveness for the navigation of a mobile robot. In [18], path planning and path tracking via optimal control are combined for UAVs. A distributed control problem is considered in [82]. Here multiple autonomous vehicles communicate in a decentralized architecture, whereas each vehicle plans its local path by an APF such that no collisions with the other vehicles occur. In [19], APF are combined with a global path planning approach for an environment with static and dynamic obstacles. First a collision free path is planned by a roadmap method. Then an APF approach uses this path as an attractive field to navigate safely through the moving obstacles. However, methods of APF do not consider the optimality of the motion with respect to the dynamics of the system. They are not able to include the dynamics or information on predicted obstacle positions in any way.

2.2.3. Incremental Sampling-Based Approaches

A main drawback of roadmap methods is the explicit representation of the obstacle in the C-space. To overcome this time consuming problem, sampling-based approaches were developed. These methods are proven to be effective in high dimensional spaces. Instead of explicitly modeling the C-space, the methods randomly samples new configurations which are then checked against collisions, and finally added as feasible nodes to the graph. Thus a roadmap is built based on random samples, for which a path is searched by a graph-search method. The most relevant methods are probabilistic roadmaps (PRM) [55] and rapidly-exploring random trees (RRT) [68]. An overview on sampling-based approaches is given in [53, 27]. Probabilistic roadmaps sample a random node in the C-space. If the node is not within an obstacle, the sampled node is further used, otherwise it is discarded. In the next step, all neighboring nodes within a ball of a certain radius are connected to the sampled node if the connecting link is also proven to be free of collision. The sampling process is finally stopped if a maximum number of samples is reached. They have been shown to be probabilistically complete, which means that if a collision-free path exists, the probability to find this will increase to one, as the number of samples increases. Different sampling strategies exist, e.g., based on Voronoi diagrams [123], Gaussian distribution around the obstacle [11] or strategies for narrow passages [45]. During the last decades PRM algorithm found great attention in robotic path planning, see [67, 51, 44].

Instead of connecting all possible neighbors within a certain radius as it is done in PRM, the sampled configuration is connected to the nearest neighbor in the RRT approach. Hence, the RRT grows a tree starting from the root node, and is biased toward the target. Applications can be found for urban driving scenarios e.g. in [65], where the sampling strategy of the RRT planner is biased by reference values

of a controller. See [127] for a UAV, and [100] for a robot manipulator example.

If a detected obstacle or the robot itself moves, a replanning of the previously calculated path is necessary. Instead of replanning the tree from scratch, the work in [28] removes invalid parts and extends the remaining tree, until a solution is found. An extension to systems with space-time constraints is given in [131]. Here the C-space is extended by a further time dimension for each future point in time. An improvement in the search efficiency is given by the method RRT-Connect [63]. This procedure uses two trees. Both trees are biased by a heuristic to rapidly connect. The advantage of using two trees is that both trees can be computed in parallel. For some applications, where an initial solution must be found within an strict upper time bound, Anytime RRT [29] has been developed. The algorithm generates an initial suboptimal solution path quickly, and then iteratively improves the solution during computation. As iteration proceeds, the path costs are guaranteed to be reduced, since only nodes are added to the tree that contribute to a new path with lower costs.

Since RRT was proven to only converge to a suboptimal solution with probability one, an asymptotically optimal version of RRT is developed, namely RRT*, see [53]. RRT* differs from RRT in two ways. Starting with a new sample, a set of nearest nodes from the new sample, which are within a ball of a certain radius, is identified. Then, the new sample is connected to the neighboring node that minimizes the total path costs most. Secondly, all neighboring nodes are reconnected to the sampled node, if they can also reduce their overall path costs. An anytime version of the RRT* is given in [52].

Since in the sampling-based approaches mentioned above, the dynamics are not considered, the obtained paths are only understood to be geometrically feasible. This means that planned geometric path, with difficult maneuvers, can not be followed by the dynamics. Thus, situations can occur in which a collision is unavoidable. To overcome this problem, "kinodynamic" approaches were developed. Kinodynamic means that velocity, acceleration, and/or torque bounds must be satisfied together with kinematic constraints. Accordingly, the set of new samples can be reduced to samples satisfying these constraints. In [70], a new state is first sampled and then the corresponding control inputs are determined, which force the system to move towards the sampled state. Connecting sampled states with the tree by optimal controllers is shown in [121]. While the sampled states can be exactly reached, a sampled state automatically becomes a node of the tree, if the resulting path is also free of collisions. Neighboring states, which are needed for the growing of the tree, are identified by reachability analysis. A kinodynamic RRT for a robotic manipulator is described in [64]. Instead of planning with the full dynamics, the authors consider joint acceleration limits in their planning procedure and can show to be faster than planning with the system dynamics. Especially for problems with nonlinear dynamics, [111] embeds a nonlinear programming (NLP) solver in an RRT* algorithm. This allows to address nonlinear dynamics and smoothness conditions on the trajectories [111].

Indeed, sampling-based approaches can be used in high C-spaces, but it must be noted that they are only probabilistically optimal with respect to special metrics. They are not able to integrate a performance function that also considers economic aspects like energy consumption etc. The paths that are generated by the geometric planner are not smooth, because one would not iterate infinitely in practice. Thus, the executed paths seem to be jagged, and each node is reached with velocity zero. Additionally, the dynamics is not guaranteed to follow this paths. On the other side, kinodynamic planner solve the problem of dynamic feasibility, but suffer from high computation times, since the sampling space grows by velocity, acceleration and torque dimension. Additionally, including predicted information to these methods is a further computational burden.

2.2.4. Optimization-Based Approaches

A large body of work exists on optimization-based approaches in the obstacle avoidance of point-to-point problems, as they arise for manipulators, UAVs or mobile robot navigation. The idea of these methods is to describe the problem as an optimal control problem, and then to solve it by numerical optimization. Constrained point-to-point optimization problems are commonly solved by direct optimization methods [90] for reasons of speed. A well-known method is *model predictive control* (MPC) (see e.g. [83]). This powerful method allows constraints, such as obstacles, to be included in the problem formulation. However, solving optimal control problems is time demanding, which is why MPC is often limited to small horizons and/or slow system dynamics. To reduce this disadvantage, different means were proposed: efficiency may be increased by replacing state constraints though penalty terms or barrier functions [119, 97] or by use of *move-blocking strategies*, which fix the inputs over a certain period of time [34]. The exploitation of block diagonal structures in the Hessian and Jacobian matrices, combined with direct multiple shooting can reduce computation time as well [58, 23]. While for convex search spaces significant progress has been made, the situation for the steering scenario with collision avoidance is different, as the presence of obstacles leads to non-convexity of the admissible space. The solution by techniques like nonlinear programming (such as SQP) often leads to large computation times and trajectories which are far from optimal. An alternative is the formulation as mixed-integer program (MIP), in which the admissible space is split into sets of convex regions. A feasible sequence of such regions is mapped into binary variables [10, 24, 99, 22]. Solvers for these problems determine optimal sequences by techniques of branch-and-bound or branch-and-cut with embedded linear, quadratic or nonlinear programs. But even for relatively small problem instances, the combinatorial complexity limits the applicability in real-time.

To avoid the online solution of constrained optimization problem, methods like *explicit MPC* where developed [114, 129]. They use e.g. multi-parametric programming within offline computation of suitable controllers by splitting the solution space

of the parametric program into regions with constant control laws. For this method, the reduced effort for online computation is paired with high memory requirements and thus is limited to low system dimensions and small horizons.

Some research work in this context has employed *motion primitives* of which more complex motions are composed: In [107], primitives were defined as the solution of linear optimal control problems, and the primitives are combined to more complex motions through a sequence of subgoals. To achieve obstacle avoidance, the sequence is determined by an RRT search, i.e., the same restrictions to online computability as described above apply and only static obstacles are considered. In [91], the possible inputs of a linear dynamic model of a robot motion are classified as a finite set of basic actions used in motion planning for a system with 2 DOF without obstacles. The works in [48, 42] describe motions by a system of differential equations, called "dynamic movement primitives", which can be adapted to new conditions such as varying start and goal states. In [93], dynamic motion primitives are combined with potential field methods for path planning to achieve collision avoidance. Primitives such as "forward", "sideward" etc. for helicopter flight motion planning are presented in [31], and the investigation of basic control actions in the input space was presented in [83]. In [88], the learning of complex motions build from simple templates in a probabilistic setting by reinforcement learning is proposed. However, the learning is based on time consuming repetition of behaviors, and it does not consider kinematic and dynamic constraints, as well as predictions on time-varying obstacles.

2.2.5. Homotopy Based Methods in Optimization

Instead of finding solution trajectories from scratch, the use of homotopies provides a mechanism to select a desirable solution from a set of given homotopic solutions. The property of homotopy basically describes the process of continuously "deforming" one problem into another, see [38] for a general introduction to homotopy. This property is used for example, in the semi-analytic solution of non-linear differential equations, known as the Homotopy Analysis Method (HAM), [72, 71]. The HAM uses the concept of homotopies to generate and solve a convergent series of nonlinear systems. The nonlinearities in each series are handled by Maclaurin or Taylor series expansion. The HAM starts with a simplified initial guess of the problem and continuously transforms its solution to the original problem by increasing the homotopy parameter. The work of [128] uses the HAM to solve a linear optimal control problem. By means of calculus of variations and Pontryagin's maximum principle, the optimization problem is transformed into a two point boundary value problem which is subsequently solved by the HAM. Disadvantages are that the presented semi-analytic solutions take long computation times and that HAMs can not determine solutions for constrained problems, as it is the case when an obstacle is present.

In addition to using HAM for this semi-analytic method, homotopy properties are

also employed for tracking a solution path. In [2, 117], the Homotopy Continuation Method (HCM) is described. The principle is to trace the solution of an easy to solve problem, while smoothly transforming the problem to the original, while using the previous solution as a new initialization of the problem. A method based on HCM combined with the potential field method is defined in [115]. The method defines straight lines that cross through the target point and uses these lines for constructing the homotopy function. A collision-free path is then determined by taking such a line as an attractive field, while the obstacles as described by a repulsive field. The HCM is then applied on these overlapped fields. The solution of a nonlinear control problem by the HCM is described in [96]. The authors generate a parameter-dependency of the problem by varying the initial state. Finally they obtain a parameter-dependent control problem which is solved by an LQ controller. According to this method, the state trajectory and the control input of the nonlinear system can be determined in cases of a change in the initial state. Due to the iterative procedure of the HCM methods, solutions of the original nonlinear problem are obtained by continuous deformation of the problem.

There is also a group of methods that first compute so called "homotopy classes", and then searches for a path in a given class, see [39]. A homotopy class defines a set of trajectories with the same initial and final state, and which can be continuously transformed into another without intersecting an obstacle. In [9], the non-convex search space is separated into homotopy classes to constrain the search to collision-free areas. Then a cost minimal path of a homotopy class is determined by means of graph-search methods. A classification into "homotopy classes" is also used in [40], where a path in the homotopy class is determined by an RRT algorithm. There, tree expansion in the RRT method is only allowed in directions which satisfy a given homotopy class. However, these classification problems are limited to static environments, respectively, an efficient update of the classification procedure to dynamic environments is an area of further research.

2.3. Contribution of this Thesis

The review of the existing approaches for point-to-point motion problems under non-convex constraints shows some methods that are especially useful for path planning problems. However, when system dynamics is taken into account, the listed approaches are either not able to incorporate them or suffer from a high computational effort. This effort results from either C-space determination in Roadmap methods, sampling in high dimensions in sampling-based approaches, or the formulation of complex optimization problems. The mapping of obstacles into the C-space leads to complex shapes and is time demanding, which is why the proposed method in this thesis avoids these bottlenecks. The method developed here leaves the obstacle avoidance problem in euclidean space.

On the other side, the homotopy based methods, as described in Section 2.2.5,

show that solutions on computational expansive optimization problems can be obtained when taking advantage of homotopy properties. The aim of this thesis is to embed homotopy properties in optimization based control to benefit from two advantages: Firstly, the consideration of constraints, optimality, and system dynamics, by formulation of an optimization problems. Secondly, the use of homotopy properties to quickly select solutions from a set of trajectories which are obtained from a previous simple calculation. For this purpose, a set of homotopic trajectories is provided, such that the circumvention problem can then be condensed to the search of collision-free trajectories only in this set.

The main contribution of this thesis is the introduction of an algorithmic control procedure based on homotopic trajectories. This method provides near to optimal trajectories for non-convex control problems with low effort. During this work, the suggested method is adapted to new problem classes. Starting from a problem with fixed initial and goal states the method is subsequently adapted to integrate information on predicted obstacle positions, to time-varying goal states, and nonlinear system dynamics. Continuing by considering the dynamic system no longer as a moving point in space, but also as a geometric body for collision avoidance. This circumstance is simplified by the most methods by neglecting the geometry of the own system. Finally a cooperative and distributed control framework is addressed in this work, and a procedure is introduced which solves such collision avoidance problems. Thus, the method provides a comprehensive tool for trajectory planning problems, for which optimized solutions can be determined quickly.

3. Definitions and Preliminaries

This chapter presents some basic notations and theoretical background on linearization, similarity transformations, matrix decomposition, controllability, model predictive control (MPC), linear matrix inequalities (LMI's), and mixed-integer programming (MIP). This background is used for the later development of the presented control methods in this work.

3.1. System Dynamics

Let a nonlinear, continuous-time system by given by:

$$\dot{x}(t) = f(x(t), u(t)). \tag{3.1}$$

The state vector is $x(t) \in \mathbb{R}^{n_x}$ and the input vector $u(t) \in \mathbb{R}^{n_u}$. The nonlinear function $f(\grave{})$ maps the states and inputs into the state space.

Linear Discrete-Time Dynamics:

This thesis focuses on discrete-time systems, for which sampling instants are given by $t_k = t_0 + k\Delta t$, $\Delta t \in \mathbb{R}_{>0}$, with $t_0 \in \mathbb{R}_{\geq 0}$, and $k \in \mathbb{N}_{\geq 0}$. The discrete-time state and input vectors are denoted by $x_k := x(t_k)$, $u_k := u(t_k)$. A linear discrete-time system is written as:

$$x_{k+1} = Ax_k + Bu_k,\ k \in \mathbb{N}_{\geq 0}, \tag{3.2}$$

with time-invariant state matrix $A \in \mathbb{R}^{n_x \times n_x}$ and input matrix $B \in \mathbb{R}^{n_x \times n_u}$. Predictions of these vectors for $j \in \mathcal{J}_N := \{1, ..., N\}$ time steps are denoted by x_{k+j}. A prediction of x at time $k+j$, determined with information at time k is denoted by $x_{k+j|k}$. A trajectory is given by $\hat{x}^i = (x_0^i, ..., x_N^i)$, respectively for the setting with predictions $\hat{x}^i_{:|k}$. The index $i \in \mathcal{M} := \{0, 1, ..., n_c\}$ denotes the i-th trajectory of a set of trajectories $\mathcal{X}_b = \{\hat{x}^0, ..., \hat{x}^{n_c}\}$. The symbol ":" is a placeholder recording all points of time.

Linear Time-Varying System (LTV):

In Part II of this thesis, linear time-varying systems of the form:

$$x_{k+1} = A_k x_k + B_k u_k \tag{3.3}$$

are considered, for which A_k, and B_k can change arbitrarily over time.

Linear Differential Inclusion (LDI):

For a linear time-varying system as described above, a high number of time steps $j \in \mathcal{J}_{\mathcal{N}}$ obviously increases the complexity in the offline controller synthesis of Part II. A convex approximation of the nonlinear change in matrices A_k, and B_k, (3.3) can be described by a polytopic linear differential inclusion (LDI), see [12]:

$$x_{k+1} \in \mathbf{A}x_k + \mathbf{B}u_k, \tag{3.4}$$

where $\mathbf{A} := \mathbf{Co}\{A_0, ..., A_N\}$, and $\mathbf{B} := \mathbf{Co}\{B_0, ..., B_N\}$, are convex polytopic sets. The convex hull of a matrix $\mathbf{A}$ can also be described by the linear parameter-dependent matrix:

$$\mathbf{A}(\alpha) = A_0^{'} + \sum_{i=1}^{m} \alpha_i A_i^{'}, \quad \sum_{i=1}^{m} \alpha_i = 1,\ \alpha_i \geq 0, \tag{3.5}$$

with m matrices $A_i' \in \mathbb{R}^{n_x \times n_x}$ as the vertices of the matrix polytope $\mathbf{A}$, and A'_0 as the affine term.

3.2. Linear Matrix Inequalities

Linear Matrix Inequalities (LMI's), see [25], are used to define a convex set of matrices and constraints. A strict LMI is given by:

$$M(\alpha) = M_0 + \sum_{i=1}^{m} \alpha_i M_i > 0, \tag{3.6}$$

with variable α_i as described in (3.5). The matrices $M_i \in \mathbb{S}^n_{>0}$ are symmetric and positive-definite, denoted by $M_i \in \mathbb{S}^n_{>0}$ with dimension $n \times n$. The LMI (3.6) is a convex constraint on the definiteness of a matrix, which arises often in control theory e.g., for Lyapunov stabilitiy conditions or input constraints. When the matrices M_i are diagonal, the LMI becomes a set of linear inequalities.

Schur complement: The main use of the Schur complement is to transform quadratic matrix inequalities into equivalent LMI's. Consider the quadratic matrix inequality:

$$R(\alpha) > 0, \quad Q(\alpha) - S(\alpha)R^{-1}(\alpha)S^T(\alpha) > 0, \tag{3.7}$$

where $Q(\alpha) = Q^T(\alpha) \in \mathbb{S}^m$, $R(\alpha) = R^T(\alpha) \in \mathbb{S}^n$, and $S(\alpha) \in \mathbb{R}^{m \times n}$. The matrix $R(\alpha)$ is invertible. All matrices are linear in α, see (3.6). Applying Schur's formula, (3.7) is equivalent to:

$$M(\alpha) = \begin{pmatrix} Q(\alpha) & S(\alpha) \\ S^T(\alpha) & R(\alpha) \end{pmatrix} > 0. \tag{3.8}$$

Thus, matrix $M(\alpha)$ is positive-definite if (3.7) and $R(\alpha)$ are positive-definite.

Congruence Transformation: The definiteness of the symmetric matrix $Q \in \mathbb{S}^n$:

$$Q(\alpha) > 0, \tag{3.9}$$

is invariant under multiplication with an invertible matrix $W(\alpha) \in \mathbb{R}^{n \times n}$ from both sides, such that the following statement holds:

$$W(\alpha)Q(\alpha)W^T(\alpha) > 0, \quad \alpha \in \mathbb{R}^m, \tag{3.10}$$

see [43].

Change of Variables: By defining new variables, it is sometimes possible to transform nonlinear matrix inequalities into LMI's, see [43]. Given an LMI with matrix variables $P(\alpha) \in \mathbb{S}^n_{>0}$, $F(\alpha) \in \mathbb{R}^{n \times n}$, and $G \in \mathbb{R}^{n \times n}$, with $P(\alpha)$ and $F(\alpha)$ linearly dependent on α, the nonlinear matrix inequality:

$$F(\alpha)P^{-1}(\alpha) + P^{-1}(\alpha)G > 0, \tag{3.11}$$

with $Q(\alpha) := P^{-1}(\alpha)$ becomes:

$$F(\alpha)Q(\alpha) + Q(\alpha)G > 0. \tag{3.12}$$

Since (3.12) is still nonlinear, a second change $L(\alpha) := F(\alpha)Q(\alpha)$ yields the resulting LMI:

$$L(\alpha) + Q(\alpha)G > 0. \tag{3.13}$$

3.3. Pseudoinverse

A pseudoinverse matrix $M^\dagger \in \mathbb{C}^{m \times n}$, with $n \neq m$, is the generalization of the matrix inverse to non-square matrices $M \in \mathbb{C}^{n \times m}$. The pseudoinverse is used e.g. in least square regression problems, which arise for over- or underdetermined systems of linear equations [6].

Definition 3.1. *The Moore–Penrose pseudoinverse of $M \in \mathbb{C}^{n \times m}$, with $n \neq m$, is a unique matrix $M^\dagger \in \mathbb{C}^{m \times n}$, with the following properties, see [6]:*

- $MM^\dagger M = M$
- $M^\dagger MM^\dagger = M^\dagger$
- $(MM^\dagger)^H = MM^\dagger$
- $(M^\dagger M)^H = M^\dagger M$,

where M^H denotes the Hermitian. If the matrix M has only real entries, the Hermitian equals the transposed: $M^H = M^T$.

Overdetermined System of Linear Equations $\mathbf{n > m}$

For the case of $n > m$, and $\mathbf{rank}(M) = m$, matrix $(M^H M)$ is invertible and an algebraic expression of the pseudoinverse can be computed as described in [6]:

$$M^\dagger = (M^H M)^{-1} M^H. \tag{3.14}$$

In this case, $M^\dagger$ is said to be the left inverse of M: $M^\dagger M = I \in \mathbb{R}^{n \times n}$. The pseudoinverse provides a solution to the least square problem of an overdetermined system of linear equations. Such overdetermined systems have more equations than unknowns:

$$Mx = b. \tag{3.15}$$

Underdetermined System of Linear Equations $\mathbf{n < m}$

In the case of $n < m$, the linear system (3.15) *is said to be underdetermined (more unknowns than equations). Here, the pseudoinverse is given by:*

$$M^\dagger = M^H (M M^H)^{-1}. \tag{3.16}$$

The pseudoinverse $M^\dagger$ provides a least square solution to (3.15)*, and the matrix $M^\dagger$ is called the right inverse of M: $MM^\dagger = I \in \mathbb{R}^{m \times m}$. Proofs can be found in [6].*

The pseudoinverse can be efficiently computed by the QR-decomposition method, or singular value decomposition, see below.

3.4. Matrix Decomposition

Singular Value Decomposition

Consider a matrix $M \in \mathbb{R}^{n \times m}$ with the matrix rank: $\mathbf{rank}(M) = r$. Then M can be factored as described in [13]:

$$M = \mathbf{U}\mathbf{S}\mathbf{V}^T, \tag{3.17}$$

where:

$$\mathbf{U} = \text{is the matrix of eigenvectors of } MM^T,\ \mathbf{U} \in \mathbb{R}^{n \times r} \tag{3.18a}$$

$$\mathbf{S} = \sqrt{\mathbf{diag}(\mathbf{eig}(MM^T))},\ \mathbf{S} \in \mathbb{R}^{r \times r} \tag{3.18b}$$

$$\mathbf{V} = \text{is the matrix of eigenvectors of of } M^T M,\ \mathbf{V} \in \mathbb{R}^{m \times r} \tag{3.18c}$$

Here, **eig**() describes the operation of determining the vector of eigenvalues and **diag**() the diagonalization of a vector to a matrix.

Null Space

The null space (or Kernel) of a matrix $M \in \mathbb{R}^{n\times m}$, is a set of eigenvectors $\mathbf{u_i} \in \mathbf{U}$, which are obtained from the singular value decomposition of M, and are mapped to zero by M, see [13]:

$$\mathcal{N}(M) := \{\mathbf{u}_i | \mathbf{u}_i{}^T M = 0\}. \tag{3.19}$$

The vectors $\mathbf{u}_i \in \mathcal{N}(M)$ define the columns of the orthogonal basis matrix denoted by $n^\perp$:

$$n^\perp := (\mathbf{u}_1,\ \mathbf{u}_2, ...), \quad \mathbf{u}_i \in \mathcal{N}(M) \tag{3.20}$$

3.5. Linearization and Time Discretization

Since the control of continuous nonlinear systems, of type (3.1) can be challenging, a common approach is to linearize the system and to discretize the time axis.

Linearization: A linearization of $f(\cdot)$ as in (3.1) by the first-order Taylor series around a point of interest x_L, u_L, is given by:

$$\begin{aligned} f(x(t), u(t)) \approx f(x_L, u_L) &+ \left.\frac{\partial f(x(t), u(t))}{\partial x(t)}\right|_{x_L, u_L} (x(t) - x_L) \\ &+ \left.\frac{\partial f(x(t), u(t))}{\partial u(t)}\right|_{x_L, u_L} (u(t) - u_L). \end{aligned} \tag{3.21}$$

By resorting the terms, the linearized function can be written as:

$$\begin{aligned} f(x(t), u(t)) \approx &\left.\frac{\partial f(x(t), u(t))}{\partial x(t)}\right|_{x_L, u_L} x(t) + \left.\frac{\partial f(x(t), u(t))}{\partial u(t)}\right|_{x_L, u_L} u(t) \\ &+ f(x_L, u_L) - \left.\frac{\partial f(x(t), u(t))}{\partial x(t)}\right|_{x_L, u_L} x_L - \left.\frac{\partial f(x(t), u(t))}{\partial u(t)}\right|_{x_L, u_L} u_L, \end{aligned} \tag{3.22}$$

or in brief as:

$$\dot{x}(t) = A_c x(t) + B_c u(t) + \underbrace{f(x_L, u_L) - (A_c x_L + B_c u_L)}_{=: r_c}, \tag{3.23}$$

with constant affine term $r_c \in \mathbb{R}^{n_x}$, and definitions:

$$A_c = \left.\frac{\partial f(x(t), u(t))}{\partial x(t)}\right|_{x_L, u_L}, \quad B_c = \left.\frac{\partial f(x(t), u(t))}{\partial u(t)}\right|_{x_L, u_L}. \tag{3.24}$$

If x_L, u_L are chosen as steady state, then:

$$f(x_L, u_L) = 0, \tag{3.25}$$

and the affine term becomes $r_c = 0 \in \mathbb{R}^{n_x}$.

Zero-Order-Hold (ZOH) Discretization: A ZOH discretization with sampling time $\Delta t > 0$ leads to a discrete-time model:

$$x_{k+1} = Ax_k + Bu_k + r, \tag{3.26}$$

with matrices:

$$A = e^{A_c \Delta t}, \quad B = \int_0^{\Delta t} e^{A_c \tau} B_c \, \mathrm{d}\tau, \quad r = \int_0^{\Delta t} e^{A_c \tau} r_c \, \mathrm{d}\tau, \tag{3.27}$$

which exactly describes the linearized dynamics (3.26) at the sampling times.

3.6. Similarity Transformation

A similarity transformation offers the possibility to change the state vector x_k to $\tilde{x}_k$. It is used when focusing on the control of special state combinations, or when transformations between different spaces are of interest, see [3].

Definition 3.2. *Two state models, as given in* (3.26)*, described by matrices* (A, B, r) *and* $(\tilde{A}, \tilde{B}, \tilde{r})$ *are said to be similar, if an invertible matrix* $T \in \mathbb{R}^{n_x \times n_x}$ *exists, such that one system can be transformed into the other.*

Example:

Consider the affine transformation with a given T*:*

$$\tilde{x} = Tx. \tag{3.28}$$

The similarity transformation of (3.26) *follows from inserting* (3.28) *into* (3.26)*:*

$$\tilde{x}_{k+1} = Tx_{k+1} = \underbrace{TAT^{-1}}_{\tilde{A}} \tilde{x}_k + \underbrace{TB}_{\tilde{B}} u_k + \underbrace{Tr}_{\tilde{r}}. \tag{3.29}$$

The transformations to obtain the system matrices $(\tilde{A}, \tilde{B}, \tilde{r})$ *can be seen in* (3.29)*.*

3.7. Controllability

Controllability is an important property of a dynamic system. Required for stabilizability of dynamic systems by feedback control. A complete controllable system describes the ability of an input $u_k \in \mathbb{R}^{n_u}$, to move the state x_k from the initial state $x_0 = 0$, at time $k = 0$, to any other final state $x_k = x_f \in \mathbb{R}^{n_x}$.

Given the solution x_k of a linear discrete-time system, as given in (3.2), by:

$$x_k = A^{k-0} x_0 + \sum_{l=0}^{k-1} A^{k-l-1} B u_l, \quad k > 0. \tag{3.30}$$

Lemma 3.1. *A discrete-time system* (3.2) *is controllable, if the controllability matrix* $C_{AB} = (B, AB, ..., A^{n_x - 1}B)$ *has full rank:* $\mathbf{rank}(C_{AB}) = n_x$*, see [3].*

The controllability matrix results from the system of linear equations:

$$\left(B, AB, A^2B, ..., A^{n_x-1}B\right) \begin{pmatrix} u_{n_x-1} \\ u_{n_x-2} \\ \vdots \\ u_1 \\ u_0 \end{pmatrix} = x_f, \tag{3.31}$$

which has a unique solution, if $\mathbf{rank}(C_{AB}) = n_x$. The system of linear equations is given by the matrix notation of:

$$\sum_{l=0}^{n_x-1} A^{n_x-l-1}Bu_l = x_f, \tag{3.32}$$

which equals (3.30) for $x_0 = 0$.

$\mathfrak{r}$-step Controllability

Definition 3.3. *A dynamic system* (3.2) *is said to be* $\mathfrak{r}$*-step controllable, if an input sequence* $u_k \in \mathbb{R}^{n_u}$*, for* $0 \leq k \leq \mathfrak{r}-1$ *exists, such that the state* $x_k = 0$ *can be driven to any* $x_f \in \mathbb{R}^{n_x}$ *in at most* $\mathfrak{r} \in \mathbb{N}_{>0}$ *time steps.*

Lemma 3.2. *Given the* $\mathfrak{r}$*-step controllability matrix:*

$$C_{AB\mathfrak{r}} = \left(B, AB, A^2B, ..., A^{\mathfrak{r}-1}B\right), \tag{3.33}$$

The minimum number time steps $\mathfrak{r} \in \mathbb{N}_{>0}$*, in which the system* (3.2) *can reach every final state* x_f*, is given by the next higher integer value of* $\mathfrak{r} = \lceil n_x/n_u \rceil$*, and the condition that* $\mathbf{rank}(C_{AB\mathfrak{r}}) = n_x$*.*

Proof. If $\mathfrak{r} = \lceil n_x/n_u \rceil$, and $\mathbf{rank}(C_{AB\mathfrak{r}}) = n_x$, x_f is uniquely determined by:

$$\left(B, AB, A^2B, ..., A^{\mathfrak{r}-1}B\right) \begin{pmatrix} u_{\mathfrak{r}-1} \\ u_{\mathfrak{r}-2} \\ \vdots \\ u_1 \\ u_0 \end{pmatrix} = x_f, \tag{3.34}$$

for $x_0 = 0$. □

3.8. Model Predictive Control

Model Predictive Control (MPC) is a control method, typically based on discrete-time models, which determines the optimal future control inputs, and is often used for complex and constrained systems with many variables [83]. The procedure

uses a model of the system to be controlled, depending on a sequence of discrete input signals. The number of the available future input signals $u_{k+j|k}$ follows from $j \in \mathcal{J}_{\mathcal{H}} := \{0, ..., H\}$, where H is called the prediction horizon. The index $k+j|k$ of $u_{k+j|k}$ denotes the input u at point of time $k+j$ determined based on information at time k. To determine the optimal input signal at time k, an optimization problem is solved by minimizing a given performance measure, like the weighted sum of the quadratic deviation of the states and inputs to their steady state x_f, u_f, and an additional end cost term:

$$\min_{x_{k+j|k},\, u_{k+j|k}} \quad \|x_{k+H|k} - x_f\|^2_{Q_{end}} + \sum_{j=0}^{H-1} \|x_{k+j|k} - x_f\|^2_Q + \|u_{k+j|k} - u_f\|^2_R \tag{3.35a}$$

$$\text{s.t.} \quad x_{k+j+1|k} = f(x_{k+j|k}, u_{k+j|k}),\ \forall j \in \mathcal{J}_{\mathcal{H}}, \tag{3.35b}$$

$$x_{k|k} = x_s, \tag{3.35c}$$

$$x_{k+j|k} \notin \mathcal{P}_x, \quad \forall j \in \mathcal{J}_{\mathcal{H}}, \tag{3.35d}$$

$$u_{k+j|k} \in \mathcal{U}, \quad \forall j \in \mathcal{J}_{\mathcal{H}} \setminus \{H\}, \tag{3.35e}$$

with $\|x_{k+H|k} - x_f\|^2_{Q_{end}} = (x_{k+H|k} - x_f)^T Q_{end}(x_{k+H|k} - x_f)$, where $Q_{end}, Q \in \mathbb{S}^{n_x}_{>0}$, and $R \in \mathbb{S}^{n_u}_{>0}$. The set $\mathcal{P}_x$ denotes a set of forbidden states. The advantage of this method is that especially constraints on states and input can be considered. While the input sequence is determined over the complete prediction horizon, only the input for the next time step is applied. After execution, the new state is measured, the prediction horizon is shifted forward by one step, and the optimization is repeated. If on the one side, the prediction horizon is long enough, such that the final state is reachable, hard constraints can be formulated for reaching them. In the case of a too short time horizon, such hard formulations can lead to infeasible solutions, because the solvability of the optimization problem is not guaranteed. Furthermore, a strict formulation of these constrains implies high numerical effort. A simplification may be achieved by formulating the end constraints as penalty terms. Besides the question of solvability of an optimization problem like (3.35), a stabilizing control of the system from its initial to the final state requires further investigations in recursive feasibility. While criteria on recursive feasibility can be derived for linear systems without constraints, the proof for more complex problems becomes difficult. Nevertheless, MPC often requires significant computing capacities such that dynamically fast systems with many variables and long prediction horizons are difficult to control in real-time.

3.9. Mixed-Integer Programming

In the case of mixed variables, i.e. the co-existence of continuous variables $x_{k+j|k}$, $u_{k+j|k}$, $j \in \mathcal{J}_{\mathcal{H}}$ and additional discrete variables like e.g. binary variables $b_f \in \{0, 1\}$, the MPC optimization problem (3.35) becomes a mixed-integer program (MIP), [95]. Such formulations often arise when e.g. a dynamical system should circumvent

an obstacle. There, the collision avoidance constraints can be formulated by a binary half-plane representation of the obstacle and disjunctive programming. In the case of linear dynamics the MIP becomes a mixed-integer quadratic program (MIQP), as the cost function is quadratic.

Consider a problem of a disjunctive constraint, where one of the linear constraints $c_1 x \leq d_1$ and $c_2 x \leq d_2$ is exclusively active. This disjunction can be implemented by means of binary variables b_f as follows:

$$c_1 x \leq d_1 + M b_1 \tag{3.36a}$$
$$c_2 x \leq d_2 + M b_2 \tag{3.36b}$$
$$b_1 + b_2 = 1, \tag{3.36c}$$
$$b_f \in \{1, 0\}, \forall f \in \{1, 2\}. \tag{3.36d}$$

The variable M is a large positive number. If e.g. $b_1 = 1$ is chosen, constraint (3.36a) is relaxed, which means that this constraint is always satisfied. Regarding (3.36c), b_2 has to be zero, which activates (3.36b). Consider non-convex constraints, as they arise in obstacle avoidance problems, where $x \notin \mathcal{P}_x := \{x |\ Cx \leq d\} \subseteq \mathbb{R}^{n_x}$ with $C \in \mathbb{R}^{c \times n_x}$ and $d \in \mathbb{R}^c$, and c_f as the f-th row of C and d_f as the f-th entry of d. A realization with binary variables is given as:

$$c_f x \leq d_f + M b_f, \quad \forall f \in \{1, ..., c\} \tag{3.37a}$$
$$\sum_{f=1}^{c} b_f = c - 1 \tag{3.37b}$$

The formulation is known as "big-M" method [102], [122] and is used in a wide range of applications. However, the large variable M may cause numerical instability and the class of MIQP problems is much harder to solve, why alternative formulations of the problem should be preferred.

Part II.

Homotopic Control Algorithms for Linear Systems

4. Optimizing Control using Homotopy Properties

For human-robot cooperation in industrial processes, it is of great interest to combine the advantages of human flexibility and robotic precision in the same working area. Assume that a robotic manipulator has to accomplish tasks in two different positions and has to move between these points, while at the same time a human worker operates in the same region and may block the manipulator in its motion between the two positions. The task of the robot is now to compute a circumventing trajectory quickly to protect the human worker against collisions. This chapter is concerned with the control of systems in a partially constrained environment. The task considered here is to control a system from a fixed initial to a fixed target state, while the system has to react to new state constraints. The constraints formulate a forbidden region in the state space that has to be circumvented by the dynamic system. This chapter provides a solution for determining circumventing trajectories when an obstacle is detected. Information about position predictions are not available. For planning, the obstacle is assumed to be static for the rest time, but replanning is performed when a new detection (e.g. at the next point of time) updates the position information. In practice one can think of a robot, equipped with near-field sensors, detecting an obstacles. There, also only the current position of the obstacle is available. When the obstacle appears during execution, one strategy is to immediately stop the system in a safe state that is not colliding with the obstacle. If the obstacle disappears, the system resumes its trajectory to the target state. This strategy suffers from a high inefficiency, especially when the obstacle will not disappear. An alternative strategy is to determine each future state of the obstacle such that the resulting trajectory is feasible with respect to the system dynamics. The result is a collision-free trajectory at any time. This approach is commonly used in Model Predictive Control (MPC). While this strategy provides a significantly better efficiency, it suffers from high computation times, such that this method is often not capable to determine a solution timely, even for optimization over relatively small time horizons. The challenge is to obtain a solution with low effort to be online applicable, while maintaining a close to optimal solution. Therefore, this chapter introduces a method that uses homotopic trajectories. Homotopic trajectories are generated from a set of trajectories that represent different solutions for the transition problem from the initial to the final state. This set of trajectories is determined in advance and is assumed to be given in this chapter. They are termed as *base trajectories* and are determined to span a space allowing the system

to circumvent the obstacle. A homotopic target trajectory is then selected by interpolation of the *base trajectories*, see Sec. 4.1. The transition between homotopic target trajectories is realized by controls synthesized offline. In addition, an online procedure algorithmically selects a suitable homotopic target trajectory for obstacle circumvention.

The following parts of this chapter start by giving an introduction to homotopic functions used in this thesis Sec. 4.1. The formulation of the control problem is given in Sec. 4.2, which is followed by a transformation of the linear system dynamics into the homotopy space Sec. 4.3. The synthesis of offline controllers is shown in Sec. 4.4. The online procedure is described in Sec. 4.5, which is separated into an online procedure of mapping the obstacle vertices into the homotopy space Sec. 4.5.1 and the procedure of selecting the homotopic target trajectory online Sec. 4.5.2. The main ideas of this chapter are published by the author in [61].

4.1. Homotopic Functions

This section introduces the concept of homotopy and explains the term of homotopic trajectories. The concept of homotopic trajectories is used in this thesis to efficiently describe a set of "deformable" trajectories for obstacle circumvention by means of a homotopy parameter. In general, the idea of a homotopy is that a function can be continuously deformed into the other by means of a continuous homotopy parameter λ in the range of $[0, 1]$.

Definition 4.1. *Let two continuous functions $f, g : \mathbb{X} \to \mathbb{Y}$ map from the metric space $\mathbb{X}$ to $\mathbb{Y}$. A homotopy is a mapping:*

$$H : \mathbb{X} \times [0, 1] \to \mathbb{Y}, \tag{4.1}$$

with homotopy parameter $\lambda \in [0, 1]$, and the property $H(x, 0) = f(x)$ and $H(x, 1) = g(x)$.

A homotopic function is illustrated in Fig. 4.1. The interpolation between the functions $f(x)$ and $g(x)$ is given by λ. Within the scope of this thesis, discrete-time systems are considered, where the continuous functions $f(x)$, and $g(x)$ represent solutions of the difference equations in (3.2), evaluated at discrete-time steps $k \in \mathcal{J}_{\mathcal{N}}$. All trajectories are defined in the same time domain $k \in \mathcal{J}_{\mathcal{N}}$. Let the state and input trajectories be denoted by $\hat{x} = (x_0, ..., x_N)$ and $\hat{u} = (u_0, ..., u_N)$. A pair $(\hat{x}^i, \hat{u}^i)$, with $i \in \mathcal{M} := \{0, ..., n_c\}$, is the i-th input and state trajectory with $\hat{x}^i \in \mathcal{X}_b := \{\hat{x}^0, ..., \hat{x}^{n_c}\}$, and $\hat{u}^i \in \mathcal{U}_b := \{\hat{u}^0, ..., \hat{u}^{n_c}\}$ of $n_c + 1$ many trajectories. A state of trajectory i at time k is denoted by x_k^i, and a input by u_k^i. The state trajectories $\hat{x}^i \in \mathcal{X}_b$ share the same initial state $x_0^i = x_s$, for all $i \in \mathcal{M}$.

In Part II of this thesis, the end state of all trajectories is considered to be equal $x_N^i = x_f$ for all $i \in \mathcal{M}$, while Part III considers different end states $x_N^i = x_f^i$ for

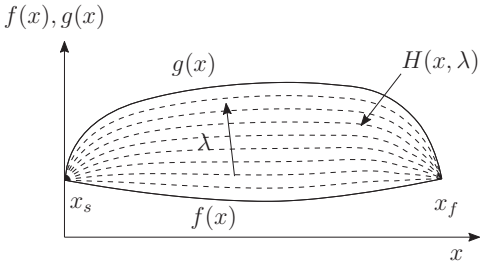


Figure 4.1.: Shows the two functions $f(x)$, $g(x)$, and the homotopic functions $H(x, \lambda)$. The functions $f(x)$, and $g(x)$ have the same initial and final points x_s, x_f.

all $i \in \mathcal{M}$, $(x_f^0 \neq x_f^1 \neq x_f^2 \neq ...)$. This can be seen in Fig. 4.2 for the trajectories $\hat{x}^0$, and $\hat{x}^1$. The figure shows the two discrete state trajectories $\hat{x}^0$, $\hat{x}^1$, and the homotopic trajectories given by $H(\hat{u}^0, \hat{u}^1, \lambda)$. For one point of time k, at trajectory $\hat{x}^0$, one can note the change of the homotopic state when it moves from trajectory $\hat{x}^0$ to trajectory $\hat{x}^1$ for λ changing from 0 to 1. A trajectory $\hat{x}^i$ can be interpreted as the image of a function: $\hat{x}^i = F^i(\hat{u}^i)$.

The following definition now defines a *homotopic* function, in the sense of linear interpolation, build from a set of given base trajectories $\hat{x}^i$. The :

Definition 4.2. *For a set of $i \in \{0, ..., n_c\}$ continuous functions $F^i : \mathbb{R}^{n_u \times N} \to \mathbb{R}^{n_x \times N}$, $i \in \mathcal{M}$, a vectorized homotopy is defined by:*

$$H : \left(\mathbb{R}^{n_u \times N}\right)^{n_c} \times [0,1]^{n_c} \to \mathbb{R}^{n_x \times N}. \tag{4.2}$$

The second argument is a vector of homotopy parameters $\boldsymbol{\lambda} := (\lambda^1, ..., \lambda^{n_c})^T \in \mathbb{R}^{n_c}$, with:

$$\lambda^i \in [0,1], \ \sum_{i=1}^{n_c} \lambda^i \leq 1. \tag{4.3}$$

The linear vectorized homotopy function is given by a matrix:

$$F = (F^1(\hat{u}^1) - F^0(\hat{u}^0), ..., F^{n_c}(\hat{u}^{n_c}) - F^0(\hat{u}^0)), . \tag{4.4}$$

and with $\lambda^0 := 1 - \sum_{i=1}^{n_c} \lambda^i$ according to:

$$H(\hat{u}^0, \ldots, \hat{u}^{n_c}, \boldsymbol{\lambda}) = \sum_{i=0}^{n_c} \lambda^i \cdot F^i(\hat{u}^i) \tag{4.5a}$$

$$= F^0(\hat{u}^0) + \sum_{i=1}^{n_c} (F^i(\hat{u}^i) - F^0(\hat{u}^0)) \cdot \lambda^i \tag{4.5b}$$

$$= F^0(\hat{u}^0) + F \cdot \boldsymbol{\lambda}. \tag{4.5c}$$

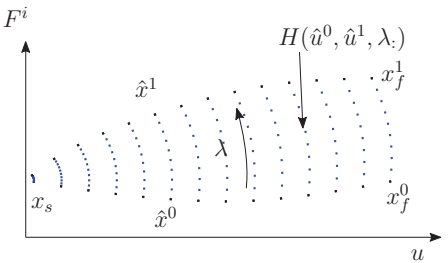


Figure 4.2.: Time discrete trajectories $\hat{x}^0$, $\hat{x}^1$, and the homotopic function $H(\hat{u}^0, \hat{u}^1, \lambda_:)$. $\lambda_:$ denotes a constant homotopy parameter over time.

Following Def. 4.2, homotopic trajectories are linear interpolations between trajectory $\hat{x}^0$ and the other trajectories $\hat{x}^i$, $i \in \mathcal{M} \setminus 0$. If, e.g., $\lambda^i = 1$ is chosen, while the other components of $\boldsymbol{\lambda}$ are zero, the trajectory $\hat{x}^i$ is obtained by (4.5c). While (4.5c) refers to complete trajectories, a homotopic state at a single point of time k is denoted by $x_k(\boldsymbol{\lambda}_k)$. It lies in between the states x_k^i, $i \in \mathcal{M}$, and is identified by the homotopy value $\boldsymbol{\lambda}_k$. Therefore the homotopic states as well as the inputs at time k are given by:

$$x_k(\boldsymbol{\lambda}_k) := x_k^0 + D_{x,k}\boldsymbol{\lambda}_k, \tag{4.6}$$

$$u_k(\boldsymbol{\lambda}_k) := u_k^0 + D_{u,k}\boldsymbol{\lambda}_k. \tag{4.7}$$

The vector $\boldsymbol{\lambda}_k := (\lambda_k^1, ..., \lambda_k^{n_c})^T \in \mathbb{R}^{n_c}$ denotes the vector of homotopy parameters at time k. The matrices:

$$D_{x,k} = (x_k^1 - x_k^0, ..., x_k^{n_c} - x_k^0) \in \mathbb{R}^{n_x \times n_c}, \tag{4.8}$$

$$D_{u,k} = (u_k^1 - u_k^0, ..., u_k^{n_c} - u_k^0) \in \mathbb{R}^{n_u \times n_c}, \tag{4.9}$$

are the differences between the state and input vectors of the trajectories $\hat{x}^i$, $\hat{u}^i$ $i \in \mathcal{M} \setminus 0$ with respect to the states and inputs of $\hat{x}^0$, and $\hat{u}^0$. A constant homotopy value over time is denoted by index ":", hence $\boldsymbol{\lambda}_:$. A homotopic trajectory with constant homotopy vector is then denoted by $\hat{x}(\boldsymbol{\lambda}_:)$, and $\hat{u}(\boldsymbol{\lambda}_:)$.

Lemma 4.1. *Let the linear system dynamics* (3.3) *and trajectories* $(\hat{x}^i, \hat{u}^i)$, $i \in \mathcal{M}$ *be given. A change of a homotopic state* $x_k(\boldsymbol{\lambda}_k)$ *from e.g.* $\boldsymbol{\lambda}_k = 0^{n_c}$ *to any* $\boldsymbol{\lambda}_k$, *according to* (4.6), *leads to an equivalent change in the input space according to* $u_k(\boldsymbol{\lambda}_k) := u_k^0 + D_{u,k}\boldsymbol{\lambda}_k$.

Proof. With the homotopy state notation $x_k(\boldsymbol{\lambda}_k)$, the system dynamics can be given by:

$$x_{k+1}(\boldsymbol{\lambda}_{k+1}) = Ax_k(\boldsymbol{\lambda}_k) + Bu_k. \tag{4.10}$$

Considering that a homotopic trajectory $\hat{x}(\boldsymbol{\lambda}_:)$ is described by a constant homotopy parameter, one can write $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k$. Inserting (4.6) into (4.10) yields:

$$x_{k+1}^0 + D_{x,k+1}\boldsymbol{\lambda}_: = Ax_k^0 + AD_{x,k}\boldsymbol{\lambda}_: + Bu_k. \tag{4.11}$$

With $D_{x,k+1} = AD_{x,k} + BD_{u,k}$, it follows:

$$x_{k+1}^0 + (AD_{x,k} + BD_{u,k})\boldsymbol{\lambda}_: = Ax_k^0 + AD_{x,k}\boldsymbol{\lambda}_: + Bu_k \tag{4.12a}$$

$$Ax_k^0 + Bu_k^0 + (AD_{x,k} + BD_{u,k})\boldsymbol{\lambda}_: - Ax_k^0 - AD_{x,k}\boldsymbol{\lambda}_: = Bu_k \tag{4.12b}$$

$$B(u_k^0 + D_{u,k}\boldsymbol{\lambda}_:) = Bu_k \tag{4.12c}$$

$$u_k^0 + D_{u,k}\boldsymbol{\lambda}_: = B^\dagger Bu_k. \tag{4.12d}$$

Since the matrix $B \in \mathbb{R}^{n_x \times n_u}$ has $n_x \geq n_u$, the pseudoinverse $B^\dagger$ is a left inverse, and from Sec. 3.3 it is known that $B^\dagger B = I$. It follows that (4.12d) equals (4.7). □

Due to linearity of the homotopic states (4.6), and inputs (4.7), each homotopic state and input is always located between the edges x_k^i, u_k^i. It follows that convex constraints like e.g. input limitations, or convex permissible state regions, imposed on the trajectories $(\hat{x}^i, \hat{u}^i)$, are also satisfied for all homotopic trajectories $(\hat{x}(\boldsymbol{\lambda}_:), \hat{u}(\boldsymbol{\lambda}_:))$, with constant homotopic parameter $\boldsymbol{\lambda}_:$. One can say that if the trajectories $(\hat{x}^i, \hat{u}^i)$ are determined to satisfy convex constraints, the homotopic trajectories $(\hat{x}(\boldsymbol{\lambda}_:), \hat{u}(\boldsymbol{\lambda}_:))$ also do. The purpose of the trajectories $\hat{x}^i \in \mathcal{X}_b$ is to span a suitable state region, in which later an efficient circumvention of an obstacle is possible.

4.2. Problem Description

Definition 4.3. *A base trajectory of the system* (3.2) *is denoted by* $\hat{x}^i = (x_0^i, ..., x_N^i)$, *consisting of* $N+1$ *states, where* i *is a trajectory with index* $i \in \mathcal{M} := \{0, ..., n_c\}$ *denoting trajectories starting at a common initial state* $x_0^i = x_s$. *The set* $\mathcal{X}_b$ *defines the set of all base trajectories:* $\mathcal{X}_b := \{\hat{x}^0, ..., \hat{x}^{n_c}\}$. *All base trajectories differ from each other in their states* x_k^i, *expect for* $k = 0$ *and eventually* $k = N$*:*

$$\forall \hat{x}^i \in \mathcal{X}_b : \; x_k^0 \neq x_k^1 \neq ... \neq x_k^{n_c}, \forall k \in \{1, ..., N-1\}. \tag{4.13}$$

Same corresponds to the input trajectories $\hat{u}^i$. *The base trajectory indexed by* $i = 0$ *are optimal with respect to a given performance measure, while the other are used to span the space for circumvention.*

This chapter deals with the problem of reaching a fixed target state, e.g. a static assembly point which a robotic manipulator has to reach. Thus all *base trajectories* that span the space of circumvention will have the same target state. The following assumption is made.

Assumption 4.1. *All base trajectories have the same target state,* $(x_f^0 = x_f^1 = ... = x_f^{n_c})$.

Since the *base trajectories* do not change over time, as initial and target states are fixed, the assumption satisfies the fact that the *base trajectories* can be provided offline by the user. They are needed for the offline controller synthesis and the later introduced online control procedure.

The dynamics considered here is linear and discrete in time, see (3.2). By means of the given *base trajectories*, all homotopic states and inputs are computable according to (4.6) and (4.7) of Sec. 4.1. While many obstacles, that may block the robot in its motion can be at least approximated as convex polytopes, the developed approaches in this work consider such convex polytopes.

Definition 4.4. *Generally, a time-varying convex polytope can be described by a finite set of linear inequalities, known as the half-space representation:*

$$\mathcal{P}_{x,k} := \{x_k |\; C_k x_k \leq d_k\} \subseteq \mathbb{R}^{n_x}, \; k \in \{1, ..., N\} \tag{4.14}$$

with $C_k \in \mathbb{R}^{c \times n_x}$ and $d_k \in \mathbb{R}^c$. It can also be defined by a finite set of n_v vertices p_l at time k:

$$\mathcal{P}_{xv,k} := \{p_l \in \mathbb{R}^{n_x} |\ (l \in \mathbb{N}_{>0}) \leq n_v\} = \{p_1, ..., p_{n_v}\}, p_l \in \mathbb{R}^{n_x}, n_v \in \mathbb{N}. \tag{4.15}$$

Besides the general Definition 4.4 for time-varying convex polytopes, this chapter assumes the polytopes to be static for the planning procedure. That does not mean that the obstacle cannot move, but in the case of movement, the procedure replanns the trajectory. By means of this, the collision avoidance procedure is based on information available from current time. A more general case, where information on the predicted obstacle location are available, is discussed in a later chapter.

Consider the case in which system (3.2) follows the optimal trajectory $\hat{x}^0$ from the initial state $x_0 = x_s$ towards $x_N = x_f$, until the obstacle $\mathcal{P}_{x,k^*}$ is detected at time $k^* \in \{1, \ldots, N-1\}$, such that $\hat{x}^0$ can not be followed further. To allow for a feasible solution, obviously $x^0_{k^*} \notin \mathcal{P}_{x,k^*}$ and $x^0_N \notin \mathcal{P}_{x,k^*}$ is required. The goal is to compute an optimized trajectory for the remaining time steps $\hat{x}^* = (x^*_{k^*}, ..., x^*_N) \in \mathbb{R}^{n_x \times (N+1-k^*)}$ and $\hat{u}^* = (u^*_{k^*}, ..., u^*_{N-1}) \in \mathbb{R}^{n_u \times (N-k^*)}$, while avoiding any collision with the polytope. The remaining time steps starting from k^* are denoted by the index $k^* + j$ with $j \in \mathcal{J}_{k^*} := \{0, \ldots, N-1-k^*\}$. The selection of the best among the feasible trajectories is based on the following quadratic performance criterion:

$$J(x_{k^*+j},\ u_{k^*+j}) = \sum_{j=0}^{N-1-k^*} \|x_{k^*+j} - x_f\|^2_Q + \|u_{k^*+j} - u_f\|^2_R, \tag{4.16}$$

with positive-definite symmetric weighting matrices $Q \in \mathbb{S}^{n_x}_{>0}$ and $R \in \mathbb{S}^{n_u}_{>0}$. A formulation of the trajectory optimization problem, starting at detection time k^*, is then given by:

$$\min_{x_{k^*+j},\ u_{k^*+j}} \quad J(x_{k^*+j},\ u_{k^*+j}) \tag{4.17a}$$

$$\text{s.t.} \quad x_{k^*+j+1} = Ax_{k^*+j} + Bu_{k^*+j}, \tag{4.17b}$$

$$x_{k^*+j} \notin \mathcal{P}_{x,k^*} \quad \forall j \in \mathcal{J}_{k^*} \tag{4.17c}$$

$$x_{k^*} = x^0_{k^*},\ x_N = x_f. \tag{4.17d}$$

The problem stated here is a non-convex problem. A possible approach is to solve the problem by MIP, here specifically mixed-integer quadratic programming (MIQP). In terms of the computation time, the problem is in most cases costly to solve, since solvers e.g., using branch-and-bound techniques [125] need to be used. These solve problem instances with integer variables encoding the separation of the free, from the occupied space. Large numbers of time steps increase the computation times for these methods considerably due to an increase number of integer variables. The objective of the presented method is to compute optimized trajectories according to the criteria stated above with significantly lower computational effort as by MIQP, while maintaining close-to-optimal solutions.

Working with homotopic trajectories, the challenge is to drive the system from an initially executed homotopic trajectory $\hat{x}(\boldsymbol{\lambda}_{:})$ (encoded by the constant homotopy parameter $\boldsymbol{\lambda}_{:}$), with states $x_k(\boldsymbol{\lambda}_{:})$, to a homotopic target trajectory $\hat{x}(\bar{\boldsymbol{\lambda}})$ for obstacle avoidance. The homotopic target trajectory is then denoted by the vector $\bar{\boldsymbol{\lambda}}$:

$$\hat{x}(\boldsymbol{\lambda}_{:}) \rightarrow \hat{x}(\bar{\boldsymbol{\lambda}}). \tag{4.18}$$

Additionally, the value $\bar{\boldsymbol{\lambda}}$ has to be chosen such that the resulting trajectory has low costs. This task is illustrated in Fig. 4.3, where the system starts following the optimal *base trajectory* $\hat{x}^0$, until the obstacle $\mathcal{P}_{x,k^*}$ is detected. Then the algorithmic

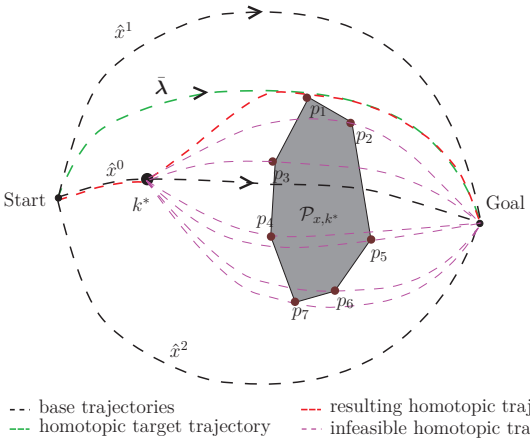


Figure 4.3.: Shows the selected homotopic target trajectory (green) by $\bar{\boldsymbol{\lambda}}$, and the resulting homotopic trajectory (red) toward $\bar{\boldsymbol{\lambda}}$. The infeasible trajectories either intersect $\mathcal{P}_{x,k^*}$ or have higher costs.

procedure determines online the best collision-free homotopic trajectory towards a target trajectory encoded by $\bar{\boldsymbol{\lambda}}$. Here, several possible target trajectories are analyzed according to feasibility and performance. The transition is realized by linear time-varying (LTV) controllers determined offline. To enable that a homotopic target trajectory $\bar{\boldsymbol{\lambda}}$ exists at all, the following assumption is necessary.

Assumption 4.2. *Let the set of base trajectories $\mathcal{X}_b$ contain at least one base trajectory $\hat{x}^i \in \mathcal{X}_b$ such that for $x^i_{k^*+j} \in \hat{x}^i$ it holds that $x^i_{k^*+j} \notin \mathcal{P}_x$, $\forall j \in \mathcal{J}_{k^*}$.*

This assumption is immediately justified by the fact that one cannot hope to find a feasible circumvention of $\mathcal{P}_{x,k^*}$ if the whole admissible space (as constructed by the choice of $\mathcal{X}_b$) is blocked. The assumption is not sufficient for finding a feasible solution, since it must be ensured that the transition to $\hat{x}(\bar{\boldsymbol{\lambda}})$ is achieved without intersecting $\mathcal{P}_{x,k^*}$.

4.3. Transformation into the Homotopy Space

The developed approach for obstacle circumvention exists of two parts:

1. Determination of a collision-free homotopic target trajectory with constant $\bar{\boldsymbol{\lambda}}_k$
2. Transition from current homotopic trajectory $\boldsymbol{\lambda}_k$ to the homotopic target trajectory $\bar{\boldsymbol{\lambda}}_k$

The first step is part of the online procedure, while the second step is realized by offline synthesized controllers. In order to synthesize controllers, the linear dynamic system (3.1) is first formulated in the space of homotopy parameters. This leads to a linear time-varying (LTV) system which is explained by the equations below. The transformation of the system dynamics in the homotopy space has a particular advantage. It allows one to describe complete trajectories by a single homotopy parameter, which leads to only optimizing over this parameter instead of each single state and input as it is done in an MPC.

Consider the task of steering (3.2) from a state $x_{k^*}(\boldsymbol{\lambda}_{k^*})$ to a future state on a homotopic target trajectory referring to $\bar{\boldsymbol{\lambda}}$. While the inputs for the currently executed trajectory and the targeted one are known from the homotopy function (4.7), the transition between these trajectories requires to extend the input by an additional value $u_{add,k} \in \mathbb{R}^{n_u}$, leading to:

$$u_{new,k}(\boldsymbol{\lambda}_k) := u_k(\boldsymbol{\lambda}_k) + u_{add,k}. \tag{4.19}$$

This superposition is a function depending on $\boldsymbol{\lambda}_k$, but may lead to signals $u_{new,k}(\boldsymbol{\lambda}_k)$ which are not in the set of homotopic input trajectories according to (4.7).

Replacing the state x_k of the system dynamics (3.2) by the homotopic states $x_k(\boldsymbol{\lambda}_k)$ of (4.6) and the input u_k by the new input $u_{new,k}(\boldsymbol{\lambda}_k)$ of (4.19), the dynamic becomes:

$$x_{k+1}(\boldsymbol{\lambda}_{k+1}) = Ax_k(\boldsymbol{\lambda}_k) + Bu_{new,k}(\boldsymbol{\lambda}_k), \tag{4.20}$$

which describes the progress of the homotopic state depending on the input $u_{new,k}(\boldsymbol{\lambda}_k)$. With the homotopic states (4.6) and inputs (4.7), equation (4.20) can be further rewritten to:

$$x^0_{k+1} + D_{x,k+1}\boldsymbol{\lambda}_{k+1} = A(x^0_k + D_{xk}\boldsymbol{\lambda}_k) + B(u^0_k + D_{u,k}\boldsymbol{\lambda}_k + u_{add,k}) \tag{4.21a}$$

$$D_{x,k+1}\boldsymbol{\lambda}_{k+1} = (AD_{x,k} + BD_{u,k})\boldsymbol{\lambda}_k + Bu_{add,k} \tag{4.21b}$$

$$D_{x,k+1}\boldsymbol{\lambda}_{k+1} = D_{x,k+1}\boldsymbol{\lambda}_k + Bu_{add,k}. \tag{4.21c}$$

As it can be seen, the last equation (4.21c) represents a linear system of equations, with n_c unknowns from $\boldsymbol{\lambda}_{k+1}$, the known values $\boldsymbol{\lambda}_k$, and $u_{add,k}$. Thus, the system has n_c unknowns and n_x equations, because of $D_{x,k+1} \in \mathbb{R}^{n_x \times n_c}$. In this context, three different cases have to be considered, that depend on the number of chosen *base trajectories*:

1. $n_x = n_c$, same number of state dimension and *base trajectories*
2. $n_x < n_c$, *underdetermined*: state dimension smaller than number of *base trajectories*
3. $n_x > n_c$, *overdetermined*: state dimension higher than number of *base trajectories*

In the first case, the number of *base trajectories* n_c is chosen equally to the dimension of the state space. This case provides a full-dimensional space for circumventing $\mathcal{P}_{x,k}$. A positive effect is, that $D_{x,k} \in \mathbb{R}^{n_x \times n_x}$ becomes quadratic and thus (4.21c) is uniquely solvable. In the second case $n_x < n_c$, more *base trajectories* are chosen than the dimension of the state space. This is plausible if one wishes to obtain a greater variety for collision avoidance by considering more *base trajectories* with different behaviors. Additionally, this leads to an *underdetermined* with fewer equations than unknowns, see Sec. 3.3 for explanation. The third case is especially important if a system has a high dimensional state space, the obstacle however exists in a much lower dimension such that one do not wants to determine so much *base trajectories*. In this case, the system has more equations than unknowns and is called *overdetermined*. In the following, the system dynamics in the homotopy space for the different cases of (4.21c) are derived.

Uniquely Solvable Homotopic System of Equations: $\mathbf{n_x = n_c}$

The matrices $D_{x,k+1} \in \mathbb{R}^{n_x \times n_x}$ are quadratic. They are assumed to have full rank:

$$\mathbf{rank}(D_{x,k+1}) = n_c, \ \forall k \in \{1, ..., N-1\}, \tag{4.22}$$

which means that all *base trajectories* differ from each other as given in Def. 4.3, and that the *base trajectories* span a full dimensional space at each time step. Thus, $\boldsymbol{\lambda}_{k+1}$ follows explicitly from (4.21c):

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + {D_{x,k+1}}^{-1} B u_{add,k}. \tag{4.23}$$

This equation describes the transition between different homotopic trajectories and leads to $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ via (4.6). The derived dynamic equation is a linear time-varying (LTV) system, because of the time varying matrix ${D_{x,k+1}}^{-1}$.

As it can be easily seen, a steady state of this system exists if $\boldsymbol{\lambda}_k$ becomes constant:

$$\bar{\boldsymbol{\lambda}} = \bar{\boldsymbol{\lambda}} + {D_{x,k+1}}^{-1} B \bar{u}_{add}, \tag{4.24}$$

which is the case when the additional input steady state is: $\bar{u}_{add} = 0$. This leads to the conclusion, that if $u_{add,k} = 0$ is chosen, the system stops transitioning to the homotopic target trajectory and moves along its current homotopic trajectory.

For realizing the transition between homotopic trajectories online, control laws with time-varying state feedback matrices $K_k \in \mathbb{R}^{n_u \times n_c}$ are used:

$$u_{add,k} := -K_k(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}) + \bar{u}_{add}. \tag{4.25}$$

Inserting (4.25) into (4.23) yields the closed-loop system:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - {D_{x,k+1}}^{-1} B K_k(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}) + {D_{x,k+1}}^{-1} B \bar{u}_{add} \tag{4.26a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - {D_{x,k+1}}^{-1} B K_k(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}) \quad | + \bar{\boldsymbol{\lambda}} - \bar{\boldsymbol{\lambda}} \tag{4.26b}$$

$$\boldsymbol{\lambda}_{k+1} = (I - {D_{x,k+1}}^{-1} B K_k)(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}) + \bar{\boldsymbol{\lambda}}, \tag{4.26c}$$

and with $\delta\boldsymbol{\lambda}_k := \boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}$, the closed-loop delta-system becomes:

$$\delta\boldsymbol{\lambda}_{k+1} = (I - {D_{x,k+1}}^{-1} B K_k)\delta\boldsymbol{\lambda}_k. \tag{4.27}$$

If the matrices $D_{x,k+1}$ have no full rank: $\mathbf{rank}(D_{x,k+1}) < n_c$, linearly depending columns of $D_{x,k+1}$ can be removed such that $n_x > n_c$. This circumstance would lead to the third case of an overdetermined system.

Underdetermined Homotopic System of Equations: $\mathbf{n_x < n_c}$

In the second case (4.21c) has more unknowns than equations. Solving (4.21c) according to $\boldsymbol{\lambda}_{k+1}$, yields the LTV-system:

$$\boldsymbol{\lambda}_{k+1} = D^{\dagger}_{x,k+1} D_{x,k+1} \boldsymbol{\lambda}_k + D^{\dagger}_{x,k+1} B u_{add,k}, \tag{4.28}$$

with the right inverse matrix $D^{\dagger}_{x,k+1} \in \mathbb{R}^{n_c \times n_x}$. In general, this means that the system of equations has unknowns free to be chosen. The use of the right inverse $D^{\dagger}_{x,k+1}$ in (4.28) corresponds to a least-square solution of (4.21c).

Lemma 4.2. *A steady state of the homotopy dynamics* (4.28) *is given by:*

$$\begin{aligned} \bar{\boldsymbol{\lambda}}_{under} &:= D^{\dagger}_{x,k+1} D_{x,k+1} \bar{\boldsymbol{\lambda}} \\ \bar{u}_{add} &= 0. \end{aligned} \tag{4.29}$$

Proof. A steady state is given, when $\boldsymbol{\lambda}_k$ is constant over time, hence $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k = \bar{\boldsymbol{\lambda}}_{under}$. Then (4.28) becomes:

$$\bar{\boldsymbol{\lambda}}_{under} = D^{\dagger}_{x,k+1} D_{x,k+1} \bar{\boldsymbol{\lambda}}_{under} + D^{\dagger}_{x,k+1} B \bar{u}_{add}. \tag{4.30}$$

With $\bar{u}_{add} = 0$, and (4.29), equations (4.30) becomes:

$$\bar{\boldsymbol{\lambda}}_{under} = D^{\dagger}_{x,k+1} D_{x,k+1} D^{\dagger}_{x,k+1} D_{x,k+1} \bar{\boldsymbol{\lambda}}. \tag{4.31}$$

With the property of the right inverse $D_{x,k+1} D^{\dagger}_{x,k+1} = I$, (4.31) becomes:

$$\bar{\boldsymbol{\lambda}}_{under} = D^{\dagger}_{x,k+1} D_{x,k+1} \bar{\boldsymbol{\lambda}}, \tag{4.32}$$

which proofs that the given steady state does not cause any changes in (4.30). □

With the control law:

$$u_{add,k} := -K_k(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}_{under}) + \bar{u}_{add}, \tag{4.33}$$

where $K_k \in \mathbb{R}^{n_u \times n_c}$, the closed-loop system of (4.28) becomes:

$$\boldsymbol{\lambda}_{k+1} = D^{\dagger}_{x,k+1} D_{x,k+1} \boldsymbol{\lambda}_k - D^{\dagger}_{x,k+1} B K_k (\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}_{under}) + D^{\dagger}_{x,k+1} B \bar{u}_{add}. \tag{4.34}$$

Subtracting on both sides $D^{\dagger}_{x,k+1} D_{x,k+1} \bar{\boldsymbol{\lambda}}_{under}$, and with (4.29), one gets:

$$\boldsymbol{\lambda}_{k+1} - \underbrace{D^{\dagger}_{x,k+1} D_{x,k+1} \bar{\boldsymbol{\lambda}}_{under}}_{\bar{\boldsymbol{\lambda}}_{under}} = (D^{\dagger}_{x,k+1} D_{x,k+1} - D^{\dagger}_{x,k+1} B K_k)(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}_{under}), \tag{4.35}$$

so that the closed-loop delta-system is obtained with $\delta\boldsymbol{\lambda}_k := \boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}_{under}$:

$$\delta\boldsymbol{\lambda}_k = (D^{\dagger}_{x,k+1} D_{x,k+1} - D^{\dagger}_{x,k+1} B K_k) \delta\boldsymbol{\lambda}_k. \tag{4.36}$$

Overdetermined Homotopic System of Equations: $\mathbf{n_x > n_c}$

Here, (4.21c) has more equations than unknowns, which in general yields a system that is not solvable. This third case is a very special case. No solution of the homotopy equation (4.21c) can be provided in one time step, only an approximation to this. This is given by the left inverse which provides a least-square solution of the system of equations:

$$\boldsymbol{\lambda}_{k+1} = D^{\dagger}_{x,k+1} D_{x,k+1} \boldsymbol{\lambda}_k + D^{\dagger}_{x,k+1} B u_{add,k}. \tag{4.37}$$

Again, a LTV-system occurs. Since $D_{x,k+1} \in \mathbb{R}^{n_x \times n_c}$ has more rows than columns, its left inverse $D^{\dagger}_{x,k+1}$ fulfills the property $D^{\dagger}_{x,k+1} D_{x,k+1} = I$, and (4.37) equals:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + D^{\dagger}_{x,k+1} B u_{add,k}, \tag{4.38}$$

for which a steady state is $\bar{\boldsymbol{\lambda}}$ and $\bar{u}_{add} = 0$.

The difficulty of this circumstance that no exact solution can be given in one time-step. This is shown exemplarily for the case of a two dimensional state space $n_x = 2$ and $n_c = 1$ in Fig. 4.4. Starting from the point of time k with a known value $\boldsymbol{\lambda}_k$ and a given input $u_{add,k}$, the state x_{k+1} may not be located in the homotopy space, given by $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ with $\boldsymbol{\lambda}_{k+1} \in [0,1]^{n_c}$. Hence the state is possibly not a homotopic state: $x_{k+1} \notin x_{k+1}(\boldsymbol{\lambda}_{k+1})$. This is obvious because we have explained that (4.37) has no exact solution. The solution of (4.38) now provides a value $\boldsymbol{\lambda}_{k+1}$ which minimizes the distance between the state x_{k+1}, and the nearest possible homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$. Similar to the uniquely solvable case, the control law (4.25) leads to the closed-loop delta-system:

$$\delta\boldsymbol{\lambda}_{k+1} = (I - D^{\dagger}_{x,k+1} B K_k) \delta\boldsymbol{\lambda}_k. \tag{4.39}$$

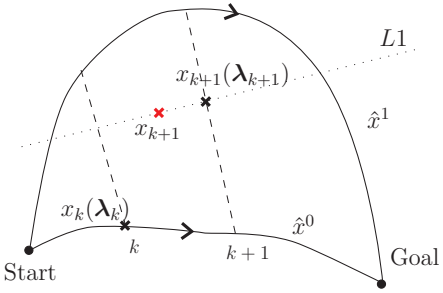


Figure 4.4.: Difference between homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$, and real state x_{k+1} for the overdetermined case.

However, when synthesizing controllers K_k, the following problem occurs: Since the controllers are synthesized in the homotopy space, which is lower-dimensional than the state space, $n_c < n_x$, and the task of the controllers is to steer the homotopic dynamics from the current to the targeted homotopic state:

$$\boldsymbol{\lambda}_k \rightarrow \bar{\boldsymbol{\lambda}}, \tag{4.40}$$

the use of the controllers in the state space x_k, by $u_{add,k}$ and $u_{new,k}$, according to (4.19), leads to a state x_{k+1} which can be located along the straight line $L1$ seen in Fig. 4.4. Hence, the synthesis of the control matrices K_k yields no unique solutions $K_k \in \mathbb{R}^{n_u \times n_c}$. The problem that now arises is that all equations and the circumvention procedure are determined in the homotopy space (remember the advantage of a smaller search space compared to the state space), but one has no guarantee that also no collision in the state space occurs.

This problem is solved in Chapter 5, where constraints on the time-varying control matrices K_k are derived that satisfy $x_k \in x_k(\boldsymbol{\lambda}_k)$ for all time steps.

4.4. Offline Controller Synthesis

This section first covers the part of the solution procedure which can be accomplished offline, namely the computation of controllers to transition to homotopic target states $\bar{\boldsymbol{\lambda}}$. The controller synthesis is shown for the closed-loop delta-system (4.27) of the uniquely solvable case in Sec. 4.3. The synthesis for the other case, with the closed-loop delta-systems (4.28), is realized in the same way, but with the corresponding matrices of this dynamics. Towards the goal of transitioning between the trajectories, semi-definite programming synthesizes stabilizing state feedback controllers based on Lyapunov stability. As performance measure for this system, consider the quadratic cost function with symmetric and positive-definite weighting

matrices $Q_C \in \mathbb{S}^{n_c}_{>0}$, and $R_C \in \mathbb{S}^{n_u}_{>0}$:

$$J(\delta\boldsymbol{\lambda}_k, \delta u_{add,k}) = \sum_{k=0}^{N-1} \delta\boldsymbol{\lambda}_k{}^T Q_C \delta\boldsymbol{\lambda}_k + \delta u_{add,k}{}^T R_C \delta u_{add,k}, \tag{4.41}$$

with:

$$\delta u_{add,k} := u_{add,k} - \bar{u}_{add}. \tag{4.42}$$

Compared to the general cost function given in (4.16), this cost function serves as a design criterion for the transition behavior by adjusting the weightings Q_C and R_C. The upcoming costs for transitioning from the actual trajectory at time k^* to the homotopic target trajectory are still determined according to the general cost function (4.16). With the control law (4.25), the cost function (4.41) can be written to:

$$J(\delta\boldsymbol{\lambda}_k, K_k) = \sum_{k=0}^{N-1} \delta\boldsymbol{\lambda}_k{}^T (Q_C + K_k{}^T R_C K_k) \delta\boldsymbol{\lambda}_k. \tag{4.43}$$

Definition 4.5. *[3] A discrete-time system $x_{k+1} = f(x_k, u_k)$, with steady state $\bar{x}$ is said to be stable, if a Lyapunov function $V_k(x_k)$ exists, such that the Lyapunov function does not increase over k, i.e.: $V_{k+1}(x_{k+1}) - V_k(x_k) \leq 0$ for all initial states $x_0 \in \mathcal{D}_{init} \subset \mathbb{R}^{n_x}$. If for the initial set applies $\mathcal{D}_{init} \in \mathbb{X}$, the system is said to be globally stable.*

For controller synthesis in the homotopy space, a quadratic Lyapunov function $V_k(\boldsymbol{\lambda}_k)$ is used:

$$V_k := \delta\boldsymbol{\lambda}_k{}^T P \delta\boldsymbol{\lambda}_k, \tag{4.44}$$

with matrix $P \in \mathbb{S}^{n_c}_{>0}$. The LTV system (4.27) is stable with a decreasing rate parametrized by the step costs of (4.41), if the following condition holds:

$$V_{k+1} - V_k \leq -\delta\boldsymbol{\lambda}_k{}^T (Q_C + K_k{}^T R_C K_k) \delta\boldsymbol{\lambda}_k. \tag{4.45}$$

Lemma 4.3. *With the Lyapunov function* (4.44)*, an upper bound on the expected value of the finite horizon costs* (4.43) *is given by the trace of matrix P:*

$$\mathbf{tr}(P) = \mathbf{E}(J(\delta\boldsymbol{\lambda}_k, K_k)) \tag{4.46}$$

Proof. When time advances from k to $k+1$, the single-step decrease of the quadratic Lyapunov function $V_k = \delta\boldsymbol{\lambda}_k{}^T P \delta\boldsymbol{\lambda}_k$ is bounded by the step costs, see (4.45). Summing up the decrease over all time steps yields:

$$\sum_{k=0}^{N-1} V_{k+1} - V_k \leq \sum_{k=0}^{N-1} -\delta\boldsymbol{\lambda}_k{}^T (Q_C + K_k{}^T R_C K_k) \delta\boldsymbol{\lambda}_k, \tag{4.47}$$

and thus:

$$V_0 - V_N \geq J(\delta\boldsymbol{\lambda}_k, K_k). \tag{4.48}$$

Since all *base trajectories* $\hat{x}^i$, $i \in \mathcal{M}$, terminate in x_f, the homotopy function $x_N(\boldsymbol{\lambda}_N)$ also equals x_f, regardless of the value of $\boldsymbol{\lambda}_N$. Thus, $\boldsymbol{\lambda}_N$ can be set to $\boldsymbol{\lambda}_N = \bar{\boldsymbol{\lambda}}$, and it follows that $\delta\boldsymbol{\lambda}_N = 0$. This observation leads to $V_N = 0$, such that V_0 becomes an upper bound of the cost value:

$$V_0 \geq J(\delta\boldsymbol{\lambda}_k, K_k). \tag{4.49}$$

When considering that the initial state $\delta\boldsymbol{\lambda}_0$ is selected by the rules of normal distribution, the expected value of the Lyapunov function is an upper bound on the expected costs of (4.43):

$$\mathbf{E}(V_0) \geq \mathbf{E}(J(\delta\boldsymbol{\lambda}_k, K_k)), \tag{4.50}$$

and from stochastic consideration one can show:

$$\mathbf{tr}(P) = \mathbf{E}(V_0), \tag{4.51}$$

when the mean value $\mathbf{E}(\delta\boldsymbol{\lambda}_0) = 0$ (as it is the case for a delta-system). So it is shown that the well-established theory on controller synthesis based on trace functions is suitable for the finite horizon case considered here. For detailed explanations on the quadratic form (4.51), see [103]. □

According to this upper bound, the optimization problem for the controller synthesis can be formulated as follows:

$$\min_{\delta\boldsymbol{\lambda}_k,\ P,\ K_k} \quad \mathbf{tr}(P) \tag{4.52a}$$

$$\text{s.t.} \quad \delta\boldsymbol{\lambda}_{k+1}{}^T P \delta\boldsymbol{\lambda}_{k+1} - \delta\boldsymbol{\lambda}_k{}^T P \delta\boldsymbol{\lambda}_k \leq -\delta\boldsymbol{\lambda}_k{}^T (Q_C + K_k{}^T R_C K_k)\delta\boldsymbol{\lambda}_k, \tag{4.52b}$$

$$\forall k \in \{0, ..., N-2\}.$$

The time step $k = N-1$, and its corresponding controller K_{N-1} are neglected, since the offline computed input $u_{N-1}(\boldsymbol{\lambda}_{N-1})$, that brings any homotopic states into the final state $x_N(\boldsymbol{\lambda}_N) = x_f$, is already known. As it can be seen, constraint (4.52b) depends nonlinearly on the variables $\delta\boldsymbol{\lambda}_k$, P, K_k. Transforming this constraint into a linear matrix inequality (LMI) turns problem (4.52) independent of $\delta\boldsymbol{\lambda}_k$ and provides an overall convex controller synthesis problem.

Lemma 4.4. *The non-convex constraint* (4.52b) *can be transformed into:*

$$(I - D_{x,k+1}{}^{-1} B K_k)^T P (I - D_{x,k+1}{}^{-1} B K_k) - P + Q_C + K_k{}^T R_C K_k \leq 0. \tag{4.53}$$

With $P = Y^{-1}$ and $K_k = L_k Y^{-1}$, and if a symmetric matrix $Y \in \mathbb{S}^{n_c}_{>0}$ and matrices $L_k \in \mathbb{R}^{n_u \times n_c}$ exist for all time steps $k = \{0, ..., N-2\}$, then also the following LMI holds:

$$\begin{pmatrix} Y & (Y - {D_{x,k+1}}^{-1} B L_k)^T & Y^T & {L_k}^T \\ Y - {D_{x,k+1}}^{-1} B L_k & Y & 0 & 0 \\ Y & 0 & {Q_C}^{-1} & 0 \\ L_k & 0 & 0 & {R_C}^{-1} \end{pmatrix} \geq 0, \tag{4.54}$$

in which 0 denotes zero matrices of appropriate dimensions.

Proof. Inequality (4.52b) can be reformulated by replacing $\delta \boldsymbol{\lambda}_{k+1}$ according to the homotopy dynamics (4.27) to obtain (4.53). If the latter is multiplied from the left and right by Y, and if the substitution $Y = P^{-1}$ is used, one gets:

$$\begin{aligned} (Y - {D_{x,k+1}}^{-1} B K_k Y)^T Y^{-1} (Y - {D_{x,k+1}}^{-1} B K_k Y) - Y \\ + Y^T Q_C Y + Y^T {K_k}^T R_C K_k Y \leq 0. \end{aligned} \tag{4.55}$$

Applying the Schur complement to (4.55) yields:

$$\begin{pmatrix} Y & (Y - {D_{x,k+1}}^{-1} B K_k Y)^T & Y^T & (K_k Y)^T \\ Y - {D_{x,k+1}}^{-1} B K_k Y & Y & 0 & 0 \\ Y & 0 & {Q_C}^{-1} & 0 \\ K_k Y & 0 & 0 & {R_C}^{-1} \end{pmatrix} \geq 0, \tag{4.56}$$

for which the substitution $L_k = K_k Y$ completes the proof. □

Since the cost function (4.46) depends on P and the matrix inequality (4.54) on Y, a closed upper bound $\tilde{P} \in \mathbb{S}^{n_c}_{>0}$ on P can be obtained from the non-strict inequality:

$$\tilde{P} \geq Y^{-1}. \tag{4.57}$$

Applying the Schur complement, this inequality is equivalent to:

$$\begin{pmatrix} \tilde{P} & I \\ I & Y \end{pmatrix} \geq 0. \tag{4.58}$$

The overall optimization problem for the controller synthesis of the LTV-system

(4.27) can now be summarized to:

$$\min_{\tilde{P},\, Y,\, L_k} \quad \mathbf{tr}(\tilde{P}) \tag{4.59a}$$

$$\text{s.t.} \quad \begin{pmatrix} Y & (Y - {D_{x,k+1}}^{-1} B L_k)^T & Y^T & {L_k}^T \\ Y - {D_{x,k+1}}^{-1} B L_k & Y & 0 & 0 \\ Y & 0 & {Q_C}^{-1} & 0 \\ L_k & 0 & 0 & {R_C}^{-1} \end{pmatrix} \geq 0, \tag{4.59b}$$

$$\forall k \in \{0, ..., N-2\}$$

$$\begin{pmatrix} \tilde{P} & I \\ I & Y \end{pmatrix} \geq 0 \tag{4.59c}$$

The result of the optimization is a time-varying linear quadratic regulator (LQR) for the LTV system (4.27). The controllers are obtained from the solution of the optimization problem and $K_k = L_k Y^{-1}$. The controllers are designed by fixed controller matrices Q_C and R_C that specify the transition behavior between homotopic trajectories and are used subsequently in the online procedure. The offline controller synthesis can be carried out with solvers like MOSEK [86].

4.5. Online Control with State Constraints

The online procedure is initiated upon detection of an obstacle at time k^*. For the planning process, the obstacle is assumed to be constant over time. Nevertheless, if the obstacle moves, a new circumventing homotopic trajectory is selected. The procedure explained here uses the advantage that when the obstacle blocks the actual motion, a new trajectory must not be determined for each state, as described in the problem description of Sec. 4.2. Instead, a homotopic target trajectory is selected, such that the resulting trajectory is free of collisions. The selected homotopic target trajectory should have lower costs compared to other possible once that can be chosen. The questions that arise are: Which value of $\bar{\boldsymbol{\lambda}}$ selects a homotopic target trajectory that circumvents the obstacle and has minimum costs for the overall transition to this trajectory and the movement on this. The approach considered in the first part of this section 4.5.1 is to define the homotopic target trajectories by so called *passing points*. These can be, e.g., the vertices p_l of the vertex representation $\mathcal{P}_{xv} = \{p_1, ..., p_{n_v}\}$ of the obstacle polytope $\mathcal{P}_x$ or other points on the surface of the polytope. To accomplish this, an algorithmic procedure is shown which maps the *passing points* from the state space into the homotopy space. Sec. 4.5.2 explains how a homotopic target trajectory is identified such that the system can transition to it safely and efficiently. To imagine how homotopic target trajectories are represented, it is referred to Fig. 4.3. In this illustration, the *passing points* correspond to the vertices of the polytope.

4.5.1. Mapping of Obstacle Passing Points into the Homotopy Space

The purpose of mapping points from the state space to the homotopy space is to assign a homotopy parameter to a selected point from the state space. Such a state space point is called a *passing point*:

Definition 4.6. *A passing point p_l with index l is a point which is located on the boundary $\partial\mathcal{P}_x$ of a given polytope $\mathcal{P}_x$:*

$$p_l \in \partial\mathcal{P}_x. \tag{4.60}$$

The set of passing points is denoted by:

$$\mathcal{P}_{pas} := \{p_l \in \mathbb{R}^{n_x} |\ (l \in \mathbb{N}_{>0}) \leq n_{pas}\} = \{p_1, ..., p_{n_{pas}}\}, \tag{4.61}$$

with n_{pas} as the total number of passing points.

Since the passing point describes a point on the surface of the obstacle, the corresponding homotopy parameter directly provides the homotopic trajectory which moves through this point. Thus a target trajectory is directly given after the mapping process.

Definition 4.7. *Let $\bar{\boldsymbol{\lambda}}(p_l) \in \mathbb{R}^{n_c}$ denote the vector of constant homotopy parameters of trajectory $\hat{x}(\bar{\boldsymbol{\lambda}}(p_l))$ running from the initial state x_0 through a passing point p_l of $\mathcal{P}_x$ to the final state x_f. The set of all passing points p_l of $\mathcal{P}_x$ mapped into the homotopy space is denoted by $\mathcal{P}_{\boldsymbol{\lambda}pas}$. The mapping of the passing points is a function with the properties:*

$$\texttt{MapPas} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_c},\ p_l \mapsto \bar{\boldsymbol{\lambda}}(p_l). \tag{4.62}$$

Generally, the mapping of a passing point from the state space to the homotopy space is not unique. This can be seen from the homotopy function (4.6). If p_l is used on the left side, the equation provides different solutions $\bar{\boldsymbol{\lambda}}(p_l)$ for different times k, since the matrix $D_{x,k}$ changes over time:

$$p_l = x_k^0 + D_{x,k}\bar{\boldsymbol{\lambda}}(p_l). \tag{4.63}$$

For the uniquely solvable case (Sec. 4.3, $n_x = n_c$) $D_{x,k} \in \mathbb{R}^{n_c \times n_c}$ is quadratic. A unique solution of (4.63) exists for each time $k \in \{0, ..., N\}$. Thus, a passing point p_l can be mapped $N-1$ times into the homotopy space (initial and end time are not considered). Each homotopy parameter is then denoted by $\bar{\boldsymbol{\lambda}}_k(p_l)$.

If $n_x < n_c$, which refers to the underdetermined case, the matrix $D_{x,k} \in \mathbb{R}^{n_x \times n_c}$ has more columns than rows and (4.63) has infinitely many solutions at each time step k. For the overdetermined case $n_x > n_c$, (4.63) is not uniquely solvable, but

the *passing point* p_l is approximated by a homotopy value $\bar{\boldsymbol{\lambda}}(p_l)$ by means of the pseudoinverse $D^{\dagger}_{x,k}$.

The circumstance that each *passing point* can get mapped multiple times, leads to a high number of homotopic target trajectories that have to be evaluated according to the costs, and checked for collisions. To avoid for this circumstance, which slows down the online method, a procedure is provided to map each *passing point* only once into the homotopy space:

1. Divide the space spanned space by the *base trajectories* into partitions, see Def. 4.8, and check in which of the partitions p_l is contained.

2. Determine for a selected homotopic partition that contains p_l a closest upper time bound $\bar{k}$ and lower time bound $\underline{k}$ with $\{\bar{k},\ \underline{k}\} \in \{0, ..., N\}$ and $\underline{k} < \bar{k}$, such that the point p_l is located in between $x_{\bar{k}}(\boldsymbol{\lambda}_{\bar{k}})$ and $x_{\underline{k}}(\boldsymbol{\lambda}_{\underline{k}})$. Finally determine the homotopy parameter $\bar{\boldsymbol{\lambda}}(p_l)$.

The definition of a homotopic partition is given as follows.

Definition 4.8. *A homotopy partition is defined by $n_x - 1$ entries λ^i, $i = \{1, ..., n_c\}$, of the homotopy vector $\boldsymbol{\lambda}$ being freely choosable, while the remaining are fixed to zero. Thus, the homotopy vector $\bar{\boldsymbol{\lambda}}(p_l)$ of point p_l only operates in the considered homotopic partition. The selection of a partition is made by a matrix $C_s \in \mathbb{R}^{n_c \times n_c}$. The corresponding homotopy parameter of point p_l in partition C_s is denoted by $\bar{\boldsymbol{\lambda}}(p_l, C_s)$:*

$$\bar{\boldsymbol{\lambda}}(p_l, C_s) := C_s \cdot \bar{\boldsymbol{\lambda}}(p_l). \tag{4.64}$$

C_s is diagonal matrix with entries $c_i \in \{0, 1\}$:

$$C_s = \begin{pmatrix} c_1 & & 0 \\ & \ddots & \\ 0 & & c_{n_c} \end{pmatrix}, \tag{4.65}$$

with the additional condition:

$$\sum_{i=1}^{n_c} c_i = n_x - 1. \tag{4.66}$$

According to (4.66)*, the total number of possible permutations s of C_s is given by:*

$$N_p = \begin{cases} \binom{n_c}{n_x - 1} & ,\ \textit{if } n_c \geq n_x - 1 \\ 1 & ,\ \textit{else} \end{cases}. \tag{4.67}$$

One of these is selected by index $s \in \{1, ..., N_p\}$. The set of all possible C_s is denoted by $\mathcal{C}$:

$$\mathcal{C} := \{C_s | c_i \in \{0, 1\}, \sum_{i=1}^{n_c} c_i = n_x - 1\}. \tag{4.68}$$

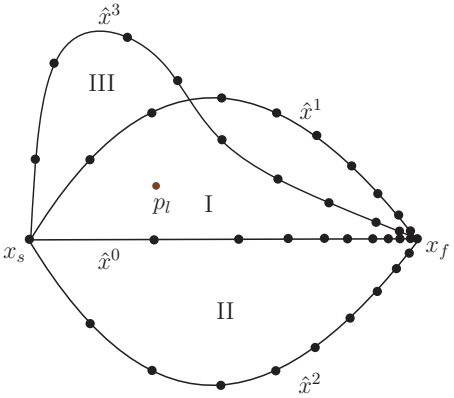


Figure 4.5.: Shows three different homotopic partitions. The point p_l is located in partition I and III.

An illustration of homotopic partitions is shown in Fig. 4.5 for the example of $n_x = 2$ and $n_c = 3$. With (4.67), $N_p = 3$, so that three partitions are possible:

I. the homotopy between $\hat{x}^0$, and $\hat{x}^1$: C_1, $\boldsymbol{\lambda} = (\lambda^1,\ 0,\ 0)^T$

II. the homotopy between $\hat{x}^0$, and $\hat{x}^2$: C_2, $\boldsymbol{\lambda} = (0,\ \lambda^2,\ 0)^T$

III. the homotopy between $\hat{x}^0$, and $\hat{x}^3$: C_3, $\boldsymbol{\lambda} = (0,\ 0,\ \lambda^3)^T$

In Fig. 4.5, p_l is located in partition I and III. Partition II can map the point only by negative values of the homotopy parameter which, however, are not in the range of permissible values. Subsequently, the values $\bar{\boldsymbol{\lambda}}(p_l, C_1)$, $\bar{\boldsymbol{\lambda}}(p_l, C_3)$ have to be determined.

In Fig. 4.6, the mapping procedure of the algorithm is exemplary shown for partition I. The first *Step (i)* linearly approximates the *base trajectories*. *Step (ii)* determines upper and lower time bounds $\bar{k}$ $\underline{k}$, in which point p_l is located between. *Step (iii)* finally determines the value of $\bar{\boldsymbol{\lambda}}(p_l, C_s)$ by a backtracking procedure:

Step (i): This step starts by selecting $\bar{k} = N - 1$, and $\underline{k} = 0$. Then each *base trajectory* $\hat{x}^i$ is linearly approximated by a continuous function between these times. The *base trajectory* $\hat{x}^0$ is approximated by $g^0(\tau)$ and $\hat{x}^1$ by $g^1(\tau)$ with continuous variable $\tau \in [0, 1]$. The approximations are colored orange in Fig. 4.6:

$$g^0(\tau) = x^0_{\underline{k}} + (x^0_{\bar{k}} - x^0_{\underline{k}})\tau, \tag{4.69}$$

$$g^1(\tau) = x^1_{\underline{k}} + (x^1_{\bar{k}} - x^1_{\underline{k}})\tau. \tag{4.70}$$

A homotopy between the two functions $g^0(\tau)$ and $g^1(\tau)$ for a point p_l (and of course for the homotopy subspace C_s of this example) is:

$$h(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s)) = g^0(\tau) + (g^1(\tau) - g^0(\tau))\bar{\boldsymbol{\lambda}}(p_l, C_s). \tag{4.71}$$

Inserting (4.69) and (4.70) into (4.71) leads to:

$$\begin{aligned} h(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s)) =& x^0_{\underline{k}} + (x^0_{\bar{k}} - x^0_{\underline{k}})\tau + (x^1_{\underline{k}} - x^0_{\underline{k}})\bar{\boldsymbol{\lambda}}(p_l, C_s) \\ &+ \left((x^1_{\bar{k}} - x^1_{\underline{k}}) - (x^0_{\bar{k}} - x^0_{\underline{k}})\right)\tau\bar{\boldsymbol{\lambda}}(p_l, C_s), \end{aligned} \tag{4.72}$$

which equals:

$$h(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s)) = x^0_{\underline{k}} + (x^0_{\bar{k}} - x^0_{\underline{k}})\tau + D_{x,\underline{k}}\bar{\boldsymbol{\lambda}}(p_l, C_s) + \left(D_{x,\bar{k}} - D_{x,\underline{k}}\right)\tau\bar{\boldsymbol{\lambda}}(p_l, C_s). \tag{4.73}$$

With given values of $\bar{k}$ and $\underline{k}$, (4.73) is a bilinear function in the variables $\bar{\boldsymbol{\lambda}}(p_l, C_s)$ and τ, which is hard to solve. Therefore, the bilinear equation (4.72) is conically approximated by setting $x^1_{\underline{k}} = x^0_{\underline{k}}$, respectively in (4.73), $D_{x,\underline{k}} = 0$, which leads to the red cone in Fig. 4.6, and the equation for the conic approximation $h_{con}(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s))$ becomes:

$$h_{con}(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s)) = x^0_{\underline{k}} + (x^0_{\bar{k}} - x^0_{\underline{k}})\ \tau + D_{x,\bar{k}}\ \tau\bar{\boldsymbol{\lambda}}(p_l, C_s). \tag{4.74}$$

This approximation can be easily solved for τ and $\bar{\boldsymbol{\lambda}}(p_l, C_s)$, by means of the substitution:

$$T_{con} := \tau\bar{\boldsymbol{\lambda}}(p_l, C_s) \tag{4.75}$$

Lemma 4.5. *The conic approximation $h_{con}(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s))$ in (4.74) equals the bilinear function $h(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s))$ in (4.73) for $\tau = 1$.*

Proof. $\tau = 1$ in (4.72) yields:

$$h(1, \bar{\boldsymbol{\lambda}}(p_l, C_s)) = x^0_{\bar{k}} + (x^1_{\underline{k}} - x^0_{\underline{k}})\bar{\boldsymbol{\lambda}}(p_l, C_s) + \left((x^1_{\bar{k}} - x^1_{\underline{k}}) - (x^0_{\bar{k}} - x^0_{\underline{k}})\right)\bar{\boldsymbol{\lambda}}(p_l, C_s) \tag{4.76a}$$

$$= x^0_{\bar{k}} + (x^1_{\bar{k}} - x^0_{\bar{k}})\bar{\boldsymbol{\lambda}}(p_l, C_s) \tag{4.76b}$$

$$= x^0_{\bar{k}} + D_{x,\bar{k}}\bar{\boldsymbol{\lambda}}(p_l, C_s) \tag{4.76c}$$

$$= h_{con}(1, \bar{\boldsymbol{\lambda}}(p_l, C_s)) \tag{4.76d}$$

□

The lemma states that the conic approximation is exact when $\tau = 1$. If p_l is exactly located on the homotopy function at the identified $\bar{k}$, one would immediately obtain the homotopy parameter by setting:

$$h_{con}(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s)) := p_l, \tag{4.77}$$

and solving for $\bar{\boldsymbol{\lambda}}(p_l, C_s)$. However, p_l does not necessarily lie on $\bar{k}$, but can be located between the discrete time-steps k. If p_l is then described by a $\tau < 1$ the

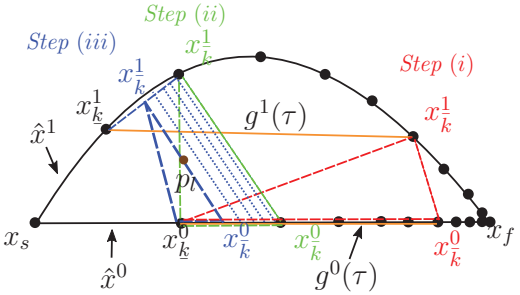


Figure 4.6.: Procedure for determining $\bar{\boldsymbol{\lambda}}(p_l, C_1)$.

conic approximation becomes inexact. Therefore, first the time bounds $\bar{k}$ and $\underline{k}$ have to be adjusted.

Step (ii): As it can be seen in the example in Fig. 4.6, the *passing point* p_l is not located near $\tau = 1$ so that the conic approximation (4.74) provides no satisfying solution. Furthermore, with the initially selected time bounds $\bar{k}$ and $\underline{k}$, the homotopic function (4.73) is not a good approximation of the real homotopy, since the linear approximations (4.69) and (4.70) already do not describe the *base trajectories* $\hat{x}^0$ and $\hat{x}^1$ sufficiently accurate. The task of this step is then to iteratively tighten the upper and lower time bounds $\bar{k}$ and $\underline{k}$, until p_l is located between the closest upper and lower time bound shown by the green cone. The algorithmic procedure is shown in Alg. 4.1 and starts by selecting a *passing point* $p_l \in \mathcal{P}_{pas}$ and a partition $C_s \in \mathcal{C}$. The conic approximation (4.74) (***(line 5)*** of Alg. 4.1) is solved. If τ was identified to be in the range $\tau \in [0, 1]$ the upper time bound $\bar{k}$ is set to the next integer value, see ***(line 8-10)***:

$$\tau_{int} = \lfloor \tau(\bar{k} - \underline{k}) + \underline{k} \rceil. \tag{4.78}$$

If, however, $\tau > 1$, which means that p_l is located near to a point of time which is greater than the actual upper bound $\bar{k}$, the lower time bound is first raised and the upper time bound is set to the old value of the previous iteration ***(line 12)***. Then τ is determined again with the conic approximation, but now with the new lower time bound $\underline{k}$, and the reduction of the upper time bound according to (4.78) follows ***(line 12-15)***. This procedure is repeated iteratively in a while loop until the upper bound $\bar{k}$ can not be reduced further ***(line 6-18)***.

Step (iii): Regarding to the example in Fig. 4.6, the problem that p_l is located between two adjacent points of time shows well that the green conic approximation provides an insufficient homotopy value. This is due to the fact, that only when p_l is located near to the upper time bound, with $\tau \approx 1$, where the conic approximation obtains the same solution as the homotopy function (explained in *Step (i)*). To achieve that p_l is located near to the upper time bound, a backtracking procedure

(line 19-27) is executed. The identified value $\bar{k}$ of *Step (ii)* is used to shift the values $x_{new} = x_{\bar{k}}^0$ and $D_{x,new} = D_{x,\bar{k}}$, piecewise towards $\bar{k} - 1$, by changing the values according to:

$$D_{x,new} := D_{x,\bar{k}-1} + (D_{x,new} - D_{x,\bar{k}-1})\varrho \tag{4.79a}$$

$$x_{new} := x_{\bar{k}-1} + (x_{new} - x_{\bar{k}-1})\varrho. \tag{4.79b}$$

Parameter $\varrho \in (0, 1)$ shrinks the values stepwise ***(line 22)*** or when replacing ϱ by a parameter $\upsilon > 1$, increases the values stepwise ***(line 24)***. The new values $D_{x,new}$ and x_{new} are finally inserted into the conic approximation (4.74), for which τ and $\bar{\boldsymbol{\lambda}}(p_l, C_s)$ are obtained. The procedure then stops, if the identified value τ is in the range:

$$1 - \sigma < \tau < 1 + \sigma, \tag{4.80}$$

with σ chosen as a small fault tolerance value. Additionally, this algorithmic part ***(line 20-30)*** terminates if the change in τ is smaller than a value $\beta \in (0, 1)$. The result of this step can be seen in Fig. 4.6, where the front border of the blue colored conic approximation is shifted backwards, until p_l is located near to a value of $\tau \approx 1$. This backtracking procedure finally yields the value $\bar{\boldsymbol{\lambda}}(p_l, C_s)$. The execution of Alg. 4.1 maps the *passing points* into the homotopy space, stored in the set $\mathcal{P}_{\boldsymbol{\lambda}pas}$.

Algorithm 4.1.: Mapping of *passing points*: `MapPas`

Given: $\mathcal{P}_{pas}$, n_c base trajectories, $C_s \in \mathcal{C}$, $\beta \in (0,1)$, $\nu > 1$, $\varrho \in (0,1)$, $\upsilon > 1$
Define: $\tau = 1$, $\tau_{int} = N-1$, $\tau_{intold} = 0$, $\bar{k} = N-1$, $\underline{k} = 0$, $\tau_{conold} = 0$
for all $p_l \in \mathcal{P}_{pas}$ **do**
  for all $C_s \in \mathcal{C}$ **do**
    compute τ, $\bar{\boldsymbol{\lambda}}(p_l, C_s)$ according to (4.74).
    while $\tau_{intold} \neq \tau_{int}$ & $\bar{\boldsymbol{\lambda}}(p_l, C_s) \geq \{0\}^{n_c}$ **do**
      if $\tau \in [0,1]$ **then**
        set $\tau_{intold} := \tau_{int}$.
        compute the nearest integer time $\tau_{int} = \lfloor \tau(\bar{k} - \underline{k}) + \underline{k} \rceil$.
        set $\underline{k} := \underline{k}$ and $\bar{k} := \tau_{int}$.
      else
        set $\underline{k} := \tau_{int}$ and $\bar{k} = \tau_{intold}$.
        compute τ, according to (4.74).
        compute the nearest integer time $\tau_{int} = \lfloor \tau(\bar{k} - \underline{k}) + \underline{k} \rceil$.
        set $\underline{k} := \underline{k}$ and $\bar{k} := \tau_{int}$.
      end if
      compute τ, $\bar{\boldsymbol{\lambda}}(p_l, C_s)$ according to (4.74).
    end while
    $D_{x,new} = D_{x,\bar{k}}$, $x_{new} = x_{\bar{k}}^0$
    while $|\tau_{conold} - \tau| < \beta$ **do**
      set $\tau_{conold} := \tau$.
      if $\tau > 1 + \sigma$ **then**
        $D_{x,new} := D_{x,\bar{k}-1} + (D_{x,new} - D_{x,\bar{k}-1})\varrho$
        $x_{new} := x_{\bar{k}-1} + (x_{new} - x_{\bar{k}-1})\varrho$
      else if $\tau < 1 - \sigma$ **then**
        $D_{x,new} := D_{x,\bar{k}-1} + (D_{x,new} - D_{x,\bar{k}-1})\upsilon$
        $x_{new} := x_{\bar{k}-1} + (x_{new} - x_{\bar{k}-1})\upsilon$
      end if
      compute τ, $\bar{\boldsymbol{\lambda}}(p_l, C_s)$ according to (4.74), with $D_{x,new}$, and x_{new}.
    end while
    $\mathcal{P}_{\boldsymbol{\lambda}pas} := \bigcup \bar{\boldsymbol{\lambda}}(p_l, C_s)$
  end for
end for

4.5.2. Selection of a Homotopic Target Trajectory

In the online selection, a homotopic target trajectory $\bar{\boldsymbol{\lambda}} \in \mathcal{P}_{\boldsymbol{\lambda}pas}$ is selected from the set of mapped *passing points*. The controllers K_{k^*+j}, $j \in \mathcal{J}_{k^*}$, offline computed from Sec. 4.4, drive the system from x_{k^*} to the homotopic target trajectory $\hat{x}(\bar{\boldsymbol{\lambda}})$. Thereby, the trajectory passes the obstacle through the *passing point* p_l to the final state x_f. With respect to the $\boldsymbol{\lambda}$-space, this equals the transition from the actually executed trajectory at time k^*, encoded by $\boldsymbol{\lambda}_{k^*}$, to the homotopic target trajectory $\bar{\boldsymbol{\lambda}}$. All resulting trajectories for each $\bar{\boldsymbol{\lambda}} \in \mathcal{P}_{\boldsymbol{\lambda}pas}$ are evaluated according to their costs, and the most appropriate one, being free of collision, is chosen. To formulate the costs depending on the homotopy parameter, i.e. as $J(\bar{\boldsymbol{\lambda}})$, the values x_{k^*+j} and u_{k^*+j} in (4.16) are replaced by $x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j})$ and $u_{new,k^*+j}(\boldsymbol{\lambda}_{k^*+j})$ according to (4.6) and (4.19). With the closed-loop system dynamics (4.26c), the transformed costs result to:

$$J(\bar{\boldsymbol{\lambda}}) = \sum_{j=0}^{N-1-k^*} \|x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) - x_f\|_Q^2 + \|u_{new,k^*+j}(\boldsymbol{\lambda}_{k^*+j}) - u_f\|_R^2 \tag{4.81a}$$

with:

$$x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) = x^0_{k^*+j} + D_{x,k^*+j}\boldsymbol{\lambda}_{k^*+j} \tag{4.81b}$$

$$u_{new,k^*+j}(\boldsymbol{\lambda}_{k^*+j}) = u_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) + u_{add,k^*+j} \tag{4.81c}$$

$$u_{add,k^*+j} = -K_{k^*+j}(\boldsymbol{\lambda}_{k^*+j} - \bar{\boldsymbol{\lambda}}) \tag{4.81d}$$

$$\boldsymbol{\lambda}_{k^*+j+1} = (I - D_{x,k^*+j+1}{}^{-1} B K_{k^*+j})(\boldsymbol{\lambda}_{k^*+j} - \bar{\boldsymbol{\lambda}}) + \bar{\boldsymbol{\lambda}} \tag{4.81e}$$

$$j \in \mathcal{J}_{k^*} \tag{4.81f}$$

$$\bar{\boldsymbol{\lambda}} \in \mathcal{P}_{\boldsymbol{\lambda}pas}. \tag{4.81g}$$

Definition 4.9. *Let $\boldsymbol{\Lambda}$ denote the set of passing points $\mathcal{P}_{\boldsymbol{\lambda}pas}$ in ascending order of the costs $J(\bar{\boldsymbol{\lambda}})$. An element of $\boldsymbol{\Lambda}$ is referred to by $\Lambda(i)$. The first element $\Lambda(1)$ has the lowest costs.*

The developed method is also able to adapt its solution trajectory to changes in the obstacle position by replanning. As shown in (4.81), the costs for the circumventing trajectories through any passing point $\bar{\boldsymbol{\lambda}} \in \mathcal{P}_{\boldsymbol{\lambda}pas}$ are determined. But when the obstacle moves, the situation can occur that a homotopic trajectory, going directly to the final state, provides lower costs and is collision-free compared to any trajectory that goes through a passing point. Thus, the algorithmic procedure is structured in such a way that the optimal homotopic target trajectory $\bar{\boldsymbol{\lambda}}^* \in [0,1]^{n_c}$ is determined first. In case of a post-processed collision check, circumventing target trajectories are determined according to the cost evaluation (4.81). The optimization problem for the best homotopic target trajectory $\bar{\boldsymbol{\lambda}}^* \in [0,1]^{n_c}$ is:

$$\min_{\bar{\boldsymbol{\lambda}}} \quad J(\bar{\boldsymbol{\lambda}}) \tag{4.82a}$$

$$\text{s.t.} \quad (4.81b),\ (4.81c),\ (4.81d),\ (4.81e),\ (4.81f) \tag{4.82b}$$

$$\bar{\boldsymbol{\lambda}} \in [0,1]^{n_c},\ \sum \bar{\boldsymbol{\lambda}} \leq 1. \tag{4.82c}$$

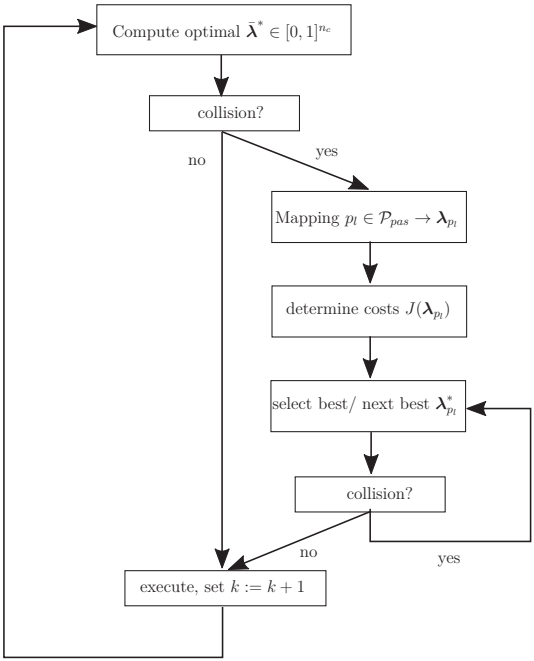


Figure 4.7.: Flowchart of the online trajectory selection.

A chart describing the flow of the algorithmic procedure is shown in Fig. 4.7.

The algorithmic procedure is shown in Alg. 4.2 and starts by executing the optimal *base trajectory* $\hat{x}^0$. If an obstacle is detected at time k^*, first the optimal homotopic target trajectory $\bar{\boldsymbol{\lambda}}^*$ is determined according to (4.82), ***(line 4-6)***. If the resulting trajectory is free of collision, ***(line 7)***, the circumvention procedure will not be executed. The system then executes one time step, increments the actual point of time and starts from the beginning. If a collision with an obstacle is detected, the circumvention procedure is initiated. Therefore, the *passing points* $p_l \in \mathcal{P}_{pas,k^*}$ of the currently detected polytope at time k^* are mapped into the homotopy space according to Alg. 4.1 and ordered according to increasing costs, resulting in the set $\boldsymbol{\Lambda}$, ***(line 8-9)***. Now, the desired homotopy value is set to $\bar{\boldsymbol{\lambda}} := \Lambda(1)$, and the state sequence $x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j})$ is computed according to (4.6) and (4.26c), ***(line 11-13)***. It is subsequently checked against collisions with the obstacle $\mathcal{P}_{x,k^*}$ ***(line 14)***. If this trajectory is free of collision, the algorithm terminates directly with the desired homotopy value $\bar{\boldsymbol{\lambda}} = \Lambda(1)$. If this is not the case, the next element $\Lambda(2)$ is tested. Hence, a new trajectory passing the obstacle along the *passing point* $\Lambda(2)$ (with higher costs) is computed and checked against collision. Now again, the

system executes its trajectory for one time step, increments the time index k^* ***(line 23-24)***, and the algorithm starts at the beginning.

If no feasible $\bar{\boldsymbol{\lambda}}$ can be determined, i.e. no *passing point* exists to pass $\mathcal{P}_{x,k^*}$ without collision, Alg. 4.2 terminates with this result. Consequently, an emergency stopping routine has to be started, which stops the system in a save position.

Algorithm 4.2.: Online Selection of Homotopic Target Trajectory

1: **Start:** execute $\hat{x}^0$
2: **Given:** k^*, $\boldsymbol{\lambda}_{k^*}$, $\mathcal{P}_{x,k^*}$, $\mathcal{P}_{pas,k^*}$
3: **while** $k^* \leq N-1$ **do**
4: compute the best minimizing constant homotopy value $\bar{\boldsymbol{\lambda}}^*$ by (4.81)
5: compute for all $j \in \mathcal{J}$ the states $x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j})$ according to (4.6)
6: and (4.26c) by setting $\bar{\boldsymbol{\lambda}} := \bar{\boldsymbol{\lambda}}^*$
7: **if** $\exists j \in \mathcal{J}_{k^*} : x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) \in \mathcal{P}_{x,k^*}$ **then**
8: map all *passing points* $p_l \in \mathcal{P}_{pas,k^*}$ into the $\boldsymbol{\lambda}$-space by `MapPas`
9: compute the set $\boldsymbol{\Lambda}$ ordered according to $J(\bar{\boldsymbol{\lambda}})$
10: **for** $i \in \{1, \ldots, |\boldsymbol{\Lambda}|\}$ **do**
11: compute for all $j \in \mathcal{J}$ the states
12: $x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j})$ according to (4.6) and (4.26c)
13: with homotopy value $\bar{\boldsymbol{\lambda}} := \Lambda(i)$.
14: **if** $\exists j \in \mathcal{J}_{k^*} : x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) \in \mathcal{P}_{x,k^*}$ **then**
15: trajectory from $\boldsymbol{\lambda}_{k^*+j}$ to $\bar{\boldsymbol{\lambda}}$ is not feasible
16: **else**
17: trajectory from $\boldsymbol{\lambda}_{k^*+j}$ to $\bar{\boldsymbol{\lambda}}$ is an optimized
18: feasible trajectory with $\bar{\boldsymbol{\lambda}} = \Lambda(i)$.
19: **break**
20: **end if**
21: **end for**
22: **end if**
23: execute the system dynamics with u_{new}
24: set $k^* := k^* + 1$
25: **end while**

4.6. Simulation Results

In order to illustrate the proposed procedure, the point-to-point control of a generic system with $n_x = 3$ and $n_u = 3$ is considered. The linear discrete-time system is given by (3.2). For reason of comparability between different trajectories, the matrices of the system are chosen with diagonal dominant entries such that a straight trajectory towards the target state is optimal:

$$A = 1e^{-3} \cdot \begin{pmatrix} 953 & 24 & 12 \\ 24 & 911 & 9.4 \\ 12 & 9.4 & 965 \end{pmatrix}, B = 1e^{-4} \cdot \begin{pmatrix} 967 & -7 & 40 \\ -7 & 1026 & -46 \\ 40 & -46 & 1057 \end{pmatrix}. \tag{4.83}$$

The weighting matrices of the performance function (4.16) are chosen to:

$$Q = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 10 & & 0 \\ & \ddots & \\ 0 & & 10 \end{pmatrix}, \tag{4.84}$$

i.e., the inputs are more penalized. This prevents the system from solutions like jumping in one step to the final state. For the controller synthesis, the weighting matrices Q_C, and R_C are selected such that the system is allowed to execute the transition to a homotopic target trajectory fast. That means, Q_C is stronger weighted than R_C:

$$Q_C = \begin{pmatrix} 50 & & 0 \\ & \ddots & \\ 0 & & 50 \end{pmatrix}, \quad R_C = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}. \tag{4.85}$$

The finite time domain is selected to $N = 60$, the initial state is located in the origin:

$$x_0 = x_s = [0, 0, 0]^T,$$

and the final state, and input, at the end time N are:

$$x_N = x_f = [5, 5, 5]^T, \quad u_N = u_f = [0.5, 2.7, 0.7]^T.$$

With $n_c = 3$, the set of offline computed trajectories is chosen to: $\mathcal{X} = \{(\hat{x}^0, \hat{u}^0), (\hat{x}^1, \hat{u}^1), (\hat{x}^2, \hat{u}^2), (\hat{x}^3, \hat{u}^3)\}$. The optimal trajectory $(\hat{x}^0,\hat{u}^0)$ is determined by solving (4.17a) without the collision avoidance constraints (4.17c). In the following figures, this trajectory is colored magenta. The backtracking parameters in Alg. 4.1 are set to: $\beta = 1e^{-5}$, $\varrho = 0.9$, $\upsilon = 1.1$, $\sigma = 0.05$. The other n_c base trajectories (black) of $\mathcal{X}$ are specified to obtain the shown area for obstacle avoidance.

Circumvention of an Obstacle detected during Execution

The considered scenario is to drive the system from the initial state x_0 to the final $x_N = x_f$, while avoiding the polytopic obstacle $\mathcal{P}_{x,k^*}$, which appears at $k^* = 10$, and then remains static until the end time. The polytopic obstacle is chosen as shown in Fig. 4.8. Obviously $\hat{x}^0$ intersects with $\mathcal{P}_{x,k^*}$, while this is not true for the three black base trajectories. The scenario shows only the solution for one execution of Alg. 4.2, hence that for the detection time k^*. The green trajectory part represents the executed trajectory until the detection time k^*. Since the actual trajectory can not be followed further (magenta), the homotopy method determines a collision-free homotopic trajectory with low costs. The blue trajectory in Fig. 4.8, passes the obstacle free of collision along its lower vertex. In this case the total number of *passing points* is set to $|\mathcal{P}_{pas}| = 8$, which correspond to the vertices of the polytope $\mathcal{P}_{x,k^*}$. For the red trajectory an arbitrarily chosen number of passing points $|\mathcal{P}_{pas}| = 280$ is distributed along the polytope edges. One can see the red trajectory passing the obstacle at its lower edge. By increasing the number of *passing points*, the chance of finding a better circumventing trajectory increases.

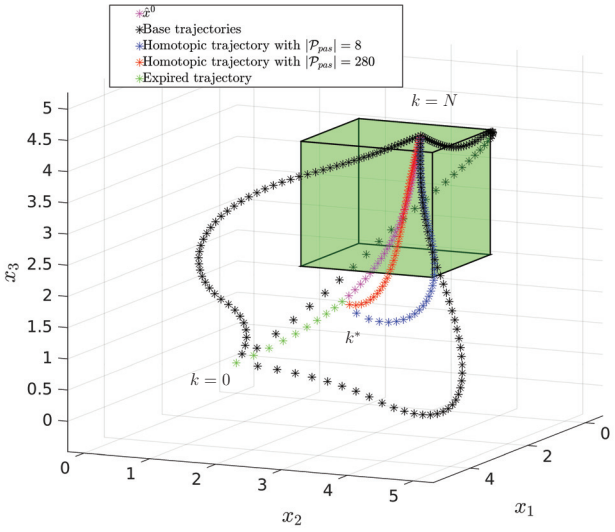


Figure 4.8.: State trajectories with optimal trajectory $\hat{x}^0$ (magenta), base trajectories $\hat{x}^1$, $\hat{x}^2$, $\hat{x}^3$ (black), a part of the optimal trajectory (green) up to $k^* = 10$, the trajectory obtained by the homotopy approach: (blue) 8 *passing points* only at polytope vertices, and (red) 280 *passing points*. The polytope $\mathcal{P}_{x,k^*}$ is the green area.

Table 4.1.: Comparison of Computation Times and Costs

Method	Comp. Time	Costs
Homotopy method $\lvert\mathcal{P}_{pas}\rvert = 8$	4.3 [ms]	1584
Homotopy method $\lvert\mathcal{P}_{pas}\rvert = 280$	110 [ms]	1375
MIQP	1370 [ms]	1339

While the costs for the red trajectory, with $|\mathcal{P}_{pas}| = 280$, is lower compared to the blue one, the computation time increases from 4.3 milliseconds [ms] to 110 [ms] (see Table 4.1), using a Matlab implementation on a standard PC. Compared to an MIQP approach, the homotopy method shows significant advantages in computation times. Here, the MIQP approach provides the lowest costs, 1339, but suffers from significantly higher computation times, 1370 [ms], even compared to the case with many *passing points*. The MIQP-based approach is implemented in Matlab, with embedded solution of the optimization problems by the CPLEX-tool used on the same PC. The high computation time of the MIQP approach results from the fact that for all time steps starting from k^* (i.e. for 50 steps) the collision avoidance constraints have to be formulated using binary variables, leading to a large number of algebraic constraints involving these variables. Furthermore, it is observed that if the input variables are not constrained in the MIQP-based approach the initialization of the MIQP-Solver takes longer, which also contributes to the higher computation times.

Adapting Circumvention to a Moving Obstacle

Since it is assumed in this chapter that prediction information of the obstacle is available, the developed procedure has to replan its trajectory if a change in the obstacle position occurs at a certain point of time. Based on the above-mentioned system dynamics, the start and goal states, and the given set of *base trajectories*, the following example shows a complete run of Alg. 4.2, starting from point of time $k = 5$, with detection of the obstacle $\mathcal{P}_{x,k}$ at each time. Furthermore, the obstacle moves in this example from an initial to a final position. The movement of the obstacle starts at $k = 5$ and is completed at time $k = 10$, where the obstacle remains for the rest time until $k = 60$. The initial obstacle position is shown in Fig. 4.9 by the white dashed polytope, and the actual polytope positions by the green polytope. The green part of trajectory is the initial execution of the system along the optimal trajectory $\hat{x}^0$ until the detection time of $k = 5$. The red and blue trajectories show the planned trajectories that circumvent the obstacle. The red trajectory represents the solution of the algorithm based on $|\mathcal{P}_{pas}| = 280$ *passing points* distributed on the obstacle edges. The blue trajectory is the solution with $|\mathcal{P}_{pas}| = 8$ *passing points* representing only the vertices of the polytope. The trajectory parts marked by circles show the planned future states based on the

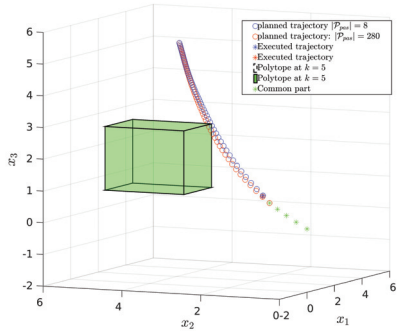
Table 4.2.: Comparison of Costs for a Complete Simulation

Method	Costs
Homotopy method $\lvert\mathcal{P}_{pas}\rvert = 8$	1150
Homotopy method $\lvert\mathcal{P}_{pas}\rvert = 280$	1507

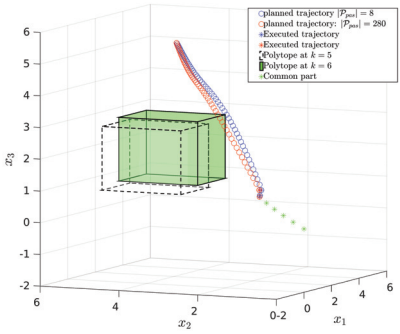
obstacle information from the current point of time. The trajectory parts marked by asterisks show the executed trajectories for both cases. Comparing the two results (the red and blue trajectories), it can be observed that the obstacle is circumvented in different ways, each depending on the considered number of *passing points* (and optimal for these numbers). The red trajectory determines lower costs when passing the obstacle from above. The blue trajectory adapts at times $k = \{6, 8, 9\}$ its best *passing point* during the motion of the obstacle, and finally passes the obstacle along the lower left vertex of the polytope. In Fig. 4.9e and Fig. 4.9f, it can be seen that the rear part of the trajectories changes although the obstacle has reached its end position at $k = 10$. This is because Alg. 4.2 starts its computation always with determining the best homotopic trajectory $\bar{\boldsymbol{\lambda}}^* \in [0, 1]^{n_c}$ and only in case of collisions, steers through the *passing points*. This situation occurs shortly after time step $k = 20$ (see Fig. 4.9e), when both trajectories pass the obstacle, so that the remaining space to the goal state $x_f = [5,\ 5,\ 5]^T$ is free. For the sake of clarity, the *base trajectories* are not illustrated in Fig. 4.9, but match exactly to those in Fig. 4.8.

The computation times for each sampling time in the presented obstacle avoidance scenario is shown in Fig. 4.10. As expected, the computation times are much lower when using less *passing points*, see Fig. 4.10a. They are mainly in the order of 5 [ms], compared to approximately 100 [ms] in the case of 280 *passing points* (see Fig. 4.10b). After passing the obstacle, the remaining computation times originate from solving the optimization problem (4.82), which is then the only step that has to be carried out in Alg. 4.2. Both figures show that the computation times have the tendency to decrease, when time progresses. This is obviously due to the fact, that the mapping algorithm Alg. 4.1 terminates faster, if less time steps have to be considered.

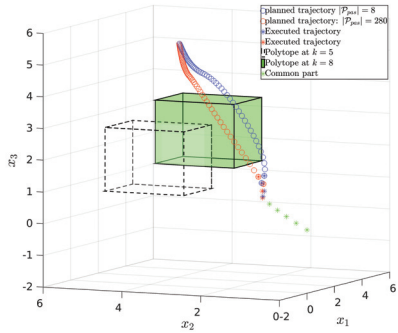
For both cases with different numbers of *passing points*, optimal collision-free trajectories can be obtained. It can be compared, that the case with many *passing points* benefits from lower costs 1150 compared to the other case with 1507. On the other hand, the computation times with many *passing points* are much larger.



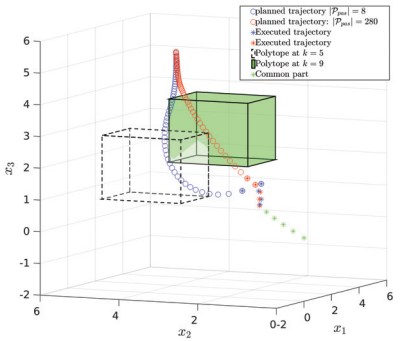
(a) Circumvention at $k = 5$.



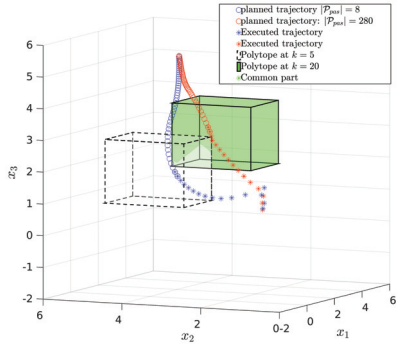
(b) Circumvention at $k = 6$.



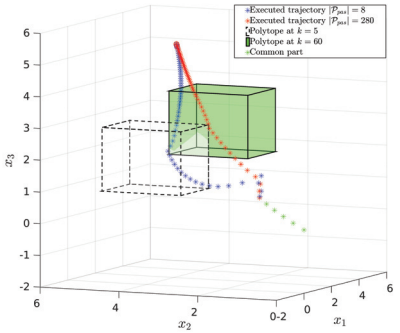
(c) Circumvention at $k = 8$.



(d) Circumvention at $k = 9$.

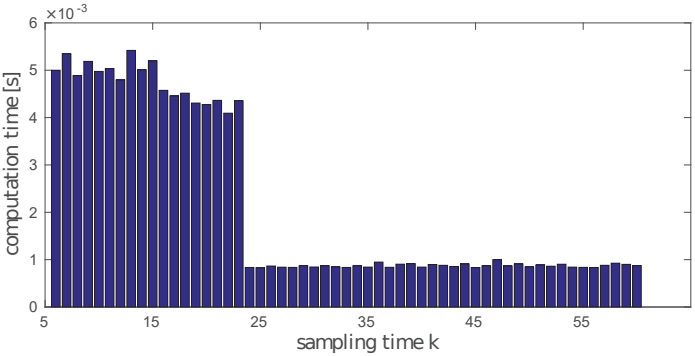


(e) Circumvention at $k = 20$.

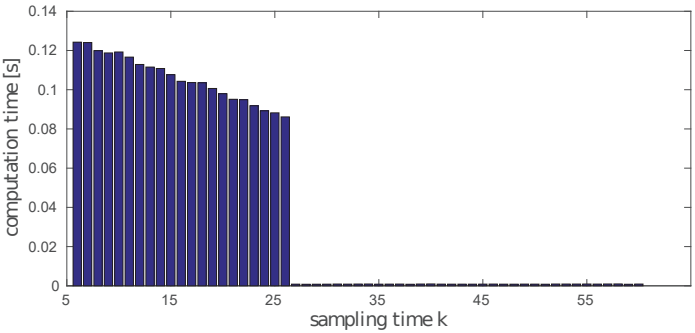


(f) Circumvention at $k = 60$.

Figure 4.9.: Comparison of the computation times for each sampling time k, with different numbers of *passing points*: blue $|\mathcal{P}_{pas}| = 8$, red $|\mathcal{P}_{pas}| = 280$.



(a) 8 *passing points*.



(b) 280 *passing points*.

Figure 4.10.: Comparison of the computation times for each sampling time k with different numbers of *passing points* $|\mathcal{P}_{pas}|$.

4.7. Discussion

This chapter has introduced the optimal point-to-point control problem of linear discrete-time systems in a non-convex state space, in particular for polytopic obstacles. The solution strategy is based on using homotopy properties, which enables to span a space of homotopic trajectories by means of a few offline predetermined *base trajectories*. In this space, the circumvention of the obstacle has taken place. A precise approach how these *base trajectories* have to be selected is not discussed in this chapter. A suitable choice depends on possible obstacle positions that can occur and the conclusion in which part of the state space an evasion should be enabled. Thus, this procedure can be understood as a semi-autonomous method. This chapter has comprehensively explained how the number of chosen *base trajectories* affects the controller synthesis. The time-varying controllers are required to realize transitions between homotopic trajectories. Therefore the system dynamics has been transformed into the homotopy space, resulting in a linear time-varying system, for which the three possible cases (uniquely solvable, underdetermined and overdetermined) have been discussed. A controller synthesis based on LMI formulations has been presented. The overdetermined case ($n_x > n_c$) requires additional constraints in the formulation. This is because the controllers, determined in the lower dimensional homotopy space, cannot be uniquely determined without further constraints. This case will be revisited in the subsequent chapter. It is especially important when considering a high dimensional state space, but a non full-dimensional obstacle in a subspace, such that it will be possible to use just a few *base trajectories* for circumvention. Furthermore, the transition behavior caused by the controllers did not underlie input constraints in this chapter. The use of the input signals has been considered by the weights of the cost function in the controller synthesis. The requirement of strictly satisfying input constraints, as it is the case for many real applications, is treated in the next chapter.

For the identification of possible locations for obstacle circumvention, *passing points* have been introduced. These points were mapped into the homotopy space such that a homotopy value can be allocated to each *passing point*. Therefore the algorithm `MapPas` has been presented. The algorithm handles the problem that one *passing point* can be mapped into the homotopy space multiple times. The multiple mapping problem is due to the fact that a point in the state space can be described in the homotopy space at different points of time. To prevent the number of mapped points from becoming too large, `MapPas` partitioned the homotopy space and identifies upper and lower time bounds between which a *passing point* is located. By doing this, a *passing point* can be uniquely allocated to a point of time and therefore also to a single homotopic state. However, it must be pointed out, that the number of sub-spaces is combinatorial, meaning that a high number of *base trajectories* leads to a high number of partitions that have to be considered in `MapPas`. During the online procedure, `MapPas` is embedded in an algorithm that realizes the circumvention in cases of an upcoming collision. This

has been done by selecting a suitable *passing point* such that the corresponding homotopic target trajectory circumvents the obstacle. It has been shown that the number of possible *passing points*, distributed on the obstacle surface, effects the solution of the procedure. The higher the number, the better the trajectory becomes with respect to its costs, since the algorithm can choose from a higher number of possible homotopic target trajectories. On the other side, the computation time has raised. However, it is recommended to keep the number small, because the cost decrease does not scales immediately with the number of *passing points*. It should be mentioned that the procedure can also terminate with no result, e.g., if the obstacle is detected too late, such that the controllers can not complete the transition. If this occurs, the system has to stop immediately.

Furthermore, the online procedure which has been introduced is able to adapt its determined homotopic target trajectory to a changing obstacle position. In total, the use of homotopy properties for obstacle avoidance is shown to be very powerful. Although the solution space for suitable circumventing trajectories is smaller compared to an MIQP problem formulation, the variety of trajectories (predetermined by the *base trajectories*) is sufficient in many cases. Despite of slight losses in performance, the resulting trajectories have shown a significant advantage in computation times, compared to a MIQP solution.

As mentioned above, since many dynamic systems can have many states but an obstacle that may be described in a lower dimensional subspace, also only a few *base trajectories* are needed to span a space for circumvention. This circumstance and the additional consideration of input constraints in the controller synthesis are part of the next chapter.

5. Extension to Input Constraints

The last chapter has shown the benefits of using homotopy properties for obstacle avoidance problems. It has been shown, that the use of homotopies is a reasonable alternative for managing these time demanding problems (when solved by MIQP for many time steps), especially when the system requires solutions to be available quickly. The advantage of this method is that the trajectories do not have to be determined from scratch like in MIQP, rather a homotopic target trajectory is "selected" and the linear time-varying controllers ensure the transition to it. By this separation of the problem into an offline and online problem, the efficiency of the developed method was shown. Nevertheless, two essential points were not addressed in the previous chapter. Namely the issue of considering input constraints in the controller synthesis, and the case of an overdetermined system ($n_x > n_c$), where the system state is higher than the number of *base trajectories*. The satisfaction of input constraints is important for real world applications, e.g. a robotic manipulator can not transition infinitely fast to a collision-free homotopic target trajectory because of actuator limitations. Therefore this chapter investigates how input limitations can be included in the controller synthesis. Within this framework, a controller synthesis based on LMI techniques and semi-definite programming is presented. The purpose is to complete trajectory changes as quickly as possible. To realize this, the constrained input resources have to be used optimally. This chapter investigates the effect of the constrained additional input signal $u_{add,k}$ in the controller synthesis. A certain limitation already exists because of the existing *base trajectories*. Thus, $u_{add,k}$ impacts the speed of transition between homotopic trajectories. The controllers are synthesized in a way that stability is guaranteed for the complete region which is spanned by the *base trajectories*, hence for all steps from $\boldsymbol{\lambda} \in (0,1)^{n_c}$ to a homotopic target trajectory $\bar{\boldsymbol{\lambda}} \in (0,1)^{n_c}$. As in the previous chapter, this chapter refers to linear discrete-time systems and its corresponding homotopic differential equations introduced in Chapter 4. Furthermore, it is noticed that dealing with the property that there are less *base trajectories* than system states is an important aspect. Obstacles may exist only in a subspace of the system dimension, such that a few *base trajectories* are sufficient for circumvention. This circumstance results in an unclear determination of the controllers, for which the purpose of this chapter is to derive conditions on the controller synthesis to overcome this problem.

This chapter is structured such, that first the problem is stated in Sec. 5.1, and then the offline controller synthesis is given in Sec. 5.2. For illustration of the results, a numerical example is given in Sec. 5.3. The main ideas of this chapter were already reported in [60].

5.1. Problem Description

The problem considered here is mainly based on the problem described in Sec. 4.2, where the task is to drive a system from an initial to a final state, while avoiding collisions with an obstacle. As already described in Sec. 4.5, the online control strategy is identical in this chapter, so that the reader is referred to that section. This chapter focuses on the offline part, hence the controller synthesis. Here, the following two problems are discussed:

1. Since the homotopy input $u((\lambda)_k)$ is completely known by the base trajectories, the amount of the additional input signal is constraint here already in the controller synthesis $|u_{add,k}| < |u_{max}|$
2. Handling the problem that x_{k+1} is not a homotopic state $x_{k+1}(\boldsymbol{\lambda}_k)$, $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_k)$, which occurs by determining the controllers in the lower dimensional $\boldsymbol{\lambda}$-space, and subsequently using them in the higher state space by applying the input $u_{new,k}(\boldsymbol{\lambda}_k)$ (remember $n_x > n_c$), see (4.19)

The following definition first gives the permissible regions of $\boldsymbol{\lambda}_k$, $\bar{\boldsymbol{\lambda}}$, and $\delta\boldsymbol{\lambda}_k$.

Definition 5.1. *Consider that each element $\lambda_k^i \in \boldsymbol{\lambda}_k$ is in the range $\lambda_k^i \in [0, 1]$, as given in Def. 4.2. Then the set:*

$$\mathcal{G} := \{\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_{2^{n_c}}\}, \tag{5.1}$$

with $|\mathcal{G}| = 2^{n_c}$ elements $\boldsymbol{\gamma}_i \in \mathbb{R}^{n_c}$ contains all vertices of the permissible region $\boldsymbol{\lambda}_k$. The number of vertices is given by 2^{n_c}. Hence, the set of permissible $\boldsymbol{\lambda}_k$ is given by the convex hull of the finite set $\mathcal{G}$:

$$\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{G}\}. \tag{5.2}$$

The set $\mathcal{D}$, with elements $d_w \in \mathbb{R}^{n_c}$ represents the finite set of vertices, obtained from $\delta\boldsymbol{\lambda}_k = \boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}$:

$$\mathcal{D} := \{d_w | d_w = \boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}, \boldsymbol{\lambda}_k \in \mathcal{G}, \bar{\boldsymbol{\lambda}} \in \mathcal{G}\}, \tag{5.3}$$

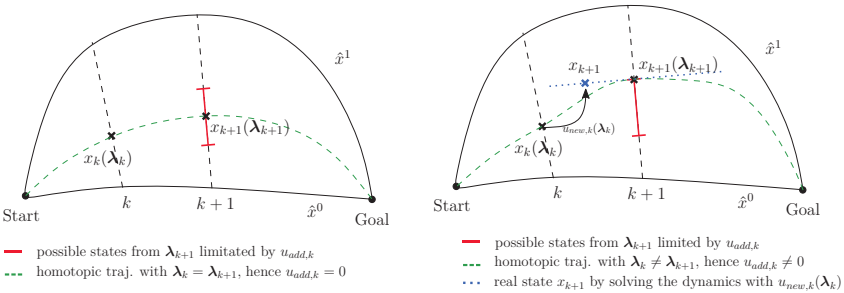
so that:

$$\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}. \tag{5.4}$$

The cardinality of $\mathcal{D}$ is given by $|\mathcal{D}|$, and a vertex d_w is indexed by $w \in \{1, ..., |\mathcal{D}|\}$.

Problem 5.1. *With respect to the above given input constraints, the control law of $u_{add,k}$ is:*

$$u_{add,k} := -K_k \delta\boldsymbol{\lambda}_k, \tag{5.5}$$



(a) Shows the region in which the homotopic trajectory can be changed according to the input constraints.

(b) Ambiguity in the controller synthesis leads to states x_{k+1} that do not satisfy $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1})$.

Figure 5.1.: Effects of input constraints and the controller synthesis.

which is assumed to be element-wise norm bounded:

$$||K_{k,s}\delta\boldsymbol{\lambda}_k|| \leq ||u_{max,s}||, \ \forall s \in \{1, ..., n_u\}, \ \forall k \in \mathcal{J}_{\mathcal{N}}, \ \delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\} \tag{5.6}$$

with $K_{k,s} \in \mathbb{R}^{1\times n_c}$ denoting the s-th row of the controller matrix $K_k \in \mathbb{R}^{n_u \times n_c}$ at time k, and $u_{max,s} \in \mathbb{R}$ the s-th element of the vector $u_{max} \in \mathbb{R}^{n_u}$. The input constraints have to hold for all times k, all elements s, and all possible values $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}$.

Since it is known by (4.19), that the applied input $u_{new,k}(\boldsymbol{\lambda}_k)$ on the dynamics (3.2) consists of the homotopy input $u_k(\boldsymbol{\lambda}_k)$, and the additional input $u_{add,k}$:

$$u_{new,k}(\boldsymbol{\lambda}_k) = u_k(\boldsymbol{\lambda}_k) + u_{add,k},$$

a value $u_{add,k} = 0$ leads the system to stay on its trajectory, see Fig, 5.1a (green trajectory). A value $u_{add,k} \neq 0$ with controllers K_k now forces the system to transition to a new homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$, and therefore to a new homotopic target trajectory (see red area in Fig, 5.1a). To what extent this change may be is limited by the constraints on $u_{add,k}$, and therefore also on the controller matrices K_k as shown in (5.6). The problem which now occurs is that if the system is considered to be overdetermined ($n_x > n_c$), the synthesized controllers K_k bring the homotopic vector from $\boldsymbol{\lambda}_k$ to the target $\bar{\boldsymbol{\lambda}}$, but the state x_{k+1} is not a homotopic state: $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1})$. This effect results if no additional constraints on the controller synthesis are made for overdetermined systems.

An outline of the problem is shown in Fig. 5.2. Starting from the left in Fig. 5.2, the accomplished controller synthesis provides the controllers $K_k \in \mathbb{R}^{n_u \times n_c}$ of the lower dimensional $\boldsymbol{\lambda}$-space. With the known value of $\boldsymbol{\lambda}_{k+1}$ from the closed loop dynamics (4.39), the homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ can be determined according to (4.6). This process is shown in the lower part of Fig. 5.2. In fact the propagation of

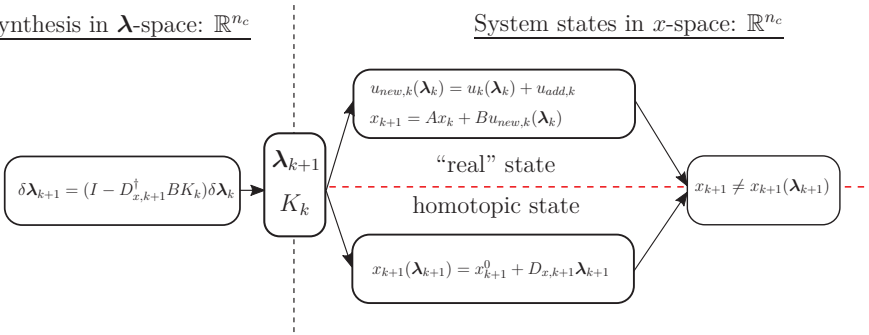


Figure 5.2.: Comparison between the computation of an homotopic state $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1})$, and the state x_{k+1} which is really executed.

the system dynamics (3.2) must be forced by an input signal $u_{new,k}(\boldsymbol{\lambda}_k)$. The upper part of Fig. 5.2 shows the "real" state x_{k+1}, resulting from $u_{new,k}(\boldsymbol{\lambda}_k)$ and therefore also from the controller matrix K_k. This is not equal to the homotopic state $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1})$, if no additional constraints handle this effect in the controller synthesis.

Problem 5.2. *Comparing the homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ with the "real" state x_{k+1} it can be observed that x_{k+1} is not equal to the solution which is projected back from the homotopy space:*

$$x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1}), \forall \boldsymbol{\lambda}_{k+1} \in \mathcal{G}, \tag{5.7}$$

when the controllers K_k, obtained from the synthesis in the $\boldsymbol{\lambda}$-space, are applied in the state space.

The effect of this can be explained based on Fig. 5.1b. When the system has to transition from a trajectory encoded by $\boldsymbol{\lambda}_k$ to one by $\boldsymbol{\lambda}_{k+1}$, the determined controller will not drive the state x_{k+1} towards the homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$. This is because in the overdetermined case, the mapping from the homotopy space to the state space is not unique. Nevertheless, from the perspective of the $\boldsymbol{\lambda}$-space, the state x_{k+1} and $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ have the same value $\boldsymbol{\lambda}_{k+1}$. Fig. 5.1b shows that the overdetermined homotopy system (4.39) yields the same $\boldsymbol{\lambda}_{k+1}$ for all states located on the blue dashed line.

The requirement $x_{k+1} = x_{k+1}(\boldsymbol{\lambda}_{k+1})$ is an important property for the online procedure of Sec. 4.5. If the states are no longer homotopic states, then a *passing point* can not be allocated to a homotopic state $\boldsymbol{\lambda}(p_l, C_s)$, and therefore not to a homotopic target trajectory. Thus, the task is to derive additional constraints for the controller synthesis to overcome this problem for overdetermined systems.

5.2. Offline Controller Synthesis

This section provides a solution of the named problems, the inclusion of input constraints into the synthesis and the ambiguity in the controller synthesis.

Inclusion of Input Constraints into the Synthesis

Let a quadratic Lyapunov function with matrix $P \in \mathbb{S}^{n_c}_{>0}$ be given by:

$$V_k = \delta\boldsymbol{\lambda}_k^T P \delta\boldsymbol{\lambda}_k. \tag{5.8}$$

The LTV closed-loop delta-system (4.39):

$$\delta\boldsymbol{\lambda}_{k+1} = (I - D^{\dagger}_{x,k+1} B K_k)\delta\boldsymbol{\lambda}_k,$$

is stable with given decreasing rate parametrized by the matrix ${Q_C}^{-1} \in \mathbb{S}^{n_c}_{>0}$, if the following condition holds for every $k \in \mathcal{J}_{\mathcal{N}}$ (see Def. 4.5 for Lyapunov stability):

$$V_{k+1} - V_k \leq -\delta\boldsymbol{\lambda}_k^T {Q_C}^{-1} \delta\boldsymbol{\lambda}_k. \tag{5.9}$$

The matrix ${Q_C}^{-1}$ has the form:

$${Q_C}^{-1} := \begin{pmatrix} \frac{1}{q_1} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{q_{n_c}} \end{pmatrix} \tag{5.10}$$

with elements $q_i \in \mathbb{R}$, $i \in \{1, ..., n_c\}$. The goal is to determine the controllers such that the decrease of the homotopy parameter $\delta\boldsymbol{\lambda}_k$ rapidly converges to zero, what means, that the transition from $\boldsymbol{\lambda}_k$ to the homotopic target trajectory $\bar{\boldsymbol{\lambda}}$ is fast. This is forced by decreasing the right side of (5.9) and can be achieved by increasing the trace of the matrix ${Q_C}^{-1}$. Hence, compared to Chapter 4, matrix Q_C is considered here as a matrix variable. Since one could think of reaching the highest decrease in the Lyapunov function simply by setting $\mathbf{tr}({Q_C}^{-1}) \to \infty$, it must be considered that a fast convergence of $\delta\boldsymbol{\lambda}$ forces high input signals. Since the additional input value $u_{add,k}$ is assumed to be element-wise norm bounded, see (5.6), the decrease in inequality (5.9) can not be done arbitrarily fast. Consequently, the maximization of $\mathbf{tr}({Q_C}^{-1})$ is upper bounded by the input constraints (5.6).

In order to include the input constraints, consider the ellipsoidal set:

$$\mathcal{E}(P) := \{\delta\boldsymbol{\lambda}_k \in \mathbb{R}^{n_c} |\ \delta{\boldsymbol{\lambda}_k}^T P \delta\boldsymbol{\lambda}_k \leq 1\}, \tag{5.11}$$

or equivalently:

$$\mathcal{E}(P) := \{P^{-\frac{1}{2}} z |\ \|z\| \leq 1\}, \tag{5.12}$$

with $P \in \mathbb{S}^{n_c}_{>0}$. It describes the level set of the quadratic Lyapunov function (5.8) by the shape matrix P. This function is centered at the origin of the LTV closed-loop delta-system (4.39).

Definition 5.2. *The ellipsoidal set $\mathcal{E}(P)$ denotes the region of attraction of the LTV closed-loop delta-system* (4.39).

As argued in [12], an LTV system, with its time-dependent, bounded matrices given by $D_{x,k}$ is Lyapunov-stable if the Lyapunov stability condition (5.9) is satisfied for all $k \in \mathcal{J}_{\mathcal{N}}$. The input-constrained controller synthesis task can be formulated as the following optimization problem for all $k \in \mathcal{J}_{\mathcal{N}}$ and the input components $s \in \{1, ..., n_u\}$:

$$\max_{P,\ K_k,\ Q_C,\ \delta\boldsymbol{\lambda}_k} \quad \mathbf{tr}({Q_C}^{-1}) + \mathbf{tr}(P) \tag{5.13a}$$

$$\text{s.t.} \quad V_{k+1} - V_k \leq -\delta{\boldsymbol{\lambda}_k}^T {Q_C}^{-1} \delta\boldsymbol{\lambda}_k,\ \forall k \in \mathcal{J}_{\mathcal{N}} \tag{5.13b}$$

$$||K_{k,s}\delta\boldsymbol{\lambda}_k|| \leq ||u_{max,s}||,\ \forall s \in \{1, ..., n_u\},\ \forall k \in \mathcal{J}_{\mathcal{N}},\ \forall \delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\} \tag{5.13c}$$

$$\delta{\boldsymbol{\lambda}_k}^T P \delta\boldsymbol{\lambda}_k \leq 1,\ \forall \delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\},\ \forall k \in \mathcal{J}_{\mathcal{N}}. \tag{5.13d}$$

The objective of the optimization problem consists of two parts. The first is to determine a fast converging controller, as enforced by the convergence condition on the Lyapunov function (5.13b) and the maximization of the cost term $\mathbf{tr}({Q_C}^{-1})$. However, the maximum descent of the Lyapunov function is bounded by the input constraints (5.13c). Meanwhile, the second part of the cost function maximizes $\mathbf{tr}(P)$. On the one side the matrix variable P is used to accomplish the descending conditions (5.13b), specified by the right side with matrix ${Q_C}^{-1}$. On the other side, the size of the ellipsoid $\mathcal{E}(P)$ (which equals the region of attraction) is minimized by maximizing the trace of the shape matrix $\mathbf{tr}(P)$. The size and shape of an ellipsoid can generally be minimized by different criteria, here the trace is considered which affects the semi-axes of the ellipsoid $\mathcal{E}(P)$. To make full use of the available input range, the ellipsoidal region of attraction $\mathcal{E}(P)$ has to be shrunk up to the permissible region $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}$, which is forced by the cost term $\mathbf{tr}(P)$ and the constraint (5.13d). Hence, constraint (5.13d) describes an important lower bound on the region of attraction. If one would shrink the ellipsoid below this size, stability and the input satisfaction is not guaranteed for the permissible region $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}$. By means of this, the input signal is fully utilized with respect to the permissible region of $\delta\boldsymbol{\lambda}_k$ and the requirement of a fast converging controller. If however, the ellipsoidal region of attraction would not be shrunken, the determined controllers would have to satisfy the input constraints for a larger region of $\delta\boldsymbol{\lambda}_k$ than the permissible region. This would lead to a slower transition behavior, since the closed-loop delta-system only has to start from its permissible region $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}$, but now with conservatively synthesized controllers for a larger region.

Due to the nonlinearity in (5.13b), (5.13c), and (5.13d), caused by the multiplication of $\delta\boldsymbol{\lambda}_k$ with other variables, the square root in (5.13c), and the difficulty of maximizing a convex cost function (5.13a), problem (5.13) is hard to solve in its original form. It can be shown, that the optimization problem can be re-formulated

as a convex semi-definite program, which yields optimal controllers K_k independently of the homotopy state $\delta\boldsymbol{\lambda}_k$. The following lemmata assist the re-formulation to a convex problem.

Lemma 5.1. *The non-convex constraint* (5.13b) *can be transformed into:*

$$(I - D^{\dagger}_{x_{k+1}} B K_k)^T P (I - D^{\dagger}_{x_{k+1}} B K_k) - P + {Q_C}^{-1} \leq 0, \tag{5.14}$$

with $P = Y^{-1}$ *and* $K_k = L_k Y^{-1}$. *If* $Y \in \mathbb{S}^{n_c}_{>0}$ *and* $L_k \in \mathbb{R}^{n_u \times n_c}$ *exist for all* $k \in \mathcal{J}_{\mathcal{N}}$, *then the following LMI holds:*

$$\begin{pmatrix} Y & (Y - D^{\dagger}_{x_{k+1}} B L_k)^T & Y^T \\ Y - D^{\dagger}_{x_{k+1}} B L_k & Y & 0 \\ Y & 0 & Q_C \end{pmatrix} \geq 0, \tag{5.15}$$

with 0 *denoting zero matrices of appropriate dimension.*

Proof. By inserting the Lyapunov functions (5.8) for k and $k+1$ into (5.13b):

$$\delta\boldsymbol{\lambda}_{k+1}^T P \delta\boldsymbol{\lambda}_{k+1} - \delta\boldsymbol{\lambda}_k^T P \delta\boldsymbol{\lambda}_k \leq -\delta{\boldsymbol{\lambda}_k}^T {Q_C}^{-1} \delta\boldsymbol{\lambda}_k, \tag{5.16}$$

and by substituting $\delta\boldsymbol{\lambda}_{k+1}$ according to the closed-loop delta-system (4.39), the inequality (5.16) becomes:

$$\delta{\boldsymbol{\lambda}_k}^T (I - D^{\dagger}_{x,k+1} B K_k)^T P (I - D^{\dagger}_{x,k+1} B K_k) \delta\boldsymbol{\lambda}_k - \delta\boldsymbol{\lambda}_k^T P \delta\boldsymbol{\lambda}_k \leq -\delta{\boldsymbol{\lambda}_k}^T {Q_C}^{-1} \delta\boldsymbol{\lambda}_k, \tag{5.17}$$

which yields the matrix inequality (5.14).

Now, using the substitution $Y = P^{-1}$ and multiplying (5.14) from the left and right with Y leads to:

$$(Y - D^{\dagger}_{x_{k+1}} B K_k Y)^T Y^{-1} (Y - D^{\dagger}_{x_{k+1}} B K_k Y) - Y + Y^T {Q_C}^{-1} Y \leq 0. \tag{5.18}$$

Applying the Schur complement to (5.18) yields:

$$\begin{pmatrix} Y & (Y - D^{\dagger}_{x_{k+1}} B K_k Y)^T & Y^T \\ Y - D^{\dagger}_{x_{k+1}} B K_k Y & Y & 0 \\ Y & 0 & Q_C \end{pmatrix} > 0, \tag{5.19}$$

for which the substitution $L_k = K_k Y$ completes the proof. □

Lemma 5.2. *The input constraint* (5.13c) *holds for all* $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}$, *if the following LMI holds:*

$$\begin{pmatrix} \|u_{max,s}\|^2 & L_{k,s} \\ {L_{k,s}}^T & Y \end{pmatrix} \geq 0. \tag{5.20}$$

The index $s \in \{1, ..., n_u\}$ *in* $L_{k,s} = K_{k,s} Y \in \mathbb{R}^{1 \times n_c}$ *denotes the s-th row of the corresponding matrix* L_k. *The LMI is valid for every component* $s \in \{1, ..., n_u\}$ *of the input vector* $u_{max,s}$ *and for all times* $k \in \mathcal{J}_{\mathcal{N}}$.

Proof. With the ellipsoidal set representation (5.12), the input constraint (5.13c) can be written as:

$$||K_{k,s}P^{-\frac{1}{2}}z|| \leq ||u_{max,s}||. \tag{5.21}$$

Replacing $K_{k,s} = L_{k,s}Y^{-1}$ yields:

$$||L_{k,s}Y^{-1}P^{-\frac{1}{2}}z|| \leq ||u_{max,s}||, \tag{5.22}$$

and with $P = Y^{-1}$, (5.22) becomes:

$$||L_{k,s}Y^{-1}(Y^{-1})^{-\frac{1}{2}}z|| \leq ||u_{max,s}||. \tag{5.23}$$

Squaring (5.23) leads to:

$$||L_{k,s}Y^{-\frac{1}{2}}z||^2 \leq ||u_{max,s}||^2 \tag{5.24a}$$
$$||L_{k,s}Y^{-\frac{1}{2}}||^2||z||^2 \leq ||u_{max,s}||^2 \tag{5.24b}$$
$$L_{k,s}Y^{-\frac{1}{2}}(Y^{-\frac{1}{2}})^T{L_{k,s}}^T \leq ||u_{max,s}||^2 \tag{5.24c}$$
$$L_{k,s}Y^{-1}{L_{k,s}}^T \leq ||u_{max,s}||^2 \tag{5.24d}$$
$$0 \leq ||u_{max,s}||^2 - L_{k,s}Y^{-1}{L_{k,s}}^T \tag{5.24e}$$

Taking note that the supremum of the left side of (5.24e) yields:

$$\sup_{||z||^2\leq 1} ||L_{k,s}Y^{-\frac{1}{2}}||^2||z||^2 = L_{k,s}Y^{-\frac{1}{2}}(Y^{-\frac{1}{2}})^T{L_{k,s}}^T, \tag{5.25}$$

the following steps can be executed:

$$L_{k,s}Y^{-\frac{1}{2}}(Y^{-\frac{1}{2}})^T{L_{k,s}}^T \leq ||u_{max,s}||^2 \tag{5.26a}$$
$$L_{k,s}Y^{-1}{L_{k,s}}^T \leq ||u_{max,s}||^2 \tag{5.26b}$$
$$0 \leq ||u_{max,s}||^2 - L_{k,s}Y^{-1}{L_{k,s}}^T, \tag{5.26c}$$

which is equal to (5.20) when applying the Schur complement. □

Lemma 5.3. *The ellipsoidal region of attraction* (5.11) *is shrunken, such that the permissible region* $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}$ *is still contained in the ellipsoid. Ensuring this requirement according to* (5.13d) *holds, if the following LMI holds:*

$$\begin{pmatrix} 1 & {d_w}^T \\ d_w & Y \end{pmatrix} \geq 0, \ \forall d_w \in \mathcal{D}. \tag{5.27}$$

Proof. The delta homotopy vector $\delta\boldsymbol{\lambda}_k$ can be determined from the convex linear combination of vertices $d_w \in \mathcal{D}$ by the following polytopic description:

$$\delta\boldsymbol{\lambda}_k = \sum_w \alpha_w d_w, \tag{5.28}$$

with weight α_w, and $\sum_w \alpha_w = 1$, $\alpha_w \geq 0$. Thus:

$$(\sum_w \alpha_w d_w)^T P (\sum_w \alpha_w d_w) \leq 1 \tag{5.29}$$

holds if all corners d_w of the polytopic description satisfy the inequality, hence leading to:

$$d_w^T P d_w \leq 1 \ \forall d_w \in \mathcal{D}. \tag{5.30}$$

Replacing $P = Y^{-1}$ and applying the Schur complement, (5.30) is converted into (5.27). It follows that $\mathcal{E}(P)$ is a stabilizing region of the saturated system for all permissible $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathcal{D}\}$. □

The convex reformulation of the original constraints (5.13b)-(5.13d) leads to the LMIs (5.15), (5.20) and (5.27), with variables L_k, Q_C and Y. On the other hand, the cost function (5.13a) still contains the matrix variables in its inverse form $P = Y^{-1}$ and ${Q_C}^{-1}$. Hence the cost function has to be suitably transformed to obtain a solution.

Lemma 5.4. *The objective* (5.13a) *with the reformulated constraints* (5.15) *and* (5.20), (5.27) *can be cast into a minimization problem by a conservative estimation, in which the objective function of the problem no longer contains the variables Y and Q_C in its inverse forms:*

$$\min_{Y,\ L_k,\ Q_C} \quad \mathbf{tr}(Q_C) + \mathbf{tr}(Y) \tag{5.31a}$$

$$s.t. \quad (5.15),\ (5.20),\ (5.27), \tag{5.31b}$$

$$\forall d_w \in \mathcal{D},\ \forall s \in \{1, ..., n_u\},\ \forall k \in \mathcal{J}_{\mathcal{N}}. \tag{5.31c}$$

Proof. Given a matrix $W \in \mathbb{R}^{n \times n}$ with $W = W^T$, $W > 0$, the inequality:

$$\mathbf{tr}(W^{-1}) \geq (\mathbf{tr}(W))^{-1} \tag{5.32}$$

holds. Since the cost function (5.13a) has to be maximized, a lower bound of the cost function (5.13a) can be given with (5.32) and $P = Y^{-1}$, according to the function $f_{lb} : \mathbb{S}^{n_c}_{>0} \times \mathbb{S}^{n_c}_{>0} \to \mathbb{R}$:

$$f_{lb}(Q_C, Y) := (\mathbf{tr}(Q_C))^{-1} + (\mathbf{tr}(Y))^{-1}. \tag{5.33}$$

As being addressed in [43], positive definite matrices (as Q_C and Y are) have positive diagonal elements and a positive value of the trace:

$$Q_C \in \mathbb{S}^{n_c}_{>0} \to \text{diagonal elements } q_i > 0,\ \mathbf{tr}(Q_C) > 0 \tag{5.34}$$

$$Y \in \mathbb{S}^{n_c}_{>0} \to \text{diagonal elements } y_i > 0,\ \mathbf{tr}(Y) > 0. \tag{5.35}$$

With these properties it can be shown that $f_{lb}(Q_C, Y)$ is strictly monotonically decreasing, by first rewriting (5.33) to:

$$f_{lb}(Q_C, Y) := (\mathbf{tr}(Q_C))^{-1} + (\mathbf{tr}(Y))^{-1} = \left(\sum_{i=i}^{n_c} q_i\right)^{-1} + \left(\sum_{i=1}^{n_c} y_i\right)^{-1}, \tag{5.36}$$

and determining its gradient $\nabla f_{lb} : \mathbb{S}^{n_c}_{>0} \times \mathbb{S}^{n_c}_{>0} \to \mathbb{R}^{n_c+n_c}$:

$$\nabla f_{lb}(Q_C, Y) = \begin{pmatrix} -(\sum_{i=1}^{n_c} q_i)^{-2} q_1 \\ \vdots \\ -(\sum_{i=1}^{n_c} q_i)^{-2} q_{n_c} \\ -(\sum_{i=1}^{n_c} y_i)^{-2} y_1 \\ \vdots \\ -(\sum_{i=1}^{n_c} y_i)^{-2} y_{n_c} \end{pmatrix} < 0, \tag{5.37}$$

so that one can see $\nabla f_{lb}(Q_C, Y) < 0$, because all $q_i > 0$ and $y_i > 0$, according to (5.34) and (5.35).

Since (5.33) is strictly monotonically decreasing, its inverse function:

$$f_{lb}(Q_C, Y)^{-1} = \frac{1}{(\mathbf{tr}(Q_C))^{-1} + (\mathbf{tr}(Y))^{-1}}, \tag{5.38}$$

is strictly monotonically increasing. Now, an upper bound $f_{ub}(Q_C, Y)$ on the cost function $f_{lb}(Q_C, Y)^{-1}$ is obtained by applying the triangle inequality $\frac{1}{a+b} \leq \frac{1}{a} + \frac{1}{b}$ (for $a, b > 0$) to (5.38), yielding:

$$f_{lb}(Q_C, Y)^{-1} \leq f_{ub}(Q_C, Y) \tag{5.39a}$$

$$\frac{1}{(\mathbf{tr}(Q_C))^{-1} + (\mathbf{tr}(Y))^{-1}} \leq \frac{1}{(\mathbf{tr}(Q_C))^{-1}} + \frac{1}{(\mathbf{tr}(Y))^{-1}} \tag{5.39b}$$

$$f_{ub}(Q_C, Y) := \frac{1}{(\mathbf{tr}(Q_C))^{-1}} + \frac{1}{(\mathbf{tr}(Y))^{-1}}, \tag{5.39c}$$

or respectively:

$$f_{ub}(Q_C, Y) = \mathbf{tr}(Q_C) + \mathbf{tr}(Y), \tag{5.40}$$

It follows that the original maximization of a monotonically decreasing function can be cast into a conservative estimated minimization problem of a strictly monotonically increasing function:

$$\min_{Y, L_k, Q_C} \quad f_{lb}(Q_C, Y)^{-1} \tag{5.41}$$

$$\text{s.t.} \quad (5.15),\ (5.20),\ (5.27) \tag{5.42}$$

$$\forall d_w \in \mathcal{D},\ \forall s \in \{1, ..., n_u\},\ \forall k \in \mathcal{J}_{\mathcal{N}}. \tag{5.43}$$

□

Ambiguity in the Controller Synthesis

As already mentioned in the problem description, the synthesized controllers K_k can generate states x_{k+1} which are not homotopic states $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1})$. In order to prevent this, constraints on the controllers K_k are introduced. With respect to Fig. 5.1b, the state x_{k+1} has to be pushed along the blue dashed line toward $x_{k+1}(\boldsymbol{\lambda}_{k+1})$.

Assumption 5.1. *The system is considered to be 1-step controllable, see Def. 3.3.*

This assumption is made to guarantee that the dynamics enables the system to bring x_{k+1} to a homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ at all. If this assumption is not satisfied, no controller K_k can be found that satisfies $x_{k+1} = x_{k+1}(\boldsymbol{\lambda}_{k+1})$.

As it can be seen in Fig. 5.1b, the blue bashed line represents a vector $n_{k+1}^{\perp} \in \mathbb{R}^{n_x}$ at time $k+1$, which is orthogonal to matrix $D_{x,k+1} = (x_{k+1}^1 - x_{k+1}^0)$ (the black dashed line at $k+1$):

$$n_{k+1}^{\perp} \cdot D_{x,k+1} = 0 \tag{5.44}$$

In this example, the matrix $D_{x,k+1}$ has only one column $D_{x,k+1} \in \mathbb{R}^{n_x \times 1}$, representing the directional vector from the state x_{k+1}^0 of the 0-th base trajectory to the state x_{k+1}^1 of base trajectory one, hence $D_{x,k+1} = (x_{k+1}^1 - x_{k+1}^0)$.

Lemma 5.5.

$$n_{k+1}^{\perp} \cdot x_{k+1} = n_{k+1}^{\perp} \cdot x_{k+1}^0, \text{ for any } x_{k+1}, \tag{5.45}$$

the state x_{k+1} at time $k+1$ must be a homotopic state: $x_{k+1} = x_{k+1}(\boldsymbol{\lambda}_{k+1})$.

Proof. Assume $x_{k+1} = x_{k+1}(\boldsymbol{\lambda}_{k+1})$, then (5.45) is:

$$n_{k+1}^{\perp} \cdot x_{k+1}(\boldsymbol{\lambda}_{k+1}) = n_{k+1}^{\perp} \cdot x_{k+1}^0. \tag{5.46}$$

With the homotopy equation (4.6), the equation (5.46) becomes:

$$n_{k+1}^{\perp} \cdot (x_{k+1}^0 + D_{x,k+1}\boldsymbol{\lambda}_{k+1}) = n_{k+1}^{\perp} \cdot x_{k+1}^0 \tag{5.47}$$

$$n_{k+1}^{\perp} \cdot x_{k+1}^0 + n_{k+1}^{\perp} \cdot D_{x,k+1}\boldsymbol{\lambda}_{k+1} = n_{k+1}^{\perp} \cdot x_{k+1}^0, \tag{5.48}$$

and since $n_{k+1}^{\perp} \cdot D_{x,k+1} = 0$, see (5.44), the equality condition (5.45) can only be satisfied as shown with $x_{k+1} \in x_{k+1}(\boldsymbol{\lambda}_{k+1})$.

Now with $n_c > 1$, the matrix $D_{x,k+1} \in \mathbb{R}^{n_x \times n_c}$ can have $n_c > 1$ columns for which the vector $n_{k+1}^{\perp}$ is also an orthogonal basis of dimension $n_{k+1}^{\perp} \in \mathbb{R}^{n_x \times n_c}$. □

The matrix $n_{k+1}^{\perp}$ is an orthogonal basis for the null space of $D_{x,k+1}$ obtained from the singular value decomposition. Hence, the singular value decomposition of $D_{x,k+1}$ is given by:

$$D_{x,k+1} = \mathbf{U}_{k+1}\mathbf{S}_{k+1}{\mathbf{V}_{k+1}}^T, \tag{5.49}$$

with the matrix $\mathbf{U}_{k+1} \in \mathbb{R}^{n_c \times n_c}$ containing the eigenvectors of $D_{x,k+1}{D_{x,k+1}}^T$, see Sec. 3.4. The null space $\mathcal{N}(D_{x,k+1})$ then has the orthogonal basis $n_{k+1}^{\perp}$ for which each column of $n_{k+1}^{\perp}$ is an eigenvector $\mathbf{u}_i \in \mathbf{U}_{k+1}$, such that ${\mathbf{u}_i}^T D_{x,k+1}$ has zero elements:

$$n_{k+1}^{\perp} = \mathcal{N}(D_{x,k+1}) := \{\mathbf{u}_i | {\mathbf{u}_i}^T D_{x,k+1} = 0\}. \tag{5.50}$$

With respect to these orthogonal bases, the connection between the requirement (5.46) and the controller synthesis can be given.

Lemma 5.6. *The requirement* (5.46) *is satisfied if the following LMI holds:*

$$\begin{pmatrix} 0 & ({n_{k+1}^{\perp}}^T B L_k)^T \\ {n_{k+1}^{\perp}}^T B L_k & Y \end{pmatrix} \geq 0, \tag{5.51}$$

with $L_k = K_k Y$.

Proof. Replacing x_{k+1} in (5.45) by the system dynamics (3.2) and the input $u_{new,k}(\boldsymbol{\lambda}_k)$ from (4.19), the equation becomes:

$${n_{k+1}^{\perp}}^T (Ax_k + Bu_{new,k}(\boldsymbol{\lambda}_k)) = {n_{k+1}^{\perp}}^T x_{k+1}^0 \tag{5.52}$$

$${n_{k+1}^{\perp}}^T (Ax_k + B(u_k(\boldsymbol{\lambda}_k) + u_{add,k})) = {n_{k+1}^{\perp}}^T x_{k+1}^0, \tag{5.53}$$

and with the control low (5.5):

$${n_{k+1}^{\perp}}^T (Ax_k + B(u_k(\boldsymbol{\lambda}_k) - K_k \delta\boldsymbol{\lambda}_k)) = {n_{k+1}^{\perp}}^T x_{k+1}^0. \tag{5.54}$$

Assuming that the state x_k is a homotopic state $x_k = x_k(\boldsymbol{\lambda}_k)$, since the controller K_{k-1} at time $k-1$ already achieved this, (5.54) together with the homotopy input (4.7) gets:

$${n_{k+1}^{\perp}}^T (A(x_k^0 + D_{x,k}\boldsymbol{\lambda}_k) + B(u_k^0 + D_{u,k}\boldsymbol{\lambda}_k - K_k\delta\boldsymbol{\lambda}_k)) = {n_{k+1}^{\perp}}^T x_{k+1}^0 \tag{5.55}$$

$${n_{k+1}^{\perp}}^T (\underbrace{Ax_k^0 + Bu_k^0}_{x_{k+1}^0} + \underbrace{(AD_{x,k} + BD_{u,k})}_{D_{x,k+1}} \boldsymbol{\lambda}_k - BK_k\delta\boldsymbol{\lambda}_k) = {n_{k+1}^{\perp}}^T x_{k+1}^0 \tag{5.56}$$

$$-{n_{k+1}^{\perp}}^T BK_k\delta\boldsymbol{\lambda}_k = 0. \tag{5.57}$$

Taking the norm and relaxing (5.57) yields:

$$||{n_{k+1}^{\perp}}^T BK_k\delta\boldsymbol{\lambda}_k|| \leq 0. \tag{5.58}$$

With $K_k = L_k Y^{-1}$ and $\delta\boldsymbol{\lambda}_k \in \mathcal{E}(P)$ (see (5.12)), the last inequality becomes:

$$||{n_{k+1}^{\perp}}^T BL_k Y^{-1} P^{-\frac{1}{2}} z|| \leq 0, \tag{5.59}$$

which equals with $P = Y^{-1}$:

$$||{n_{k+1}^{\perp}}^T B L_k Y^{-1} Y^{\frac{1}{2}}||\ ||z|| \leq 0. \tag{5.60}$$

Taking note that the supremum of the left side of (5.60) yields:

$$\sup_{||z||^2} ||{n_{k+1}^{\perp}}^T B L_k Y^{-1} Y^{\frac{1}{2}}||\ ||z|| = ||{n_{k+1}^{\perp}}^T B L_k Y^{-\frac{1}{2}}||^2, \tag{5.61}$$

the following steps can be executed:

$$||{n_{k+1}^{\perp}}^T B L_k Y^{-\frac{1}{2}}||^2 \leq 0 \tag{5.62}$$

$$({n_{k+1}^{\perp}}^T B L_k) Y^{-1} ({L_k}^T B^T n_{k+1}^{\perp}) \leq 0 \tag{5.63}$$

$$({n_{k+1}^{\perp}}^T B L_k) Y^{-1} ({n_{k+1}^{\perp}}^T B L_k)^T \leq 0, \tag{5.64}$$

and by applying the Schur complement to (5.64), the LMI (5.51) is obtained. □

With the derived constraints (5.15), (5.20) and (5.27), the objective (5.40), and the LMI (5.51), the complete synthesis procedure of this section is summarized by:

$$\min_{Y,\ L_k,\ Q_C} \quad \mathbf{tr}(Q_C) + \mathbf{tr}(Y) \tag{5.65a}$$

$$\text{s.t.} \quad (5.15),\ (5.20),\ (5.27),\ (5.51) \tag{5.65b}$$

$$\forall d_w \in \mathcal{D},\ \forall s \in \{1, ..., n_u\},\ \forall k \in \mathcal{J}_{\mathcal{N}}. \tag{5.65c}$$

The optimization problem determines time-dependent controller matrices K_k which guarantee a fast stabilizing transition between the trajectories, while satisfying the input constraints. The determined controllers are optimal for all $\{\boldsymbol{\lambda}_k, \bar{\boldsymbol{\lambda}}\} \in \mathbf{Co}\{\mathcal{C}\}$, since they are the optimal solution of the convex semi-definite program 5.65.

5.3. Numerical Example

The focus of this numerical example is not to show a scenario of an obstacle circumvention, as already introduced in Sec. 4.6 of Chapter 4. Rather, it is shown that the determined controllers, satisfy the input constraints for all possible transitions between homotopic trajectories. Thus, obstacle circumvention under input constraints can be accomplished with the here determined controllers and the introduced online circumvention procedure from the previous chapter.

For comparison, the same linear discrete-time system as in Sec. 4.6 is considered, see system matrices A and B of (4.83). The system dimension is $n_x = 3$ and the dimension of the input space $n_u = 3$. Furthermore, it can be shown that the

system is *1-step controllable*, see Sec. 3.7 for explanations of $\mathfrak{r}$-step controllability. In comparison to Chapter 4, the number of base trajectories is now chosen with $n_c = 2$. This leads to an overdetermined homotopic system dynamics (see, Sec. 4.3), for which the controller synthesis provides a solution. With $n_c = 2$, the set of *base trajectories* is : $\mathcal{X}_b = \{(\hat{x}^0, \hat{u}^0), (\hat{x}^1, \hat{u}^1), (\hat{x}^2, \hat{u}^2)\}$. The *base trajectories* are chosen as in Sec. 4.6. Each *base trajectory* has $N = 60$ discrete time steps. Again, the system starts at the initial state:

$$x_0 = x_s = [0, 0, 0]^T,$$

and reaches the end state:

$$x_N = x_f = [5, 5, 5]^T, \quad u_N = u_f = [0.5, 2.7, 0.7]^T.$$

Each element of the additional input value $u_{add,k}$ is bounded by:

$$|u_{max,s}| = 15. \tag{5.66}$$

The red colored region in Fig. 5.3 shows the permissible region of the delta homotopy space $\delta\boldsymbol{\lambda} \in \mathbf{Co}\{\mathcal{D}\}$. The red asterisks are the vertices $d_w \in \mathcal{D}$. The ellipsoid $\mathcal{E}(P)$ shows the region of attraction, determined when solving problem (5.65). Since the ellipsoid $\mathcal{E}(P)$ is a level set of the Lyapunov function (5.8), and while this level set is considered in the controller synthesis, the ellipsoid further guarantees stability for all initial values of $\delta\boldsymbol{\lambda} \in \mathbf{Co}\{\mathcal{D}\}$ by the fact that $\mathcal{E}(P)$ overapproximates $\mathbf{Co}\{\mathcal{D}\}$, hence $\mathcal{E}(P) \supseteq \mathbf{Co}\{\mathcal{D}\}$. Additionally, $\mathcal{E}(P)$ is also the smallest ellipsoid that still contains the permissible region $\mathbf{Co}\{\mathcal{D}\}$. This is obtained by minimizing the trace of Y, as formulated in the costs of the optimization problem. The minimized ellipsoid enables to maximize the use of the input $u_{add,k}$ within its constraints.

The satisfaction of the input constraints (5.66) on $u_{add,k}$ is shown in Fig. 5.4a. The illustration contains the enumeration of all inputs $u_{add,k}$ that occur when the system has to transition from one corner $d_w \in \mathcal{D}$ of the permissible region to any other corner, i.e., from one base trajectory to another. This enumeration is also done for every time step $k \in \mathcal{J}_\mathcal{N}$, since a transition can be initiated at every time step. The simulation results show all obtained input trajectories in Fig. 5.4b and all resulting homotopic trajectories in Fig. 5.4b. The results show that the maximum allowed input of $|u_{max,s}| = 15$ is not violated at any time. Obviously, the maximum inputs occur at time $k = 18$, when the *base trajectories* span the largest area for transition. The three given *base trajectories* are at the parts where the trajectories accumulate. If e.g. the input constraints were stricter, Fig. 5.4b would look more dense at its center, since no big steps can be made towards the target trajectory. A relaxation of the input constraints would allow the system to complete the transition faster such that the center in Fig. 5.4b would be sparsely occupied. Finally the simulations show that the controller synthesis provides stabilizing controllers which resolve the problems introduced above.

The computation is performed with the standard solver MOSEK [86]. For online circumvention with obstacles, the reader is revered to the method introduced in Sec. 4.5 of Chapter 4, which will be combined with the controllers determined here.

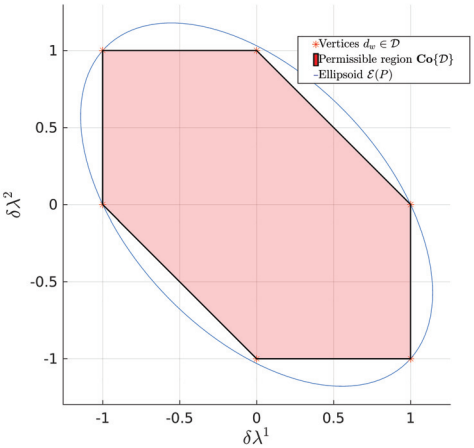
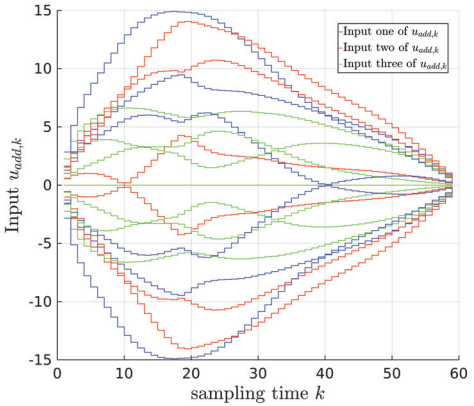
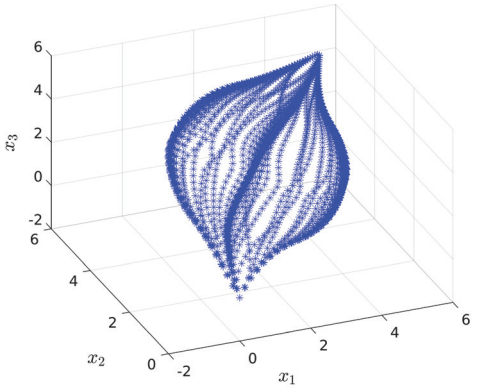


Figure 5.3.: Permissible region $\delta\boldsymbol{\lambda} \in \mathbf{Co}\{\mathcal{D}\}$ and the ellipsoidal region of attraction $\mathcal{E}(P)$ containing $\mathbf{Co}\{\mathcal{D}\}$.



(a) Input values of vector $u_{add,k}$.



(b) All state trajectories, when transition for each time step between the *base trajectories*.

Figure 5.4.: Enumeration of all state and input trajectories.

5.4. Discussion

This chapter has shown, that the use of homotopy properties for online circumvention of an obstacle can be extended to the two important aspects of including input constraints in the controller synthesis, and the problem of ambiguity in the controller synthesis, when the system has a higher state dimension than number of *base trajectories*. Under the assumption that the system dynamics is *1-step controllable*, it is shown by means of orthogonality conditions, that LMIs can be derived which force the states to be homotopic states. This requirement is especially important for the online procedure described in Sec. 4.5.

With respect to including input constraints in the controller synthesis, a new approach is presented that allows to formulate a linear, convex semi-definite program, and guarantees stability for the defined region of permissible homotopic values. With the developed optimization problem it is possible to determine controllers that realize transitions as fast as possible and consider input constraints. Therefore, a special cost function and LMI formulations for the Lyapunov descending conditions are introduced. The synthesized controllers of the method presented here can be used for online obstacle avoidance with the procedure described in Sec. 4.5.

While the developed online method so far is merely based on current information of the obstacle position, a circumvention may fail if the obstacle is too close, hence detected too late. To avoid this circumstance a logic extension is to take predicted obstacle information into account.

As mentioned, the class of systems was here restricted to *1-step controllable* systems. This further motivates to adapt the method to systems which are $\mathfrak{r}$-*step controllable*, with $\mathfrak{r} > 1$. The named extensions are part of the next chapter.

6. Homotopic Control Algorithms for Predicted Constraints

In the previous chapters, a control method was introduced that uses homotopy properties from offline determined *base trajectories* to compute a homotopic target trajectory that passes a moving obstacle free of collisions. The initial and final states were assumed to be fixed, and no obstacle position predictions were considered. The strategy for online circumvention evaluates a suitable homotopic target trajectory by means of so called *passing points* which were distributed on the obstacle surface. In Chapter 4, controllers for the realization of transitions between the homotopic trajectories were determined in an offline procedure for cases when the number of *base trajectories* is greater or equal to the state space dimension, $n_c \geq n_x$. This offline controller synthesis was then extended in Chapter 5 to the case in which less *base trajectories* are defined than the state space dimension is ($n_c < n_x$). Furthermore input constraints were embedded in the synthesis procedure. However this extension is limited to systems which are *1-step controllable*. This chapter now extends the previous results in two important aspects, to $\mathfrak{r}$-*step controllable* systems with $\mathfrak{r} > 1$, and the consideration of predicted obstacle positions.

The first extension to $\mathfrak{r}$-*step controllable* systems is particularly important since many physical systems have less inputs than states. This leads to the fact that one cannot guarantee that the state at every time step k is a homotopic state $x_k \neq x_k(\boldsymbol{\lambda}_k)$. Thus a controller synthesis as described in Chapter 5 is no longer possible for the here considered system class. The approach here is to guarantee that every $\mathfrak{r}$-th time step leads to a homotopic state $x_{k\mathfrak{r}} = x_{k\mathfrak{r}}(\boldsymbol{\lambda}_{k\mathfrak{r}})$. In place of determining control matrices K_k as in the previous chapters, an auxiliary linear time-varying system is defined, which describes the change of the homotopy parameter $\boldsymbol{\lambda}_k$ to its target value $\bar{\boldsymbol{\lambda}}$ on a lifted time scale. By means of this auxiliary system, control inputs can be derived such that at least the state of every $\mathfrak{r}$-th step is a homotopic state, $x_{k\mathfrak{r}} = x_{k\mathfrak{r}}(\boldsymbol{\lambda}_{k\mathfrak{r}})$. Hence, the state feedback approach from the previous chapters is replaced here by feed forward control.

The second extension of including predicted obstacle positions for online planning contributes to a safe and more efficient circumvention of the obstacle. Since these predicted information can already be considered during the online planning, a successful circumvention of the moving obstacle is more likely. Furthermore, the here presented online procedure is not based on the approach of using *passing points*. In contrast, the online procedure to be presented makes use of the half-planes of the obstacle and uses techniques to reduce the amount of online computations. There-

fore, a suitable homotopic target trajectory is determined with low effort by a tree search of moderate size. The online procedure of this chapter represents the core contribution of this work, and is a fundamental basis for the next chapters.

The structure of this chapter starts by describing the problem (Sec. 6.1), followed by the offline procedure of transitioning between homotopic trajectories (Sec. 6.2) and the online part (Sec. 6.3). Finally a numerical example is given in Sec. 6.4. The main ideas of this chapter were already reported in [62].

6.1. Problem Description

The problem definition is mainly based on the definition of Sec. 4.2. Hence, the task is to bring a linear discrete-time system from an initial state x_s to a final state x_f in N time steps, while avoiding collisions with a moving obstacle.

Assumption 6.1. *The considered linear discrete-time system, as given in* (3.2)*, is* $\mathfrak{r}$*-step controllable, with* $\mathfrak{r} > 1$*.*

According to Def. 4.3, a set of *base trajectories* $\mathcal{X}_b$ is given that spans a space for circumvention and by Assumption 4.1, the *base trajectories* end up in the same state. Again, the number of *base trajectories* is smaller than the state dimension, $n_c < n_x$. The time-varying obstacle is given in half-space representation according to Def. 4.4. In contrast to the previous chapters, the following assumption on the obstacle is made.

Assumption 6.2. *For a prediction horizon* $H \in \mathcal{J}_{\mathcal{N}}$*, the convex state space obstacle:*

$$\mathcal{P}_{x,k+j} := \{x_{k+j} |\ C_{k+j} x_{k+j} \leq d_{k+j}\} \subseteq \mathbb{R}^{n_x}, \tag{6.1}$$

with $C_{k+j} \in \mathbb{R}^{c \times n_x}$*, and* $d_{k+j} \in \mathbb{R}^c$ *is known for* $j \in \mathcal{J}_{\mathcal{H}} := \{0, ..., H\}$*.*

The position prediction may be obtained from estimation using appropriate models or from communicated information on the planned motion of $\mathcal{P}_{x,k+j}$. The advantage of using predicted information in the planning procedure plays an important role for the success and the quality of determined trajectories, and is understood as an essential component of this work.

Since the system is $\mathfrak{r}$*-step controllable* with $\mathfrak{r} > 1$, any state within the input constraints can be reached in at most $\mathfrak{r}$ time steps, if the controllability matrix has full rank: $\mathbf{rank}(C_{AB\mathfrak{r}}) = n_x$, see Sec. 3.7 for details. For the example of $\mathfrak{r} = 2$, Fig. 6.1 shows for starting from a homotopic state $x_k(\boldsymbol{\lambda}_k)$ at time k, that the state x_{k+1} cannot reach a homotopic state $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1})$ in one time step. According to this fact, state feedback controllers cannot be specified, as it was done in the previous chapters. However, with appropriate input values $u_{add,k}$ and $u_{add,k+1}$, a homotopic state can be reached after $\mathfrak{r}$-steps.

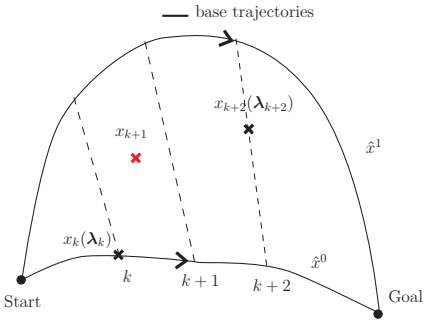


Figure 6.1.: Illustration of the problem for a 2-*step controllable* system where $x_{k+1} \neq x_{k+1}(\boldsymbol{\lambda}_{k+1})$, but $x_{k+2} = x_{k+2}(\boldsymbol{\lambda}_{k+2})$.

Problem 6.1. *Given are the following properties:*

1. *base trajectories $\mathcal{X}_b$ are given, and the number of given base trajectories is smaller then the system dimension $n_c < n_x$,*
2. *the system (3.2) is $\mathfrak{r}$-step controllable, with $\mathfrak{r} > 1$,*
3. *input constraints are specified: $|u_{add,k}| \leq u_{max}$*
4. *predicted obstacle information $\mathcal{P}_{x,k+j}$ is available, with $j \in \mathcal{J}_{\mathcal{H}}$.*

The system has to move from the current state $x_k = x_s$ along homotopic states $x_{k+j}(\boldsymbol{\lambda}_{k+j})$ towards a final state x_f, while considering predicted obstacle information to avoid collisions. The task is to find a homotopic target trajectory $\bar{\boldsymbol{\lambda}}$ for which the trajectory of transitioning from $\boldsymbol{\lambda}_k$ to $\bar{\boldsymbol{\lambda}}$ satisfies:

$$x_{k+j}(\boldsymbol{\lambda}_{k+j}) \notin \mathcal{P}_{x,k+j}, \ \forall j \in \mathcal{J}_{\mathcal{H}}. \tag{6.2}$$

The appropriate homotopic target trajectory $\bar{\boldsymbol{\lambda}}$ is based on the minimization of the cost function:

$$J(x_{k+j}, \ u_{k+j}) = \sum_{j=0}^{H} \|x_{k+j} - x_f\|_Q^2 + \|u_{k+j} - u_f\|_R^2, \tag{6.3}$$

evaluated within the prediction horizon H.

6.2. Transition Between Homotopic Trajectories

This section specifies how transitions between base trajectories are modified for $\mathfrak{r} > 1$. By means of an auxiliary system, an optimization problem can be stated

which parameterizes the auxiliary system such that the transitioning behavior is defined offline. The auxiliary system can then be used for the online procedure as described in Sec. 6.3.

Since only the state of $x_{\mathfrak{t}\mathfrak{r}} \in x_{\mathfrak{t}\mathfrak{r}}(\boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}})$ with $\mathfrak{t} \in \mathcal{T}_{\mathcal{N}} := \{0, ..., \lfloor N/\mathfrak{r} \rfloor\}$ is a homotopic state at every $\mathfrak{r}$-th time step, a new time scale $\mathfrak{t}$ is introduced, such that k is obtained from $k = \mathfrak{t}\mathfrak{r}$. In order to describe a discrete-time auxiliary system with indices increase by one step, the following notation is introduced:

$$z_{\mathfrak{t}} \mathrel{\hat{=}} x_{\mathfrak{t}\mathfrak{r}}, \tag{6.4}$$

$$\boldsymbol{\mu}_{\mathfrak{t}} \mathrel{\hat{=}} \boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}}. \tag{6.5}$$

This means that $z_{\mathfrak{t}} \in \mathbb{R}^{n_x}$ corresponds to the $\mathfrak{r}$-th state $x_{\mathfrak{t}\mathfrak{r}}$ such that the $\mathfrak{r}$-th homotopy vector $\boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}}$ is described in the new time scale with $\boldsymbol{\mu}_{\mathfrak{t}} \in \mathbb{R}^{n_c}$. The relation between both time scales is illustrated in Fig. 6.2 for the case of a *2-step controllable* system ($\mathfrak{r} = 2$).

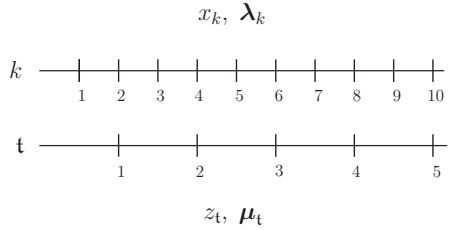


Figure 6.2.: Connection between the time scale k and $\mathfrak{t}$ with their corresponding states and homotopy values for a *2-step controllable* system.

Additionally, the homotopic state equation (4.6) now becomes:

$$z_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}}) = z_{\mathfrak{t}}^0 + D_{z,\mathfrak{t}}\boldsymbol{\mu}_{\mathfrak{t}}. \tag{6.6}$$

Definition 6.1. *Let the set $\mathcal{L}$ contain all base trajectories $\hat{x}^i \in \mathcal{X}_b$ described by their homotopy values $\bar{\boldsymbol{\mu}}^i \in \mathcal{L} := \{\bar{\boldsymbol{\mu}}^0, ..., \bar{\boldsymbol{\mu}}^{n_c}\}$ with $\bar{\boldsymbol{\mu}}^i \in \{0,1\}^{n_c \times 1}$, such that the base trajectories are described by vectors:*

$$\bar{\boldsymbol{\mu}}^0 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \bar{\boldsymbol{\mu}}^1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \bar{\boldsymbol{\mu}}^2 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \quad \dots, \quad \bar{\boldsymbol{\mu}}^{n_c} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

Now since every r-th state $x_{\mathfrak{t}r} \mathrel{\hat{=}} z_{\mathfrak{t}}$ is a homotopic state, an auxiliary linear time-varying (LTV) system is defined, which describes the change of the homotopy state $\boldsymbol{\mu}_{\mathfrak{t}}$, $\mathfrak{t} \in \mathcal{T}_{\mathcal{N}}$ to a final value $\bar{\boldsymbol{\mu}}$ by the following dynamics:

$$\boldsymbol{\mu}_{\mathfrak{t}+1} = \tilde{A}_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}. \tag{6.7}$$

With this auxiliary system, the transitioning behavior can be defined offline by parameterizing the matrices $\tilde{A}_{\mathfrak{t}} \in \mathbb{R}^{n_c \times n_c}$ in a later described optimization procedure. Thereby, the satisfaction of the input constraints strongly depends on the parameterization of these matrices. As already described in Sec. 4.3 by (4.19), a transition from an initial state $z_{\mathfrak{t}_0}(\boldsymbol{\mu}_{\mathfrak{t}_0})$, identified by the homotopy parameter $\boldsymbol{\mu}_{\mathfrak{t}_0}$ at time $\mathfrak{t}_0 \in \mathcal{T}_{\mathcal{N}}$, to a homotopic target trajectory referenced by $\bar{\boldsymbol{\mu}}$, needs an additional input value $u_{add,k}$. This additional input value is further constrained for all times $k \in \mathcal{J}_{\mathcal{N}}$ to:

$$u_{min} \leq u_{add,k} \leq u_{max}, \ \forall k \in \mathcal{J}_{\mathcal{N}}. \tag{6.8}$$

So the question arises, how the matrices $\tilde{A}_{\mathfrak{t}}$ of (6.7) have to be parametrized such that:

1. the auxiliary system (6.7) is stable,
2. the auxiliary system (6.7) transitions fast
3. the input constraints (6.8) are satisfied for all times $k \in \mathcal{J}_{\mathcal{N}}$.

As it can be seen, the auxiliary system is described in the time scale $\mathfrak{t} \in \mathcal{T}_{\mathcal{N}}$ (hence for every $\mathfrak{r}$-th step), but the "real" system (3.2) exists in the time scale $k \in \mathcal{J}_{\mathcal{N}}$. This means that when parameterizing the auxiliary system (6.7), the input signals also have to be satisfied in the time scale $k \in \mathcal{J}_{\mathcal{N}}$. Furthermore, the input constraints (6.8) also have to hold for all largest possible transitions, which are the transitions from any *base trajectory* $\bar{\boldsymbol{\mu}}^i$ to a *base trajectory* $\bar{\boldsymbol{\mu}}^s$, with $i, s \in \mathcal{M} := \{0, ..., n_c\}$ and $i \neq s$.

These conditions on the auxiliary system (6.7) can be satisfied by solving an eigenvalue problem on the matrices $\tilde{A}_{\mathfrak{t}}$. To obtain such a problem formulation, first the connection between the input constraints (6.8) and the matrices $\tilde{A}_{\mathfrak{t}}$ of (6.7) is shown.

Lemma 6.1. *The input constraints* (6.8) *hold for every time step $k \in \mathcal{J}_{\mathcal{N}}$, if the following inequality holds for all $\mathfrak{t} \in \mathcal{T}_{\mathcal{N}}$:*

$$U_{min} \leq (B, AB, ..., A^{\mathfrak{r}-1}B)^{\dagger} D_{z,\mathfrak{t}+1} (\tilde{A}_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_{\mathfrak{t}}) \leq U_{max}, \tag{6.9}$$

with $U_{min} = (u_{min}^T, ..., u_{min}^T)^T \in \mathbb{R}^{n_u r}$ and $U_{max} = (u_{max}^T, ..., u_{max}^T)^T \in \mathbb{R}^{n_u r}$.

Proof. In general, the state x_k of a linear discrete time system, as given in (4.6), can be determined from an initial state x_{k_0} by:

$$x_k = A^{k-k_0} x_{k_0} + \sum_{l=k_0}^{k-1} A^{k-l-1} B u_{new,l}(\boldsymbol{\lambda}_l), \tag{6.10}$$

where $u_{new,l}$ consists of the homotopic input and the additional input value as described in (4.19). Replacing $k = \mathfrak{t}\mathfrak{r}$, respectively $k_0 = \mathfrak{t}_0\mathfrak{r}$, and with (4.4) and (6.5), one can write:

$$z_{\mathfrak{t}} = A^{\mathfrak{t}\mathfrak{r}-\mathfrak{t}_0\mathfrak{r}} z_{\mathfrak{t}_0} + \sum_{l=\mathfrak{t}_0\mathfrak{r}}^{\mathfrak{t}\mathfrak{r}-1} A^{\mathfrak{t}\mathfrak{r}-l-1} B u_{new,l}(\boldsymbol{\lambda}_l). \tag{6.11}$$

Since $z_{\mathfrak{t}} = z_{\mathfrak{t}}(\boldsymbol{\mu}_t)$ is a homotopic state, (6.11) equals:

$$z_{\mathfrak{t}}(\boldsymbol{\mu}_t) = A^{\mathfrak{t}\mathfrak{r}-\mathfrak{t}_0\mathfrak{r}} z_{\mathfrak{t}_0}(\boldsymbol{\mu}_{\mathfrak{t}_0}) + \sum_{l=\mathfrak{t}_0\mathfrak{r}}^{\mathfrak{t}\mathfrak{r}-1} A^{\mathfrak{t}\mathfrak{r}-l-1} B u_{new,l}(\boldsymbol{\lambda}_l). \tag{6.12}$$

Now starting from time $\mathfrak{t}-1$ with state $z_{\mathfrak{t}-1}(\boldsymbol{\mu}_{\mathfrak{t}-1})$, (6.12) determines $z_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}})$ by:

$$z_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}}) = A^{\mathfrak{r}} z_{\mathfrak{t}-1}(\boldsymbol{\mu}_{\mathfrak{t}-1}) + \sum_{l=\mathfrak{t}\mathfrak{r}-\mathfrak{r}}^{\mathfrak{t}\mathfrak{r}-1} A^{\mathfrak{t}\mathfrak{r}-l-1} B u_{new,l}(\boldsymbol{\lambda}_l). \tag{6.13}$$

Furthermore, the states $z_{\mathfrak{t}-1}(\boldsymbol{\mu}_{\mathfrak{t}-1})$ on the right side and $z_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}})$ on the left are replaced according to (6.6). With $u_{new,l}$ from (4.19), one gets in matrix notation:

$$\begin{aligned} z_{\mathfrak{t}}^0 + D_{z,\mathfrak{t}}\boldsymbol{\mu}_{\mathfrak{t}} =& A^{\mathfrak{r}}(z_{\mathfrak{t}-1}^0 + D_{z,\mathfrak{t}-1}\boldsymbol{\mu}_{\mathfrak{t}-1}) \\ &+ (B, AB, ..., A^{\mathfrak{r}-1}B) \begin{pmatrix} u_{\mathfrak{t}\mathfrak{r}-1}^0 + D_{u,\mathfrak{t}\mathfrak{r}-1}\boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}-1} + u_{add,\mathfrak{t}\mathfrak{r}-1} \\ u_{\mathfrak{t}\mathfrak{r}-2}^0 + D_{u,\mathfrak{t}\mathfrak{r}-2}\boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}-2} + u_{add,\mathfrak{t}\mathfrak{r}-2} \\ \vdots \\ u_{\mathfrak{t}\mathfrak{r}-\mathfrak{r}}^0 + D_{u,\mathfrak{t}\mathfrak{r}-\mathfrak{r}}\boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}-\mathfrak{r}} + u_{add,\mathfrak{t}\mathfrak{r}-\mathfrak{r}} \end{pmatrix}. \end{aligned} \tag{6.14}$$

From the values $\boldsymbol{\lambda}_l$ in the matrix of (6.14), one can see that $\boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}-\mathfrak{r}}$ corresponds to $\boldsymbol{\mu}_{\mathfrak{t}-1} \hat{=} \boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}-\mathfrak{r}}$. The other values of $\boldsymbol{\lambda}_l$ for $l \in \{\mathfrak{t}\mathfrak{r}-\mathfrak{r}+1, ..., \mathfrak{t}\mathfrak{r}-1\}$ are located in between $\boldsymbol{\mu}_{\mathfrak{t}-1}$ and $\boldsymbol{\mu}_{\mathfrak{t}} \hat{=} \boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}}$. Since these intermediate values are not existing in the time scale $\mathfrak{t}$ of the auxiliary system, these values are set to $\boldsymbol{\mu}_{\mathfrak{t}-1}$:

$$\boldsymbol{\lambda}_l = \boldsymbol{\mu}_{\mathfrak{t}-1},\ \forall l \in \{\mathfrak{t}\mathfrak{r}-\mathfrak{r}+1, ..., \mathfrak{t}\mathfrak{r}-1\}, \tag{6.15}$$

corresponding to a zero-order hold of $\boldsymbol{\mu}_{\mathfrak{t}-1}$ and yielding:

$$\begin{aligned} z_{\mathfrak{t}}^0 + D_{z,\mathfrak{t}}\boldsymbol{\mu}_{\mathfrak{t}} =& A^{\mathfrak{r}}(z_{\mathfrak{t}-1}^0 + D_{z,\mathfrak{t}-1}\boldsymbol{\mu}_{\mathfrak{t}-1}) \\ &+ (B, AB, ..., A^{\mathfrak{r}-1}B) \begin{pmatrix} u_{\mathfrak{t}\mathfrak{r}-1}^0 + D_{u,\mathfrak{t}\mathfrak{r}-1}\boldsymbol{\mu}_{\mathfrak{t}-1} + u_{add,\mathfrak{t}\mathfrak{r}-1} \\ u_{\mathfrak{t}\mathfrak{r}-2}^0 + D_{u,\mathfrak{t}\mathfrak{r}-2}\boldsymbol{\mu}_{\mathfrak{t}-1} + u_{add,\mathfrak{t}\mathfrak{r}-2} \\ \vdots \\ u_{\mathfrak{t}\mathfrak{r}-\mathfrak{r}}^0 + D_{u,\mathfrak{t}\mathfrak{r}-\mathfrak{r}}\boldsymbol{\mu}_{\mathfrak{t}-1} + u_{add,\mathfrak{t}\mathfrak{r}-\mathfrak{r}} \end{pmatrix}. \end{aligned} \tag{6.16}$$

Replacing $\boldsymbol{\mu}_{\mathfrak{t}}$ on the left side according to the system dynamics of the auxiliary

system (6.7) and rewriting the terms leads to:

$$z_{\mathfrak{t}}^0 + D_{z,\mathfrak{t}}(\tilde{A}_{\mathfrak{t}-1}(\boldsymbol{\mu}_{\mathfrak{t}-1} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}) = \underbrace{A^{\mathfrak{r}} z_{\mathfrak{t}-1}^0 + (B, AB, ..., A^{\mathfrak{r}-1}B) \begin{pmatrix} u_{\mathfrak{t}\mathfrak{r}-1}^0 \\ u_{\mathfrak{t}\mathfrak{r}-2}^0 \\ \vdots \\ u_{\mathfrak{t}\mathfrak{r}-\mathfrak{r}}^0 \end{pmatrix}}_{z_{\mathfrak{t}}^0} +$$

$$\underbrace{\left(A^{\mathfrak{r}} D_{z,\mathfrak{t}-1} + (B, AB, ..., A^{\mathfrak{r}-1}B) \begin{pmatrix} D_{u,\mathfrak{t}\mathfrak{r}-1} \\ D_{u,\mathfrak{t}\mathfrak{r}-2} \\ \vdots \\ D_{u,\mathfrak{t}\mathfrak{r}-\mathfrak{r}} \end{pmatrix} \right)}_{D_{z,\mathfrak{t}}} \boldsymbol{\mu}_{\mathfrak{t}-1} + (B, AB, ..., A^{\mathfrak{r}-1}B) \begin{pmatrix} u_{add,\mathfrak{t}\mathfrak{r}-1} \\ u_{add,\mathfrak{t}\mathfrak{r}-2} \\ \vdots \\ u_{add,\mathfrak{t}\mathfrak{r}-\mathfrak{r}} \end{pmatrix}. \tag{6.17}$$

Equivalently to (6.11), the first term on the right side of (6.17) describes the state $z_{\mathfrak{t}}^0$, and the second term in the brace equals matrix $D_{z,\mathfrak{t}}$. Incrementing $\mathfrak{t} := \mathfrak{t} + 1$ yields:

$$z_{\mathfrak{t}+1}^0 + D_{z,\mathfrak{t}+1}(\tilde{A}_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}) = z_{\mathfrak{t}+1}^0 + D_{z,\mathfrak{t}+1}\boldsymbol{\mu}_{\mathfrak{t}} + (B, AB, ..., A^{\mathfrak{r}-1}B) \begin{pmatrix} u_{add,(\mathfrak{t}+1)\mathfrak{r}-1} \\ u_{add,(\mathfrak{t}+1)\mathfrak{r}-2} \\ \vdots \\ u_{add,(\mathfrak{t}+1)\mathfrak{r}-\mathfrak{r}} \end{pmatrix}, \tag{6.18}$$

which equals:

$$D_{z,\mathfrak{t}+1}(\tilde{A}_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_{\mathfrak{t}}) = (B, AB, ..., A^{\mathfrak{r}-1}B) \begin{pmatrix} u_{add,\mathfrak{t}\mathfrak{r}+\mathfrak{r}-1} \\ u_{add,\mathfrak{t}\mathfrak{r}+\mathfrak{r}-2} \\ \vdots \\ u_{add,\mathfrak{t}\mathfrak{r}} \end{pmatrix} \tag{6.19a}$$

$$\begin{pmatrix} u_{add,\mathfrak{t}\mathfrak{r}+\mathfrak{r}-1} \\ u_{add,\mathfrak{t}\mathfrak{r}+\mathfrak{r}-2} \\ \vdots \\ u_{add,\mathfrak{t}\mathfrak{r}} \end{pmatrix} = (B, AB, ..., A^{\mathfrak{r}-1}B)^{\dagger} D_{z,\mathfrak{t}+1}(\tilde{A}_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_{\mathfrak{t}}). \tag{6.19b}$$

Finally, constraining (6.19b) from the left and right completes the proof and represents the input constraints for one $\mathfrak{t} \in \mathcal{T}_{\mathcal{N}}$:

$$U_{min} \leq \begin{pmatrix} u_{add,\mathfrak{t}\mathfrak{r}+\mathfrak{r}-1} \\ u_{add,\mathfrak{t}\mathfrak{r}+\mathfrak{r}-2} \\ \vdots \\ u_{add,\mathfrak{t}\mathfrak{r}} \end{pmatrix} \leq U_{max}, \tag{6.20}$$

□

As described in the problem definition, the transition to a homotopic target trajectory $\bar{\boldsymbol{\mu}}$ is described by the auxiliary system (6.7). Selecting a suitable candidate $\bar{\boldsymbol{\mu}}$ is discussed in the next section. In order to make the best possible use of the constrained input signal (6.9), the matrices $\tilde{A}_\mathrm{t}$ are chosen to be time-varying, since $D_{z,\mathrm{t}+1}$ in (6.9) also changes over time.

For fast transitioning between homotopic trajectories, the eigenvalues of the matrices $\tilde{A}_\mathrm{t}$ have to be minimized subject to the input constraints (6.9). As described in [12], the eigenvalue problem (EVP) is to minimize the maximum eigenvalues of $\tilde{A}_\mathrm{t}$. This allows to formulate the convex EVP with the input constraints as a semi-definite program. By choosing $\tilde{A}_\mathrm{t}$ as diagonal matrices, the problem can be given by:

$$\min_{\tilde{A}_\mathrm{t},\gamma} \quad \gamma \tag{6.21a}$$

$$\text{s.t.} \quad \tilde{A}_\mathrm{t} - \gamma I \leq 0, \ \forall t \in \mathcal{T}_\mathcal{N} \tag{6.21b}$$

$$(6.9), \ \forall \{\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}^i, \bar{\boldsymbol{\mu}}^s\} \in \mathcal{L}, \ \text{with } i \neq s, \ \forall t \in \mathcal{T}_\mathcal{N}. \tag{6.21c}$$

With inequality (6.21b) and the cost function (6.21a), the eigenvalues of the matrices $\tilde{A}_\mathrm{t}$ are minimized. Thereby, fast convergence of the auxiliary system is limited by the input constraints in (6.21c) for which the inputs are constrained for all possible transitions between a *base trajectory* $\bar{\boldsymbol{\mu}}^i \in \mathcal{L}$ towards a *base trajectory* $\bar{\boldsymbol{\mu}}^s \in \mathcal{L}$ and for all time steps $t \in \mathcal{T}_\mathcal{N}$.

By solving this problem, the transitioning behavior of the auxiliary system is defined offline and is used in the online part to obtain a cost-efficient and circumventing homotopic target trajectory $\bar{\boldsymbol{\mu}}$.

6.3. Online Control

The online procedure starts by determining the cost optimal homotopic target trajectory encoded by $\bar{\boldsymbol{\mu}}^*$ for a given prediction horizon H, while first neglecting the obstacle. If the resulting trajectory fails to be free of collision, a fast online procedure is initiated. It determines an optimized value of $\bar{\boldsymbol{\mu}}^*$, such that the obstacle is passed free of collisions within the prediction horizon while minimizing the performance function (6.3). The advantage of using homotopies is that the optimization only has to be performed over the homotopy parameter $\bar{\boldsymbol{\mu}}$ instead of optimizing over all inputs and all time steps, see Sec. 6.3.1. Furthermore, the advantage of using homotopies is that the obstacle location can be exactly identified, respectively mapped, in the homotopy space which is spanned by the *base trajectories*. Thus, only the relevant and collision critical time steps can be identified which are important for the circumvention procedure described in Sec. 6.3.2. The procedure in Sec. 6.3.3 then determines a circumventing and optimal homotopic target trajectory encoded by $\bar{\boldsymbol{\mu}}^*$ using a tree search.

6.3.1. Optimal Homotopic Trajectory

The online procedure starts by first neglecting the obstacle and determining the optimal homotopic target trajectory $\bar{\boldsymbol{\mu}}^*$ within the prediction horizon H. Starting from an initially executed homotopic trajectory $\boldsymbol{\mu}_{\mathfrak{t}}$ at time $\mathfrak{t}$ with corresponding state $x_{\mathfrak{t}\mathfrak{r}}$, an optimization problem is given that determines the best homotopic target trajectory. The cost function of this problem evaluates the states and inputs for $j \in \mathcal{J}_{\mathcal{H}} = \{0, ..., H\}$, and since the time scale of the auxiliary system (6.7) has a lower resolution, the predicted states are indexed by $j_{\mathfrak{t}} \in \mathcal{T}_{\mathcal{H}} := \{0, ..., \lfloor H/\mathfrak{r} \rfloor\}$:

$$
\begin{aligned}
&\texttt{OHT}: \\
\min_{\bar{\boldsymbol{\mu}}} \quad & \sum_{j=0}^{H} \|x_{\mathfrak{t}\mathfrak{r}+j} - x_f\|_Q^2 + \|u_{new,\mathfrak{t}\mathfrak{r}+j} - u_f\|_R^2 && \text{(6.22a)} \\
\text{s.t.} \quad & x_{k+1} = Ax_k + Bu_{new,k},\ \forall k \in \{(\mathfrak{t}+j_{\mathfrak{t}})\mathfrak{r}, ..., (\mathfrak{t}+j_{\mathfrak{t}})\mathfrak{r}+\mathfrak{r}-1\} && \text{(6.22b)} \\
& u_{new,k} = u_k^0 + D_{u_k}\boldsymbol{\mu}_{\mathfrak{t}+j_{\mathfrak{t}}} + u_{add,k},\ \forall k \in \{(\mathfrak{t}+j_{\mathfrak{t}})\mathfrak{r}, ..., (\mathfrak{t}+j_{\mathfrak{t}})\mathfrak{r}+\mathfrak{r}-1\} && \text{(6.22c)} \\
& \begin{pmatrix} u_{add,(\mathfrak{t}+j_{\mathfrak{t}})\mathfrak{r}+\mathfrak{r}-1} \\ u_{add,(\mathfrak{t}+j_{\mathfrak{t}})\mathfrak{r}+\mathfrak{r}-2} \\ \vdots \\ u_{(\mathfrak{t}+j_{\mathfrak{t}})\mathfrak{r}} \end{pmatrix} = (B, AB, ..., A^{\mathfrak{r}-1}B)^{\dagger} D_{z,\mathfrak{t}+j_{\mathfrak{t}}+1}(\tilde{A}_{\mathfrak{t}+j_{\mathfrak{t}}}(\boldsymbol{\mu}_{\mathfrak{t}+j_{\mathfrak{t}}} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_{\mathfrak{t}+j_{\mathfrak{t}}}) && \text{(6.22d)} \\
& \boldsymbol{\mu}_{\mathfrak{t}+j_{\mathfrak{t}}+1} = \tilde{A}_{\mathfrak{t}+j_{\mathfrak{t}}}(\boldsymbol{\mu}_{\mathfrak{t}+j_{\mathfrak{t}}} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} && \text{(6.22e)} \\
& \forall j_{\mathfrak{t}} \in \mathcal{T}_H. && \text{(6.22f)}
\end{aligned}
$$

The optimization problem (6.22) shows that by means of the homotopy concept, the task of this optimization problem is just to "select" a homotopic target trajectory by optimizing over the only variable $\bar{\boldsymbol{\mu}}$, instead of optimizing over e.g. all states and inputs within the prediction horizon as normally done in MPC. This essentially reduces the search space of the problem from $(H+1) \cdot n_u \cdot n_x$ to n_c variables (hence $\bar{\boldsymbol{\mu}} \in \mathbb{R}^{n_c}$). As it can be further noticed the problem is just equality constrained, which means that for one value $\bar{\boldsymbol{\mu}}$ during the optimization process, first the auxiliary system in (6.22e) can be solved such that with the values $\boldsymbol{\mu}_{\mathfrak{t}+j_{\mathfrak{t}}}$, the intermediate inputs (6.22d) are obtained, which again are required to determine the states x_{k+1} (6.22b) and inputs $u_{new,k}$ (6.22c) for evaluation of the costs (6.22a). By assuming a well-conditioned problem with respect to the numerical properties, an equality constrained quadratic program (as (6.22) is one) can be solved with this formulation quite efficiently, even for large prediction horizons. As noticed, the optimization problem contains no constraints on the input signal, while these have already been integrated in the matrices $\tilde{A}_t$ by the solution of (6.21).

In the case of detecting a collision of the obstacle with the obtained states $x^*_{\mathfrak{t}\mathfrak{r}+j}$, that results when transitioning from the currently executed trajectory $\boldsymbol{\mu}_t$ to the

targeted $\bar{\boldsymbol{\mu}}^*$, an online circumventing procedure is initiated. Therefore the state space obstacle $\mathcal{P}_{x,k+j}$ has to be first mapped into the homotopy space.

6.3.2. Transformation of the Obstacle into the Homotopy Space

Now, the resulting state trajectory is checked against collision, i.e. $x^*_{\mathfrak{t}\mathfrak{r}+j} \notin \mathcal{P}_{x,\mathfrak{t}\mathfrak{r}+j}$ is evaluated $\forall j \in \mathcal{J}_{\mathcal{H}}$. If collisions are found, the state space obstacle $\mathcal{P}_{x,k+j}$ has to be mapped into the homotopy space. Since it was shown in the problem definition (Sec. 6.1), that only every $\mathfrak{r}$-th step, $x_{\mathfrak{t}\mathfrak{r}} = x_{\mathfrak{t}\mathfrak{r}}(\boldsymbol{\lambda}_{\mathfrak{t}\mathfrak{r}})$ with $\mathfrak{t} \in \mathcal{T}_{\mathcal{N}}$ is homotopic, also only every $\mathfrak{r}$-th time step of $\mathcal{P}_{x,\mathfrak{t}\mathfrak{r}}$ can be mapped into the homotopy space. The predicted polytope for every $\mathfrak{r}$-th time step is given according to:

$$\mathcal{P}_{x,(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}} := \{x_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}} |\ C_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}}\ x_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}} \leq d_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}}\},\ j_\mathfrak{t} \in \mathcal{T}_{\mathcal{H}}. \tag{6.23}$$

Again, with the new time scale $\mathfrak{t}$, and the corresponding notation (6.4), the polytope can be rewritten to:

$$\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}} := \{z_{\mathfrak{t}+j_\mathfrak{t}} |\ C_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}}\ z_{\mathfrak{t}+j_\mathfrak{t}} \leq d_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}}\},\ j_\mathfrak{t} \in \mathcal{T}_{\mathcal{H}}, \tag{6.24}$$

and with the homotopy equation (6.6), the polytope (6.24) becomes:

$$\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}} := \{z_{\mathfrak{t}+j_\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}) |\ C_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}}\ z_{\mathfrak{t}+j_\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}) \leq d_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}}\},\ j_\mathfrak{t} \in \mathcal{T}_{\mathcal{H}}, \tag{6.25}$$

The polytope $\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}}$ mapped into the homotopy space for one $j_\mathfrak{t} \in \mathcal{T}_{\mathcal{H}}$ is denoted by $\mathcal{P}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$, and is determined as follows:

$$C_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}}(z^0_{\mathfrak{t}+j_\mathfrak{t}} + D_{z,\mathfrak{t}+j_\mathfrak{t}}\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}) \leq d_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}} \tag{6.26}$$

$$\underbrace{C_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}} D_{z,\mathfrak{t}+j_\mathfrak{t}}}_{C_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}}} \boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} \leq \underbrace{d_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}} - C_{(\mathfrak{t}+j_\mathfrak{t})\mathfrak{r}} z^0_{\mathfrak{t}+j_\mathfrak{t}}}_{d_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}}} \tag{6.27}$$

$$\Rightarrow \mathcal{P}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}} := \{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} | C_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}\ \boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} \leq d_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}\}, \tag{6.28}$$

where $C_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}} \in \mathbb{R}^{c \times n_c}$ and $d_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}} \in \mathbb{R}^c$ denote the matrices of the half-space representation of the transformed polytope $\mathcal{P}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ at time $\mathfrak{t} + j_\mathfrak{t}$.

For illustration of the polytope mapping, it is referred to Fig. 6.3. The figure shows two *base trajectories* $\hat{x}^0$ and $\hat{x}^1$, for which the homotopic trajectories are defined. The possible homotopic states $x_k(\boldsymbol{\lambda}_k)$ for each time step k are shown by dashed/dotted lines. For the sake of clarity, the figure shows a polytopic obstacle $\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}}$, which is constant over time, and the prediction horizon is assumed to be N. Furthermore, the system is considered to be *2-step controllable*, hence only every second homotopic state can be reached exactly $x_{\mathfrak{t}2} = x_{\mathfrak{t}2}(\boldsymbol{\lambda}_{\mathfrak{t}2})$, $\mathfrak{t} \in \mathcal{T}_{\mathcal{N}}$. This is shown by the blue and red dashed lines in Fig. 6.3. For these times, the mapping (6.26)-(6.28) is performed.

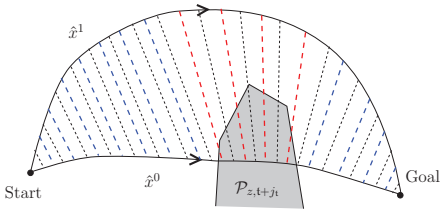


Figure 6.3.: 2-*step controllable* system with homotopic states $z_\mathfrak{t}(\boldsymbol{\mu}_\mathfrak{t})$ (blue dashed lines), and critical time steps for which a collision between the polytope and a homotopic state is possible (red dashed lines).

Identifying Possible Collision Times

A mapping of every r-th polytope (as described by (6.26)-(6.28)) can either result in an empty or non-empty polytope $\mathcal{P}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$. A polytope is non-empty, if the intersection between $\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}}$ and $z_{\mathfrak{t}+j_\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}})$ is nonempty:

$$\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}} \cap z_{\mathfrak{t}+j_\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}) \neq \emptyset \;\Rightarrow\; \mathcal{P}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}} \neq \emptyset. \tag{6.29}$$

Thus, time steps can be identified for which (6.29) holds. These time steps are colored red in Fig. 6.3 and are stored in a set denoted by $\mathcal{T}_\mathcal{I}$. It can be concluded that the red colored time steps are the critical times which are relevant for the collision avoidance. Hence, the blue colored times can be excluded from the collision avoidance procedure, since no collision for this time steps is ever possible while moving on homotopic states. The computation time of the obstacle avoidance procedure clearly benefits from this obstacle localization advantage, which is given by the idea of using homotopic trajectories.

To first check whether a polytope $\mathcal{P}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ is empty or not, a simple linear optimization problem can be set up. If the problem has a feasible solution, the obstacle $\mathcal{P}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ at time $\mathfrak{t}+j_\mathfrak{t}$ is nonempty:

$$\min_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}} \quad b^T\, \boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} \tag{6.30a}$$

$$\text{s.t.} \quad C_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}}\, \boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} \leq d_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}} \tag{6.30b}$$

$$\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} \in [0,1]^{n_c},\; \sum_{i=1}^{n_c} \mu^i_{\mathfrak{t}+j_\mathfrak{t}} \leq 1, \tag{6.30c}$$

where $b \neq 0 \in \mathbb{R}^{n_c}$ can be chosen arbitrary. The set of all times $j_\mathfrak{t} \in \mathcal{T}_\mathcal{H}$, for which (6.30) has a feasible solution is denoted by:

$$\mathcal{T}_\mathcal{I} := \{j_\mathfrak{t} |\ (6.30) \text{ is feasible}\} \subseteq \mathcal{T}_\mathcal{H}, \tag{6.31}$$

and contains times where a collision between a homotopic state $z_{\mathfrak{t}+j_\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}})$ and the obstacle $\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}}$ is possible. These time steps are considered in the collision avoidance procedure of Sec. 6.3.3. If $\mathcal{T}_\mathcal{I} = \emptyset$, no collision between the moving obstacle and the system occurs within the prediction horizon $j_\mathfrak{t} \in \mathcal{T}_\mathcal{H}$.

Removing inactive Half-Planes

While the mapping procedure (6.26)-(6.28) maps the state space obstacle from the higher-dimensional state space to a lower dimensional homotopy space $\mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_c}$ (due to $n_c < n_x$, see Problem 6.1), inactive half-planes may exists in $\mathcal{P}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$. This circumstance is shown in Fig. 6.4. The intersection of the polytope $\mathcal{P}_{z,\mathrm{t}+j_\mathrm{t}}$ with the set of homotopic states $z_{\mathrm{t}+j_\mathrm{t}}(\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}})$ determines the desired polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$ (red area). But because of the other two half-planes of $\mathcal{P}_{z,\mathrm{t}+j_\mathrm{t}}$ (colored green), two further intersections with $z_{\mathrm{t}+j_\mathrm{t}}(\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}})$ exist which are irrelevant for the description of the forbidden region. The goal is to find a homotopy parameter $\bar{\boldsymbol{\mu}}$ such that $\bar{\boldsymbol{\mu}}$ is not element of the redundancy revised polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$: $\bar{\boldsymbol{\mu}} \notin \tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$. Since the inactive half-planes unnecessarily expand the forbidden region and therefore have to be removed from $\mathcal{P}_{z,\mathrm{t}+j_\mathrm{t}}$ to obtain $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$. These half-planes can be removed by linear programming as described in [17].

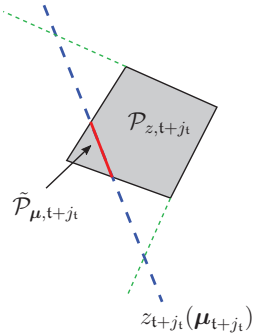


Figure 6.4.: Redundant half-planes arising when mapping $\mathcal{P}_{z,\mathrm{t}+j_\mathrm{t}}$ into $\mathcal{P}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$.

Definition 6.2. *Let the w-th half-plane of $\mathcal{P}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$ be denoted by $C^w_{\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}}}$ and $d^w_{\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}}}$ with $w \in \mathcal{W}_{\mathrm{t}+j_\mathrm{t}}$, where $\mathcal{W}_{\mathrm{t}+j_\mathrm{t}}$ is the index set of all half-planes.*

The w-th half-plane of $\mathcal{P}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$ is redundant, if the linear program:

$$\max_{\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}}} \quad C^w_{\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}}}\, \boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}} \tag{6.32a}$$

$$\text{s.t.} \quad C^i_{\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}}}\, \boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}} \leq d^i_{\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}}},\ \forall i \in \mathcal{W}_{\mathrm{t}+j_\mathrm{t}} \backslash \{w\}, \tag{6.32b}$$

has an optimal value less or equal to $d^w_{\boldsymbol{\mu}_t}$. Solving the problem for all $w \in \mathcal{W}_{\mathrm{t}+j_\mathrm{t}}$ yields the polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}$ without redundant half-planes (hence the red polytope of Fig. 6.4):

$$\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}} := \{\boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}} | \tilde{C}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}\, \boldsymbol{\mu}_{\mathrm{t}+j_\mathrm{t}} \leq \tilde{d}_{\boldsymbol{\mu},\mathrm{t}+j_\mathrm{t}}\} \tag{6.33}$$

The w-th half-plane of the polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ is similarly defined as in Def. 6.2 with corresponding matrices $\tilde{C}^w_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ and $\tilde{d}^w_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$. The set of possible half-planes of the reduced polytope at time $\mathfrak{t}+j_\mathfrak{t}$ is denoted by $\tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}$.

6.3.3. Homotopic Control Algorithm

The Homotopic Control Algorithm `HCA`, see Alg. 6.2 of this section, shows the process of driving the dynamic system from an initial to a final state along an optimal homotopic target trajectory $\bar{\boldsymbol{\mu}}^*$, while avoiding collisions with a time-varying obstacle and taking into account information on predicted obstacle positions. The `HCA` can be separated into four main parts:

1. Optimal Trajectory without Obstacle
2. Fallback Strategy
3. Tree-Search for Circumvention
4. Trajectory Pushing,

which are explained in detail in the following.

Step 1: Optimal Trajectory without Obstacle (line 3 of `HCA`)

Starting from the initial state $z_\mathfrak{t}(\boldsymbol{\mu}_\mathfrak{t}) = x_s$ at time $\mathfrak{t} := 0$, the algorithm first determines the best homotopic target trajectory $\bar{\boldsymbol{\mu}}^*$, within the prediction horizon $j_\mathfrak{t} \in \mathcal{T}_\mathcal{H}$, by solving problem `OHT` according to (6.22), where the obstacle $\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}}$ is neglected. This is done in order to attempt a first computationally cheep trial for determining an optimal state sequence of states $\phi^*_x = (x^*_{\mathfrak{t}\mathfrak{r}+0}, ..., x^*_{\mathfrak{t}\mathfrak{r}+H})$. Subsequently, the obtained state sequence is also checked on the time scale k against collisions with $\mathcal{P}_{x,\mathfrak{t}\mathfrak{r}+j}$, $\forall j \in \mathcal{J}_\mathcal{H}$. If no collisions exist, the algorithm terminates successfully. If the trajectory collides with the obstacle, the following collision avoidance procedure starts.

Step 2: Fallback Strategy (line 8 of `HCA`)

After detection of an collision, this part determines state sequences $\phi^{\bar{\boldsymbol{\mu}}^i}_x = (x_{\mathfrak{t}\mathfrak{r}+0}(\bar{\boldsymbol{\mu}}^i), \dots, x_{\mathfrak{t}\mathfrak{r}+H}(\bar{\boldsymbol{\mu}}^i))$, which transition from the current homotopic state (with its current homotopic parameter $\boldsymbol{\mu}_\mathfrak{t}$) towards each *base trajectory* $\bar{\boldsymbol{\mu}}^i \in \mathcal{L}$. Then the states $x_{\mathfrak{t}\mathfrak{r}+j}(\bar{\boldsymbol{\mu}}^i)$, $\forall j \in \mathcal{J}_\mathcal{H}$, are evaluated according to their costs (see (6.22)) and the least costly and collision-free solution $\bar{\boldsymbol{\mu}}^{i^*}$ within the set $\mathcal{L}$ is selected as a fallback trajectory. The trajectory towards this fallback trajectory represents a solution which is not the optimal solution to circumvent the obstacle, but can be determined quickly since only the costs have to be evaluated. At this point, it is noted that during the search of a circumventing trajectory with less costs,

the fallback trajectory determined in this step is always contained in the solution space when determining the better circumventing trajectory. This guarantees that a feasible solution can always be found, since at least one base trajectory has to be free of collision (see Assumption 4.2).

Step 3: Tree Search for Circumvention (line 9 of `HCA`)

Based on a tree search, this part of the `HCA` now determines algorithmically a circumventing trajectory, while solving an optimization problem during each step of the tree. For a better illustration, the tree is shown in Fig. 6.5. A level of the tree represents the trees depth. Each tree level refers to a time $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$. The tree depth is equal to the cardinality of the set of intersection times $|\mathcal{T}_\mathcal{I}|$. A branch within a tree level from a root node is denoted by the variable $w \in \tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}$. The number of possible branches in a tree level is given by the cardinality $|\tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}|$, which depends on the selected time $j_\mathfrak{t}$. The information stored for each node is a structure:

$$node_{j_\mathfrak{t},w} = (j_\mathfrak{t}, w, \bar{\boldsymbol{\mu}}^*, J(\bar{\boldsymbol{\mu}}^*), \phi_x^{\bar{\boldsymbol{\mu}}^*}, \phi_z^{\bar{\boldsymbol{\mu}}^*}, \mathcal{O}), \tag{6.34}$$

and the set of all nodes from the tree level $j_\mathfrak{t}$ is denoted by:

$$Node_{j_\mathfrak{t}} := \{node_{j_\mathfrak{t},1}, ..., node_{j_\mathfrak{t},|\tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}|}\}, \tag{6.35}$$

The node $node_{j_\mathfrak{t},w}$ contains information on the considered time $j_\mathfrak{t}$, one half-plane $w \in \tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}$ of the reduced polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$, a determined optimal homotopic target value $\bar{\boldsymbol{\mu}}^*$, the corresponding costs $J(\bar{\boldsymbol{\mu}}^*)$ with respect to the performance function in (6.22a), the resulting state sequence towards the optimal homotopic target trajectory $\phi_x^{\bar{\boldsymbol{\mu}}^*}$, and the state sequence of every r-th time step $\phi_z^{\bar{\boldsymbol{\mu}}^*}$. The set $\mathcal{O}$ stores the best nodes, $node^*_{j_\mathfrak{t},w} \in Node_{j_\mathfrak{t}}$, identified in each tree level so far. Thus when the optimization begins from the trees root, the set $\mathcal{O}$ is first empty. The information of a time step $j_\mathfrak{t}$ and half-plane w of a single node, $node_{j_\mathfrak{t},w} \in \mathcal{O}$, is obtained from the function:

$$TimePlane : node_{j_\mathfrak{t},w} \to \mathbb{N}_{\geq 0} \times \mathbb{N}_{>0},\ j_\mathfrak{t} \mapsto s,\ w \mapsto l, \tag{6.36}$$

such that one can identify a time step s and half-plane l of node $node_{j_\mathfrak{t},w}$ with:

$$(s, l) = TimePlane(node_{j_\mathfrak{t},w}). \tag{6.37}$$

The following optimization problem now selects a node $node_{j_\mathfrak{t},w}$ and determines a solution for this node. By selecting a node the critical collision time $j_\mathfrak{t}$ and the half-plane w of $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ are selected and the problem is solved for this node by solving `OHT` with the below additional constraints. These constraints are required for the

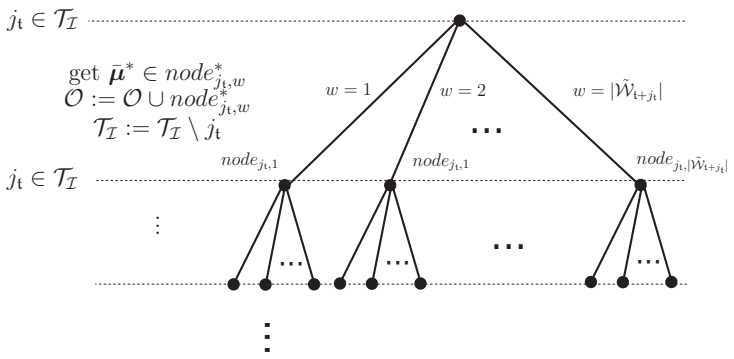


Figure 6.5.: Tree with levels $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$, decisions $w \in \tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}$, and $node_{j_\mathfrak{t},|\tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}|}$ giving the nodes.

collision avoidance.

$$\texttt{OHTadd}:$$

$$\text{select a } node_{j_\mathfrak{t},w} \text{ and solve} \tag{6.38a}$$

$$\texttt{OHT} \tag{6.38b}$$

additionally s.t.

$$\left.\tilde{C}^w_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}} \bar{\boldsymbol{\mu}}^{i^*} \geq \tilde{d}^w_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}}\right\} \text{ensures that the trajectory from } \textit{Step 2} \text{ is always contained in the solution space} \tag{6.38c}$$

$$\left.\tilde{C}^w_{\boldsymbol{\mu}_t} \boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} \geq \tilde{d}^w_{\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}}\right\} \text{collision avoidance constraint in time step } t \tag{6.38d}$$

$$\left.\begin{array}{l}\tilde{C}^l_{\boldsymbol{\mu}_{\mathfrak{t}+s}} \boldsymbol{\mu}_{\mathfrak{t}+s} \geq \tilde{d}^l_{\boldsymbol{\mu}_{\mathfrak{t}+s}}, \\ (s,l) = TimePlane(node_{j_\mathfrak{t},w}),\ \forall node_{j_\mathfrak{t},w} \in \mathcal{O}\end{array}\right\} \text{consideration of the trees sequence so far} \tag{6.38e}$$

$$\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} = \left(\prod_{i=\mathfrak{t}+j_\mathfrak{t}}^{\mathfrak{t}} \tilde{A}_i\right)(\boldsymbol{\mu}_\mathfrak{t} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}. \tag{6.38f}$$

The optimization problem consists of the `OHT` (see (6.22)) with three additional inequalities (6.38c)-(6.38e) to ensure the collision avoidance. Since for a considered node, $node_{j_\mathfrak{t},w}$, the time step $j_\mathfrak{t}$ and a half-plane w is known, the first constraint (6.38c) checks if the determined fallback value $\bar{\boldsymbol{\mu}}^{i^*}$, obtained from *Step 2*, is contained in the solution space. This is satisfied by checking if $\bar{\boldsymbol{\mu}}^{i^*}$ is outside (and therefore not contained in the polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$) of the w-th half-plane. The advantage of including this inequality in the optimization is that only nodes are explored which contain the fallback trajectory $\bar{\boldsymbol{\mu}}^{i^*}$ from *Step 2*. Hence, a feasible termination of the tree search algorithm is always guaranteed. The second inequality (6.38d) ensures that the homotopy value $\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}$ at time $\mathfrak{t}+j_\mathfrak{t}$ is outside of the w-th half-plane of $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$, which satisfies the trajectory to be free of collision for this single point

of time. The inequality (6.38e) considers the time steps and half-planes which are obtained from the trees sequence so far, hence from the trees path connecting the explored nodes. This is important because if the solution of the optimization problem would only consider the currently selected collision time $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$ of a tree layer and a half-plane, the solution $\boldsymbol{\mu}^*$ would pass the obstacle $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ on a half plane at the selected time $j_\mathfrak{t}$ but may not be a feasible solution at any other time. Thus, it is important to start from a time step and consider step by step the selected collision times $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$ and half planes $w \in \tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}$ in the optimization of the current node. The constraint (6.38e) then guarantees that the already considered time steps are still free of collision.

Lemma 6.2. *Since the optimization is performed over the variable $\bar{\boldsymbol{\mu}}$, the corresponding values $\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}}$ for* (6.38d) *and* (6.38e) *are obtained according to* (6.38f).

Proof. By considering the dynamics of the auxiliary system (6.7), one recursively obtains:

$$\boldsymbol{\mu}_{\mathfrak{t}+1} = \tilde{A}_\mathfrak{t}(\boldsymbol{\mu}_\mathfrak{t} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}, \tag{6.39}$$

$$\begin{aligned}\boldsymbol{\mu}_{\mathfrak{t}+2} &= \tilde{A}_{\mathfrak{t}+1}(\boldsymbol{\mu}_{\mathfrak{t}+1} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} = \tilde{A}_{\mathfrak{t}+1}(\tilde{A}_\mathfrak{t}(\boldsymbol{\mu}_\mathfrak{t} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} - \bar{\boldsymbol{\mu}})) + \bar{\boldsymbol{\mu}} \\ &= \tilde{A}_{\mathfrak{t}+1}\tilde{A}_\mathfrak{t}(\boldsymbol{\mu}_\mathfrak{t} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}},\end{aligned} \tag{6.40}$$

$$\vdots$$

$$\boldsymbol{\mu}_{\mathfrak{t}+j_\mathfrak{t}} = \left(\prod_{i=\mathfrak{t}+j_\mathfrak{t}}^{\mathfrak{t}} \tilde{A}_i\right)(\boldsymbol{\mu}_\mathfrak{t} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}. \tag{6.41}$$

□

The optimization problem `OHTadd` is embedded into a tree search algorithm `BfSearch`, which is based on a best-first strategy, and is shown in Alg. 6.1. The best-first search is a search algorithm which evaluates all nodes on a tree level and expands the most promising node according to the best costs (here solution of the `OHTadd`), see [50] for graph search algorithms. The algorithm `BfSearch` starts from a homotopic state $\boldsymbol{\mu}_\mathfrak{t}$ at initial time $\mathfrak{t} := \mathfrak{t}_0$ and selects first a intersection time $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$ (*line 3*), representing a tree level. According to this time, all nodes (representing different half-planes) are determined by solving the optimization problem `OHTadd` for each node (*line 4-5*). The solutions on each node are stored in the set $Node_{j_\mathfrak{t}}$ (*line 6*). Subsequently, the costs $J(\bar{\boldsymbol{\mu}}^*)$ of all nodes of the tree level, $node_{j_\mathfrak{t},w} \in Node_{j_\mathfrak{t}}$ are compared, and the node with minimal costs is selected to be further explored (*line 7*). This best-first selection is a function

$$Bestnode:\ 2^{Node_{j_\mathfrak{t}}} \to Node_{j_\mathfrak{t}}, \tag{6.42}$$

with $2^{Node_{j_\mathfrak{t}}}$ representing the power set, and the best node is given according to:

$$node^*_{j_\mathfrak{t},w} = Bestnode(Node_{j_\mathfrak{t}}). \tag{6.43}$$

At this step it can be guaranteed only that the obtained trajectory $\phi_z^{\bar{\boldsymbol{\mu}}^*} \in node^*_{j_\mathfrak{t},w}$ is free of collisions for the currently considered time $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$ and the so far considered time steps $j_\mathfrak{t}$ obtained from the set $\mathcal{O}$. As a consequence, to guarantee the complete trajectory $\phi_z^{\bar{\boldsymbol{\mu}}^*} \in node^*_{j_\mathfrak{t},w}$ to be free of collisions, a collision check is performed (*line 9*). If the check fails and a collision still exists, the currently explored time $j_\mathfrak{t}$ is removed from the set of intersection times $\mathcal{T}_\mathcal{I}$ (*line 10*), and the algorithm continues the tree search by executing the while loop again. If however, $\phi_z^{\bar{\boldsymbol{\mu}}^*}$ is free of collisions, the algorithm terminates immediately, otherwise, if the set $\mathcal{T}_\mathcal{I}$ becomes empty. Finally, the algorithm provides a homotopic target trajectory $\bar{\boldsymbol{\mu}}^* \in node^*_{j_\mathfrak{t},w}$, such that the auxiliary system (6.7) drives the "real" system (3.2), without collisions, from its current state towards the homotopic target trajectory $\bar{\boldsymbol{\mu}}^*$.

Step 4: Trajectory Pushing (line 10-14 of `HCA`)

Since the obstacle avoidance, as well as the collision checking of the `BfSearch` algorithm operate on the time scale $\mathfrak{t}$, a collision can nevertheless be detected for an intermediate point of time k, i.e. for time steps $k \in \{\mathfrak{t}\mathfrak{r}, ..., \mathfrak{t}\mathfrak{r} + H\}$ which are not considered during the collision avoidance in *Step 3*. This part then pushes the trajectory out of the obstacle, until the states on the intermediate points of time are also free of collision. Therefore, the trajectory obtained in *Step 3* (encoded by $\bar{\boldsymbol{\mu}}^*$) is pushed iteratively towards the fallback trajectory. This is realized by iteratively pushing the obtained homotopy value $\bar{\boldsymbol{\mu}}^*$ toward $\bar{\boldsymbol{\mu}}^{i^*}$ with a parameter $\alpha_{push} \in [0, 1]$, according to:

$$\bar{\boldsymbol{\mu}}^* := \bar{\boldsymbol{\mu}}^* + \alpha_{push}(\bar{\boldsymbol{\mu}}^{i^*} - \bar{\boldsymbol{\mu}}^*). \tag{6.44}$$

By iteratively increasing $\alpha_{push} \in [0, 1]$ the obtained state trajectory ϕ_x^* (determined with equations (6.22b)-(6.22f)) becomes free of collisions.

Homotopic Control Algorithm `HCA`

The complete procedure is shown in Alg. 6.2. After termination of the `HCA`, the first step of the control strategy is applied to the system, the prediction horizon is shifted one step forward and the computation is repeated. This leads to a receding horizon implementation of the `HCA`.

An illustration on how the algorithm works is schematically shown in Fig. 6.6. The `HCA` starts in Fig. 6.6a by considering the first intersection time identified at $\mathfrak{t}+4$ and then determines a trajectory which is collision-free according to the mapped polytope at this point of time $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+4}$ (red line). Furthermore, it is guaranteed that the fallback trajectory $\bar{\boldsymbol{\mu}}^{i^*}$ is still contained in the solution space. Hence it is not possible to pass $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+4}$ from the bottom side, since this would exclude $\bar{\boldsymbol{\mu}}^{i^*}$ from the solution space. Since it can be seen that the solution $\bar{\boldsymbol{\mu}}^*$ provides a trajectory which is not free of collisions for the remaining intersection times $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$, the tree search in the `HCA` iteratively considers further intersection times (see Fig. 6.6b) and therefore

Algorithm 6.1.: Best-first Search of a Homotopic Target Trajectory (`BfSearch`)

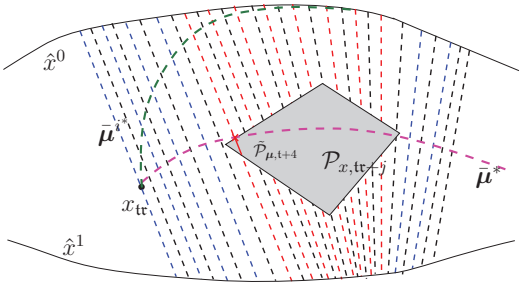
Given: $\mathcal{T}_\mathcal{I}$, initial time $\mathfrak{t} := \mathfrak{t}_0$, $\boldsymbol{\mu}_\mathfrak{t}$, fallback trajectory $\bar{\boldsymbol{\mu}}^{i^*}$, $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$, $\mathcal{O} = \emptyset$
while $\mathcal{T}_\mathcal{I} \neq \emptyset$ **do**
  select $j_\mathfrak{t} \in \mathcal{T}_\mathcal{I}$
  determine all nodes $node_{j_\mathfrak{t},w}$, $w \in \tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}$ by solving
  `OHTadd` according to (6.38)
  $Node_{j_\mathfrak{t}} := \{node_{j_\mathfrak{t},1}, ..., node_{j_\mathfrak{t},|\tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}|}\}$
  select best node according to $node^*_{j_\mathfrak{t},w} = Bestnode(Node_{j_\mathfrak{t}})$
  $\mathcal{O} := \mathcal{O} \cup node^*_{j_\mathfrak{t},w}$
  if $\exists j_\mathfrak{t} \in \mathcal{T}_\mathcal{H} : (\phi_z^{\bar{\boldsymbol{\mu}}^*} \in node^*_{j_\mathfrak{t},w}) \in \mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}}$ **then**
    $\mathcal{T}_\mathcal{I} := \mathcal{T}_\mathcal{I} \setminus j_\mathfrak{t}$
  else
    exit while loop
  end if
end while
Return($\bar{\boldsymbol{\mu}}^*$)

Algorithm 6.2.: Homotopic Control Algorithm (`HCA`)

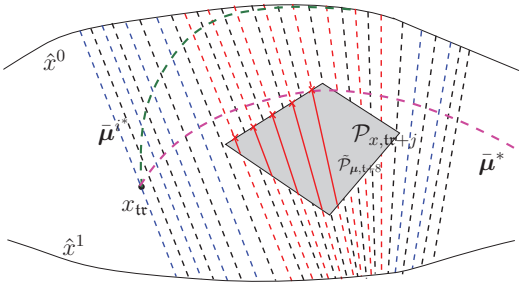
Given: $\boldsymbol{\mu}_\mathfrak{t}$, $\mathcal{J}_\mathcal{H}$, $\tilde{A}_\mathfrak{t}$, $\mathcal{P}_{x,\mathfrak{t}\mathfrak{r}+j}$, x_f^0, $\mathcal{O} = \emptyset$
Define: $\mathfrak{t} := \mathfrak{t}_0$
(*Step 1*) compute $\bar{\boldsymbol{\mu}}^*$ according to (6.22), while neglecting the obstacle $\mathcal{P}_{x,\mathfrak{t}\mathfrak{r}+j}$
if $\exists j \in \mathcal{J}_\mathcal{H} : x_{\mathfrak{t}\mathfrak{r}+j} \in \mathcal{P}_{x,\mathfrak{t}\mathfrak{r}+j}$ **then**
  map every r-th polytopes into the homotopy space $\mathcal{P}_{z,\mathfrak{t}+j_\mathfrak{t}} \rightarrow \tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$,
  see Sec. 6.3.2
  determine intersection times $\mathcal{T}_\mathcal{I}$
  (*Step 2*) determine collision-free fallback trajectory $\bar{\boldsymbol{\mu}}^{i^*}$
  (*Step 3*) execute the tree search algorithm `BfSearch`
  if $\exists j \in \mathcal{J}_\mathcal{H} : x_{\mathfrak{t}\mathfrak{r}+j} \in \mathcal{P}_{x,\mathfrak{t}\mathfrak{r}+j}$ **then**
    (*Step 4*) push obtained trajectory towards the fallback trajectory
    from *Step 2*, until trajectory becomes additionally collisions-free
    for the intermediate states
  end if
end if
Return(optimal collision free homotopic target trajectory $\bar{\boldsymbol{\mu}}^*$)

further polytopes. At time $\mathfrak{t} + 8$, the resulting trajectory colored in magenta is free of collisions for every $\mathfrak{r}$-th point of time. Nevertheless, it can be seen that the resulting trajectory still collides for an intermediate time step. This forces the

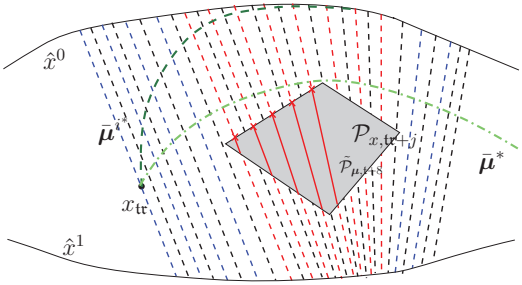
pushing procedure to be executed. As a result, the system moves to a collision free homotopic target trajectory $\bar{\boldsymbol{\mu}}^*$ (colored in bright green).



(a) `HCA` starts by considering the first intersection time $\mathfrak{t}+4$ in the tree search procedure.



(b) Progress of the `HCA` after a few iterations of the tree search. The tree search is now completed and a collision free trajectory for every $\mathfrak{r}$-th point of time is obtained (magenta).



(c) The pushing procedure is executed to guarantee a collision-free trajectory even for the intermediate time steps.

Figure 6.6.: Illustration of the `HCA` for an timely constant polytope $\mathcal{P}_{x,\mathfrak{t}\mathfrak{r}+j}$. Every r-th homotopic state is shown in blue, and possible intersection times by red dashed lines. The fallback trajectory $\bar{\boldsymbol{\mu}}^{i^*}$ is colored dark green. A transformed polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu},\mathfrak{t}+j_\mathfrak{t}}$ is by a red line, the intermediate solutions are shown in magenta and the final solution in bright green.

6.4. Numerical Example

The proposed homotopy control method is applied to a point mass moving in a 3-D space, according to the dynamics:

$$\ddot{x}(t) = -\frac{c}{m}\dot{x}(t) + \frac{1}{m}F_x(t) \tag{6.45a}$$

$$\ddot{y}(t) = -\frac{c}{m}\dot{y}(t) + \frac{1}{m}F_y(t) \tag{6.45b}$$

$$\ddot{z}(t) = -\frac{c}{m}\dot{z}(t) + \frac{1}{m}F_z(t), \tag{6.45c}$$

with input vector $\boldsymbol{u}(t) = (F_x(t), F_y(t), F_z(t))^T$, the mass m and a friction constant c. The variables x, y, z describe the Cartesian coordinates of the 3-D space in this example. The state vector of system (6.45a)-(6.45c) is given by $\boldsymbol{x}(t) = (x(t), \dot{x}(t), y(t), \dot{y}(t), z(t), \dot{z}(t))^T$. The system is discretized (ZOH) with a discretization time $T = 0.1$ seconds (sec.). With parameters $m = 2$ and $c = 1$, the discrete-time system $\hat{\boldsymbol{x}}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k$ has the following matrices:

$$A = \begin{pmatrix} 1 & 0.0975 & 0 & 0 & 0 & 0 \\ 0 & 0.9512 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0975 & 0 & 0 \\ 0 & 0 & 0 & 0.9512 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.0975 \\ 0 & 0 & 0 & 0 & & 0.9512 \end{pmatrix}, \; B = 1e^{-2} \begin{pmatrix} 0.25 & 0 & 0 \\ 4.88 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 4.88 & 0 \\ 0 & 0 & 0.25 \\ 0 & 0 & 4.88 \end{pmatrix}. \tag{6.46}$$

The system is 2-step controllable according to Def. 3.3. Totally three pairs of base trajectories $(\hat{x}^i, \hat{u}^i)$ are chosen to span a region for circumvention, i.e. $n_c = 2$. The final time for all $i \in \mathcal{M}$ is $N = 60$. All state trajectories start at the same state $x_0^i = x_s = [0, 0, 0, 0, 0, 0]^T$ and end in the final state $x_N^i = x_f = [5, 0, 5, 0, 5, 0]^T$. The base trajectories are shown in Fig. 6.7 by the three black dotted lines. The upper and lower bound on the additional input $u_{add,k}$, which limits how fast a trajectory can change to a homotopic target trajectory, is given by $u_{max} = [100, 100, 100]^T$ and $u_{min} = [-100, -100, -100]^T$. The obstacle $\mathcal{P}_{x,k}$ starts at time $k = 0$ at the upper right corner of the state space (x, y, z), shown by the transparent polytope with dashed edges in Fig. 6.7. It moves towards the lower left region until time $k = 10$, where it remains for the rest of the time. The final position is shown by the green polytope. The cost function (6.22a) is parametrized by $Q \in \mathbb{R}^{6\times6}$, chosen as the identity matrix and $R \in \mathbb{R}^{3\times3}$ as a diagonal matrix with values $1e^{-3}$, implying that trajectories with lower costs reach the final state x_f more "directly".

The computation is performed in Matlab using a PC with an Intel Core i7 (3.4GHz). The blue trajectory (see Fig. 6.7) is determined by the `HCA` starting at initial time $k = 0$ with a prediction horizon of $H = 10$. With respect to the receding horizon principle and the fact that the system is 2-step controllable, the algorithm first lets the system execute two steps, then the prediction horizon is

shifted forward and the HCA repeats its calculation starting from the new current state. From Fig. 6.7, it can be seen that the trajectory first follows an optimal homotopic trajectory and at $k = 5$ reacts to the moving polytope by transitioning to an other homotopic target trajectory. This can also be observed in Fig. 6.8 by the blue colored bars, which show the computation times for the homotopy control method with receding horizon control. It can be identified that during the first steps the computation time is very low, with approx. 3 milliseconds (ms). When the obstacle becomes relevant during the prediction horizon, the computation time raises up to 20ms and finally falls again when the obstacle is passed. Since the system is 2-step controllable, the computation has to be performed only on every second time step, which results in a total amount of 30 time steps of computations, as shown in Fig. 6.8. Although the determination is only performed at every $\mathfrak{r}$-step, the controls for the intermediate times are also determined by the procedure. This also means that a reaction to the detection and the motion of the obstacle can also only be done at every $\mathfrak{r}$-step.

For the same scenario, but with a prediction horizon of $H = 60$, the solution of the homotopy control method is given by the red trajectory in Fig. 6.7. Here, it can be seen that the system reacts at the very beginning to the moving obstacle. The computation time (see Fig. 6.8 red bars) is approx. 40ms at the beginning. Compared to the smaller prediction horizon, the computational load for the entire time is larger. However, for comparison, the computation times of a standard MPC based on MIP solved with CPLEX yields solver times of approx. 800ms at each iteration. Thus, the computation times are drastically reduced by approx. 96% (two orders of magnitude) overall, when using the proposed scheme.

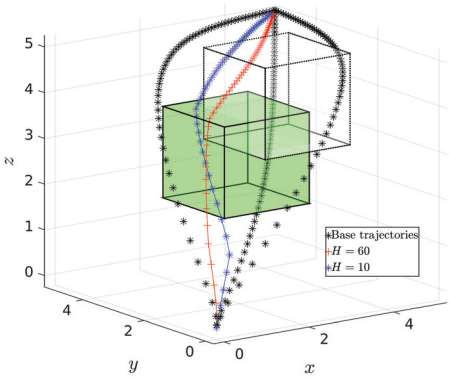


Figure 6.7.: Simulation of the base trajectories $\hat{x}^1$, $\hat{x}^2$, $\hat{x}^3$ (black stars), the trajectory (blue stars) obtained by the receding horizon `HCA` with $H = 10$, and the trajectory (red crosses) for $H = 60$. The obstacle $\mathcal{P}_{x,k}$ moves from the initial position (transparent dashed polytope) to the final position (green polytope).

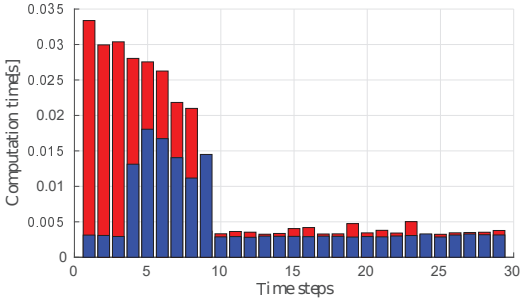


Figure 6.8.: Comparison between computation times of the receding horizon `HCA` with prediction horizon $H = 10$ (blue bars), and $H = 60$ (red bars). The computation times are shown for the entire simulation based on time scale $\mathbf{t}$.

6.5. Discussion

This chapter extended the problem class to $\mathfrak{r}$-step controllable systems, with $\mathfrak{r} > 1$. An auxiliary system was introduced that describes the transition behavior between homotopic trajectories. While the determination of the homotopic trajectories and the transition behavior are accomplished offline, a time-consuming online computation of complete trajectories is avoided, making the online computation very efficient. This chapter further considered two important aspects for a more efficient and safer circumvention of a moving obstacle, namely the possibility of passing the obstacle along its edges (and not only at *passing points* from the previous chapter), and the consideration of obstacle predictions. Therefore, the time-varying obstacle was mapped into the homotopy space. Indeed, this additional mapping process leads to further calculations, but the computational effort is negligible, since it is limited to the solution of a few linear programs that identify possible intersection times and reduce redundant half-planes of the polytopes. The proposed method identifies time steps at which collisions with a moving obstacle may occur. These are used in a collision avoidance scheme based on a tree search, namely the `HCA`.

The main advantage is that the homotopy control algorithm performs its computations over an entire trajectory (which is coded by the homotopy parameter) and not as usually done in trajectory optimization for each time step. This reduces the optimization problem to a low-dimensional problem, optimizing just over the homotopy parameter $\bar{\boldsymbol{\mu}}$, see (6.22). Furthermore, due to the homotopic space spanned by the *base trajectories*, the subset of those time steps being relevant for collision avoidance is identified. By consideration of only these intersection times in the optimization, see (6.38), the efficiency of the proposed method is increased significantly. A further advantage of this method is that if a collision-free *base trajectory* can be found, the `HCA` always guarantees to find a circumventing trajectory by considering the *base trajectory* in the subsequent tree search. In an outlook, an even bigger reduction of the computation times could be achieved, if one can presort the set $\mathcal{T}_{\mathcal{I}}$. Then the algorithm could start with the time step $j_{\mathfrak{t}} \in \mathcal{T}_{\mathcal{I}}$ that results the biggest deformation of the homotopy trajectory for circumvention. Then the other critical times would be free of collision immediately. This strategy would quickly lead to a termination of the tree search part after the exploration of a few time steps $j_{\mathfrak{t}} \in \mathcal{T}_{\mathcal{I}}$.

The simulations show that the proposed method computes trajectories with significantly lower computational effort than standard MPC, but it must be pointed out that the circumvention is limited to only homotopic trajectories, hence to a range of trajectories which are given by the *base trajectories*. The solution variety of MPC (like stopping the system, turning back or circumventing the obstacle in a different way) is not fully possible by the here developed homotopy method. But the variety can be extended by including more *base trajectories* with different behaviors. However, for many applications where obstacle positions can be estimated before, the solution variety of MPC is not required and the proposed homotopy method shows its strengths.

In this chapter the initial and target states were fixed. This allowed one to use pre-designed *base trajectories* for solving the offline transition behavior problem and then an online problem for circumventing the obstacle. If however, the target state may change over time, the pre-defined *base trajectories* are no longer correct, since they end up in the old target state. Therefore, a procedure is required, which for the first time in this thesis designs online *base trajectories* in terms of specific criteria like e.g., the size of the spanned space for circumvention, and then adapts these *base trajectories* to the time-varying target states. The fact that the basic trajectories are constantly changing also applies to nonlinear systems, if linearization is carried out at every time step. This extension is investigated in the next chapter.

Part III.

Homotopic Control Algorithms for Nonlinear Systems

7. Homotopic Control for Nonlinear Systems

The previous chapter has shown that the HCA solve the obstacle avoidance problem in low computational time. The algorithm allows to consider predictions on obstacle positions paired with the advantage of circumventing the obstacle along its half-planes for further cost reduction. A significant advantage of the HCA is, that a subset of time steps can be identified to which the collision avoidance procedure can be limited. This and the reduction of the number of optimization variables, by means of homotopic descriptions, allows to determine collision-free trajectories quickly. In advance to this method, an offline part specifies how transitions between homotopic trajectories can be realized, i.e. this part requires prior given *base trajectories*. Thus, the target state was fixed over the entire time to enable that the *base trajectories* can be determined offline. However, e.g. for the human-robot scenario, it is quite likely that time-varying target states arise. E.g. when a robotic manipulator has to work on an object moving along an conveyor belt, while at the same time collisions with a human have to be prevented. This chapter now extends the circumvention by considering time-varying target states. This issue implies, that the *base trajectories* have to be online determined and to be adapted when the target state changes. Therefore, the HCA is extended to a completely online version, namely the OnHCA, by a procedure which online realizes suitable *base trajectories* that span a space for circumvention. Besides taking time-varying target states into account, a further advantage of determining *base trajectories* online is that nonlinear dynamics can be considered during the planning. As mentioned above, a robotic manipulator represents a nonlinear system. When the manipulator moves, its current state is used for online determining linearized models of the system dynamics and consequently adapting the *base trajectories* according to the new model description tailored to the current state. With the obtained set of *base trajectories*, a circumventing homotopic target trajectory can then be identified.

This chapter is organized such that a definition of the problem is given first (Sec. 7.1), followed by describing the procedure for online determination and adaption of *base trajectories* (Sec. 7.2). The obtained *base trajectories* are used in the online homotopic control algorithm for the determination of a homotopic target trajectory, which circumvents the obstacle (Sec. 7.3). Finally, a numerical example is shown for a nonlinear robotic manipulator, as it is typically found in industrial assembly processes (Sec. 7.4).

7.1. Problem Description

As already described in the previous chapters, the basic setting is to drive a dynamic system from an initial state $x_s(t)$ to a target state $x_f(t)$ while avoiding collisions with an obstacle and satisfying input constraints. The obstacle $\mathcal{P}_{x,k+j|k}$ is again assumed to be time-varying and known from time step k within the prediction horizon $k+j$, hence for all $j \in \mathcal{J}_{\mathcal{H}}$, as given by Assumption 6.2. The inputs are taken from the convex constrained input set: $u_{k+j} \in \mathcal{U}$.

Assumption 7.1. *The considered dynamics is nonlinear and continuous in time, as described by* (3.1)*.*

While the procedure proposed here relies on a linear and discrete-time model description, the nonlinear dynamics is approximated by linearization and time discretization as described in Sec. 3.5.

Definition 7.1. *Let a nonlinear, continuous time system $\dot{x}(t) = f(x(t), u(t))$ be given. A linearization at time k at current state x_L and input u_L followed by a zero-order hold discretization from Sec. 3.4 leads to the linear model:*

$$\begin{aligned} x_{k+j+1|k} &= A_{:|k}(x_{k+j|k} - x_L) + B_{:|k}(u_{k+j|k} - u_L) + f(x_L, u_L), \\ &= A_{:|k}\ x_{k+j|k} + B_{:|k}\ u_{k+j|k} + \underbrace{f(x_L, u_L) - A_{:|k}\ x_L - B_{:|k}\ u_L}_{r_{:|k}}, \end{aligned} \tag{7.1}$$

where $A_{:|k}$, $B_{:|k}$, and $r_{:|k}$ denote the system matrices obtained at time k, and assumed to be constant over the planning horizon, see Def. 3.26.

With this linearized dynamics, a set of offline predetermined *base trajectories* is obviously unsuitable. While updating the linear model (7.1) to its current state and time, the *base trajectories*, used to span the space for circumvention, also have to be updated.

Definition 7.2. *The overall time-varying goal state is denoted by $x^0_{f|k}$. Simultaneously, this state is also the goal state of the 0-th base trajectory $\hat{x}^0$.*

Consequently, if the goal state changes over time, the *base trajectory* also has to be adapted to the new goal.

The problem to be addressed in this chapter thus becomes the following.

Problem 7.1. *Given be the linearized dynamics* (7.1)*, input constraints $u_{k+j} \in \mathcal{U}$, the current state $x_{k|k}$ of the system, and a time-varying goal state $x^0_{f|k}$. The task is to control the nonlinear system by means of the linearized dynamic towards the time-varying goal state, while avoiding collisions with a time-varying obstacle $\mathcal{P}_{x,k+j|k}$. The obstacle location is assumed to be predicted by an approximated model and is known within the prediction horizon H. The procedure is implemented in a receding horizon scheme.*

The `OnHCA` to be developed selects at every time k a suitable homotopic target trajectory $\boldsymbol{\lambda}^*_{:|k}$, which optimizes the following cost function over the prediction horizon H:

$$J(x_{k+j|k}, u_{k+j|k}) = \|x_{k+H|k} - x^0_{f|k}\|^2_{Q_{end}} + \sum_{j=0}^{H-1} \|x_{k+j|k} - x^0_{f|k}\|^2_Q + \|u_{k+j|k} - u^0_{f|k}\|^2_R \tag{7.2}$$

7.2. Online Determination and Adaptation of Base Trajectories

This section explains the online determination of *base trajectories*. Compared to the previous chapters, the *base trajectories* here do not end in the same state $x^0_{f|k}$. While $x^0_{f|k}$ is the overall goal state to be reached, the *base trajectories* are forced to reach different target states $x^i_{f|k}$ and inputs $u^i_{f|k}$, $\forall i \in \mathcal{M}$.

$$x^l_{f|k} \neq x^i_{f|k}, \forall \{l, i\} \in \mathcal{M},\ l \neq i. \tag{7.3}$$

These target states are generated and adapted in a specific scheme around the goal state $x^0_{f|k}$. This is done to span a space for the circumvention. Thus if an obstacle blocks the motion to the goal state, a homotopic trajectory to one of the *base trajectories* is selected for circumvention. Finally if the obstacle is passed, a homotopic trajectory is forced to be selected by its cost function that reaches the overall goal state. The selection of a homotopic trajectory is always forced to reach the goal state in case of no obstacle collision. Thus, the system does not move any further towards the target states $x^i_{f|k}$ $i \in \mathcal{M} \setminus 0$, or a homotopy between them in the case of a successful circumvention. The section then continues with a coordinate transition for the determination of the *base trajectories*. This is done in order to force the *base trajectories* to continuously span a wide space for the circumvention.

A *base trajectories* $\hat{x}^i$, $\hat{u}^i$ are obtained by solving a quadratic optimization problem in which the deviations of the states and inputs to their target states $x^i_{f|k}$ and inputs $u^i_{f|k}$ are penalized. The number of time steps over which the state and input *base trajectories* are defined, is given by the prediction horizon H. The optimization problem is:

$$\min_{x_{k+j|k}, u_{k+j|k}} \quad \|x_{k+H|k} - x^i_{f|k}\|^2_{Q_{B,end}} + \sum_{j=0}^{H-1} \|x_{k+j|k} - x^i_{f|k}\|^2_{Q_B} + \|u_{k+j|k} - u^i_{f|k}\|^2_{R_B} \tag{7.4a}$$

$$\text{s.t.:} \quad x_{k+j+1|k} = A_{:|k}\, x_{k+j|k} + B_{:|k}\, u_{k+j|k} + r_{:|k} \tag{7.4b}$$

$$u_{k+j|k} \in \mathcal{U},\ \forall j \in \mathcal{J}_{\mathcal{H}}, \tag{7.4c}$$

with matrices $Q_B \in \mathbb{S}^{n_x}_{>0}$, $Q_{B,end} \in \mathbb{S}^{n_x}_{>0}$, and $R_B \in \mathbb{S}^{n_u}_{>0}$ as weighting matrices for designing the *base trajectories* at time step k. The end term in (7.4a) can be used

to force the system to reach the target state with higher priority. As it can be seen, the *base trajectories* are determined again without considering the obstacle, while the circumvention of the obstacle is followed later by the `OnHCA`. Thus, this chapter, like the previous one, follows the approach to separate the determination of *base trajectories* and the circumvention, whereas the *base trajectories* here are determined online. The optimization problem (7.4) is a quadratic program (QP), which in general can be solved by sparse or dense formulation as described in [49]. With the assumption that the optimization problem is numerically well conditioned, such problems can be solved efficiently by state-of-the-art solvers.

As already mentioned, the overall goal state is defined by $x^0_{f|k}$, which is also the target state of the 0-th *base trajectory*. The target states $x^i_{f|k}$, $i \in \mathcal{M} \setminus 0$ of the other *base trajectories* are determined according to a specific method, which is described in the following.

Target States of the Base Trajectories

Definition 7.3. *Consider an odd number n_c of base trajectories. Let the target states $x^i_{f|k}$ be contained in a set $\mathcal{G}_{f|k}$:*

$$\mathcal{G}_{f|k} = \mathcal{X}^{stat}_{f|k} \cup \mathcal{X}^{var}_{f|k}, \tag{7.5}$$

with:

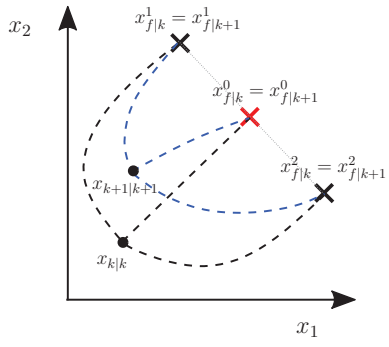
$$\mathcal{X}^{stat}_{f|k} := \{x^i_{f|k} |\ i \in \{0, ..., n_{stat}\}\}, \tag{7.6}$$

containing $n_{stat} = (n_c - 1)/2$ target states $x^i_{f|k}$, $i \in \{0, ..., n_{stat}\}$, and:

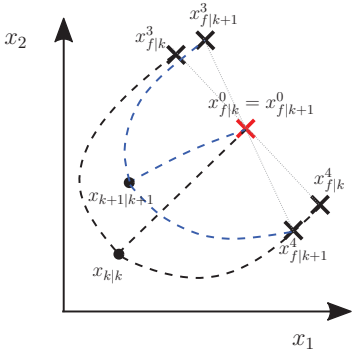
$$\mathcal{X}^{var}_{f|k} := \{x^i_{f|k} |\ i \in \{n_{stat} + 1, ..., n_c - 1\}\}, \tag{7.7}$$

with the target states $x^i_{f|k}$, $i \in \{n_{stat} + 1, ..., n_c\}$. The set $\mathcal{X}^{var}_{f|k}$ adapts its target states online, depending on the alignment between the current state $x_{k|k}$, and the overall target state $x^0_{f|k}$ at every point of time k. In contrast, the set $\mathcal{X}^{stat}_{f|k}$ only adapts its target states once when the goal state $x^0_{f|k}$ changes.

The reason of choosing target states adapted at each time step, and target states that adapt only if the goal state changes, is that existing variety of *base trajectories* allows for a better circumvention of the obstacle. With this strategy, states e.g. that are located immediately behind an obstacle, can be reached. The difference between both target sets is shown in Fig. 7.1. The figure shows the *base trajectories* determined for the target states at time k in black dashed lines, and the goal $x^0_{f|k}$. After execution of one time step, the new determined *base trajectories* towards the new target states are in blue dashed lines. The illustration shows the principle of the different target sets for the case where $n_c = 5$, and thus $n_{stat} = 2$. The left illustration (Fig. 7.1a) exemplary shows the target states of $\mathcal{X}^{stat}_{f|k} = \{x^0_{f|k}, x^1_{f|k}, x^2_{f|k}\}$



(a) Target states of $\mathcal{X}^{stat}_{f|k}$, and for the following time step $\mathcal{X}^{stat}_{f|k+1}$.



(b) Target states of $\mathcal{X}^{var}_{f|k}$, and for the following time step $\mathcal{X}^{var}_{f|k+1}$.

Figure 7.1.: Principle of how the two target sets $\mathcal{X}^{var}_{f|k}$, and $\mathcal{X}^{stat}_{f|k}$ change over time, when time advances from k to $k+1$. The black dashed lines are the *base trajectories* at time k, and the blue at time $k+1$.

for a constant goal state $x^0_{f|k}$ (red) over time. Starting from time k, the target states $x^1_{f|k}$ and $x^2_{f|k}$ are located orthogonal to the direction $(x_{k|k} - x^0_{f|k})$:

$$(x_{k|k} - x^0_{f|k}) \perp (x^i_{f|k} - x^0_{f|k}), \quad \forall x^i_{f|k} \in \mathcal{X}^{stat}_{f|k}. \tag{7.8}$$

However, at time $k+1$ orthogonality as in (7.8) is no longer given, since the target states of set $\mathcal{X}^{stat}_{f|k}$ are only adapted if the overall goal state $x^0_{f|k+1}$ also changes, hence if $x^0_{f|k+1} \neq x^0_{f|k}$. This is not the case in this illustration: In contrast, Fig. 7.1b shows that the target states of $\mathcal{X}^{var}_{f|k} = \{x^3_{f|k}, x^4_{f|k}\}$ are adapted at every time step such that it holds for all times k, that the target states are orthogonal to the direction $(x_{k|k} - x^0_{f|k})$:

$$(x_{k|k} - x^0_{f|k}) \perp (x^i_{f|k} - x^0_{f|k}), \quad \forall x^i_{f|k} \in \mathcal{X}^{var}_{f|k} \text{ and } \forall k \in \mathcal{J}_{\mathcal{N}}. \tag{7.9}$$

The strategy is used here in order to have a simple criterion for creating *base trajectories*. A reasonable choice of the distance between the target states $x^i_{f|k}$, $i \in \mathcal{M} \setminus 0$ from $x^0_{f|k}$ should be such that at least one *base trajectory* is not intersecting with the obstacle. Thus, the obstacle size plays an important role.

A target state $x^i_{f|k}$ of a *base trajectory*, e.g. for a mechanical system, commonly consisting of positions and velocities, with velocities represented as position differ-

ences, arranged in the form:

$$x^i_{f|k} = \begin{pmatrix} x^i_{1,f|k} \\ (x^i_{1,f|k} - x^i_{1,f-1|k})/\Delta T \\ x^i_{2,f|k} \\ (x^i_{2,f|k} - x^i_{2,f-1|k}))/\Delta T \\ \vdots \end{pmatrix} \in \mathbb{R}^{n_x},\ \forall i \in \mathcal{M} \setminus 0, \tag{7.10}$$

where $x^i_{1,f|k}$ denotes the first entry of state vector $x^i_{f|k}$. When considering a steady state of (7.10), the velocity entries become zero. The position entries of $x^i_{f|k}$ are summarized in a reduced state vector:

$$\tilde{x}^i_{f|k} = \begin{pmatrix} x^i_{1,f|k} \\ x^i_{2,f|k} \\ \vdots \end{pmatrix} \in \mathbb{R}^{n_{stat}}. \tag{7.11}$$

Thus, this reduced state vector specifies the dimension of *base trajectories* which are needed for a circumvention of the obstacle $\mathcal{P}_{x,k+j|k}$. While the time-varying goal state $x^0_{f|k}$ is known, the other states $x^i_{f|k} \in \mathcal{G}_{f|k} \setminus x^0_{f|k}$ are determined such that the reduced state vectors $\tilde{x}^i_{f|k}$ span a regular $(n_{stat}-1)$-simplex with the goal state $\tilde{x}^0_{f|k}$ as the center. For instance, a value of $n_{stat} = 3$ yields a 2-simplex, which is a triangle in a three-dimensional space, and $n_{stat} = 2$ leads to a 1-simplex (straight line) in a two-dimensional space, as was illustrated in Fig. 7.1. Hence a $(n_{stat}-1)$-simplex at time k is given as follows:

$$\triangle^{n_{stat}-1}_{f|k} = \{\tilde{x}_{f|k} \in \mathbb{R}^{n_{stat}} |\ \tilde{x}_{f|k} = \sum_{i=1}^{n_{stat}} t_i\ \tilde{x}^i_{f|k},\ 0 \le t_i \le 1,\ \sum_{i=1}^{n_{stat}} t_i = 1\}. \tag{7.12}$$

The simplex is obtained by a convex combination of vertices $\tilde{x}^i_{f|k}$. This vertices later correspond to the target states of the *base trajectories*. They are obtained by first generating a regular $(n_{stat}-1)$-simplex in the Cartesian plane and then orienting it orthogonal to the direction between the current Cartesian state $\tilde{x}_{k|k}$ and the goal state $\tilde{x}_{f|k}$. Finally the simplex is moved such that its center becomes $\tilde{x}_{f|k}$. Then the vertices/target states are again described in their full dimensional space $x^i_{f|k} \in \mathbb{R}^{n_x}$. The following three steps are carried out to obtain the target states of the *base trajectories*:

1. A regular $(n_{stat}-1)$-simplex with vertices $\tilde{v}^i \in \mathbb{R}^{n_{stat}}$, $i = \{1, ..., n_{stat}\}$ is created. The simplex has the center $\tilde{v}^0 = [0]^{n_{stat}}$. In general, a regular simplex has the following two properties:
   a) The distances between the vertices and the center are equal.
   b) The angle spanned by any two vertices of an $(n_{stat}-1)$-dimensional simplex through its center is $\arccos(-1/(n_{stat}-1))$, thus:

$$\triangle^{n_{stat}-1} = \{\tilde{v} \in \mathbb{R}^{n_{stat}} | \ \tilde{v} = \sum_{i=1}^{n_{stat}} t_i \ \tilde{v}^i \ d_{fac}, \ 0 \leq t_i \leq 1, \ \sum_{i=1}^{n_{stat}} t_i = 1\}, \quad (7.13)$$

The factor d_{fac} is used to either enlarge ($d_{fac} > 1$) or reduce ($d_{fac} < 1$) the area of the simplex. An example of a regular 2-simplex in a 3D-space is shown in Fig. 7.2 by the red triangle centered at the origin. The figure also shows the current reduced state vector $\tilde{x}_{k|k}$ in red and the goal state $\tilde{x}^0_{f|k}$, in blue.

2. In order to rearrange the regular simplex (7.13) according to the goal state $\tilde{x}^0_{f|k}$, the orientation of the simplex is first adjusted and then moved to the target state $\tilde{x}^0_{f|k}$. The orientation is changed by the rotation matrix $\tilde{P}_{f|k} \in \mathbb{R}^{n_{stat} \times n_{stat}}$, which is an orthonormal basis for the null space of the direction vector $\delta\tilde{x}^0_{f|k} \in \mathbb{R}^{n_{stat}}$. The null space of $\delta\tilde{x}^0_{f|k}$ is obtained by the singular value decomposition as described in Sec. 3.4. The direction vector $\delta\tilde{x}^0_{f|k}$ points from the current reduced dimensional state $\tilde{x}_{k|k}$ to the final state $\tilde{x}^0_{f|k}$:

$$\delta\tilde{x}^0_{k|k} = \frac{\tilde{x}^0_{f|k} - \tilde{x}_{k|k}}{||\tilde{x}^0_{f|k} - \tilde{x}_{k|k}||_2}, \quad (7.14)$$

and is shown in Fig. 7.2 by the dashed line connecting $\tilde{x}_{k|k}$ with $\tilde{x}^0_{f|k}$. With this orthonormal basis and the displacement to the goal state $\tilde{x}^0_{f|k}$, the vertices of the resulting simplex (7.12) can be calculated as follows:

$$\tilde{x}^i_{f|k} = \tilde{x}^0_{f|k} + \tilde{P}_{f|k} \ \tilde{v}^i, \ i \in \{1, ..., n_{stat}\}. \quad (7.15)$$

With these target states, the resulting simplex (7.12) finally gets its end position as shown in the example of Fig. 7.2 by the blue triangle.

3. To finally lift the reduced target states into their original dimension, one can extend the orthonormal basis to $P_{f|k} \in \mathbb{R}^{n_x \times n_x}$ as follows:

$$P_{f|k} = \begin{pmatrix} \tilde{p}(1,1)_{f|k} & 0 & \tilde{p}(1,2)_{f|k} & 0 & ... \\ 0 & 1 & 0 & 0 & ... \\ \tilde{p}(2,1)_{f|k} & 0 & \tilde{p}(2,2)_{f|k} & 0 & ... \\ 0 & 0 & 0 & 1 & ... \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \quad (7.16)$$

with entries $\tilde{p}(i,j)_{f|k}$ of matrix $\tilde{P}_{f|k}$. The same holds for the vertices $\tilde{v}^i$ that are lifted to $v^i \in \mathbb{R}^{n_x}$:

$$v^i = \begin{pmatrix} \tilde{v}^i_1 \\ 0 \\ \tilde{v}^i_2 \\ 0 \\ \tilde{v}^i_3 \\ \vdots \end{pmatrix}. \quad (7.17)$$

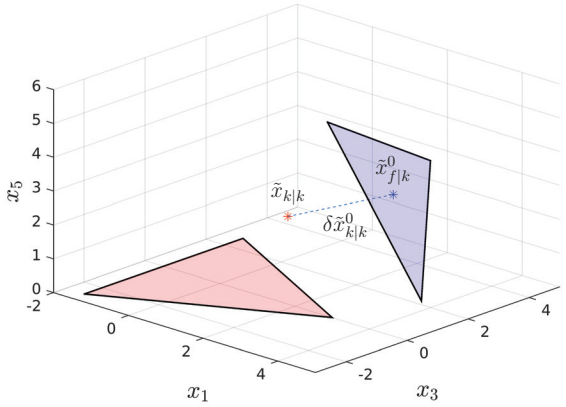


Figure 7.2.: Shows the regular simplex (red), reoriented according to the goal state $\tilde{x}^0_{f|k}$, and the directional vector $\delta\tilde{x}^0_{k|k}$ to obtain the resulting simplex (blue).

Algorithm 7.1.: Generation of Target States (`GenTarS`)

1: **Given:** $x^0_{f|k}$, $x^0_{f|k-1}$, $x_{k|k}$, n_c
2: **if** target state $x^0_{f|k} \neq x^0_{f|k-1}$ changed **then**
3: update $\mathcal{X}^{stat}_{f|k}$ according to (7.13)-(7.18)
4: **end if**
5: determine $\mathcal{X}^{var}_{f|k}$ according to (7.13)-(7.18)
6: $\mathcal{G}_{f|k} := \mathcal{X}^{stat}_{f|k} \cup \mathcal{X}^{var}_{f|k}$

Thus, the full-dimensional vertices $x^i_{f|k}$ are determined by:

$$x^i_{f|k} = x^0_{f|k} + P_{f|k}\, v^i,\ i \in \{1, ..., n_{stat}\}. \tag{7.18}$$

The obtained target states $x^i_{f|k}$, $i \in \{1, ..., n_{stat}\}$ are used to determine the *base trajectories*. For later use, the procedure described by the above three steps is formulated as Algorithm 7.1.

Determination of Base Trajectories

To obtain suitable *base trajectories* $\hat{x}^i_{:|k}$, $i \in \mathcal{M}$, determined from the current state $x_{k|k}$ towards their corresponding target state $x^i_{f|k} \in \mathcal{G}_{f|k}$, one has to perform a coordinate transformation in the optimization problem (7.4). The reason for this is

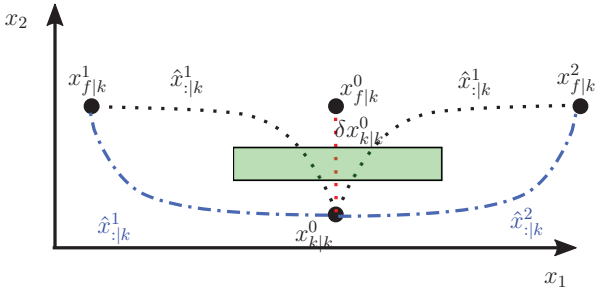


Figure 7.3.: Shows the black dotted *base trajectories* $\hat{x}^1_{:|k}$, $\hat{x}^2_{:|k}$ and the red dotted one to the final state $x^0_{f|k}$ by solving the original problem (7.4) with linearized dynamic, and the blue dashed *base trajectories* $\hat{x}^1_{:|k}$, $\hat{x}^2_{:|k}$ as the solutions of problem (7.22) with transformed coordinates. The green box is an obstacle.

shown in Fig. 7.3. The figure illustrates the obtained *base trajectories* towards the target states $x^0_{f|k}$, $x^1_{f|k}$, and $x^2_{f|k}$ by the black and red dotted lines as solutions of the optimization problem (7.4). If now the obstacle is taken into account (green box in Fig. 7.3), the possible space spanned by the *base trajectories* is not sufficiently large enough to select a collision-free homotopic target trajectory by `OnHCA`. If however, a coordinate transformation is performed by moving the origin of the coordinate system towards the goal state $x^0_{f|k}$ and re-orienting it according to the directional vector $\delta\tilde{x}^0_{k|k}$ of (7.14), one can obtain *base trajectories* which are better suited for collision avoidance, see the blue dashed lines of Fig. 7.3. By means of the orthonormal basis $P_{f|k}$ from (7.16), the transformation of a state $x_{k+j|k}$ into the new coordinate system with new states is denoted by $s_{k+j|k} \in \mathbb{R}^{n_x}$ and given by:

$$x_{k+j|k} = x^0_{f|k} + P_{f|k}\ s_{k+j|k}. \tag{7.19}$$

Inserting (7.19) into the linearized dynamics (7.1) yields the transformed dynamics:

$$s_{k+j+1|k} = (P_{f|k})^{-1}(A_{:|k}(x^0_{f|k} + P_{f|k}\ s_{k+j|k}) + B_{:|k}u_{k+j|k} + r_{:|k} - x^0_{f|k}). \tag{7.20}$$

Based on the optimization problem (7.4), the determination of the $i-$th *base trajectory* is then performed in the transformed coordinate system determined from the current state $s_{k|k}$ to a target state $v^i_{f|k}$:

$$v^i_{f|k} = (P_{f|k})^{-1}(x^i_{f|k} - x^0_{f|k}),\ \text{with}\ x^i_{f|k} \in \mathcal{G}_{f|k}, \tag{7.21}$$

by solving the QP:

$$\min_{s_{k+H|k}, u_{k+j|k}} \quad \|s_{k+j|k} - v^i_{f|k}\|^2_{Q_{Bend}} + \sum_{j=0}^{H-1} \|s_{k+j|k} - v^i_{f|k}\|^2_{Q_B} + \|u_{k+j|k} - u^i_{f|k}\|^2_{R_B} \tag{7.22a}$$

$$\text{s.t.} \quad (7.20),\ (7.21),\ u_{k+j|k} \in \mathcal{U},\ \forall j \in \mathcal{J}_{\mathcal{H}}. \tag{7.22b}$$

With the determined *base trajectories* $\hat{x}^i_{:|k}$ and $\hat{u}^i_{:|k}$ with $i \in \mathcal{M}$, the homotopic states $x_{k+j|k}(\boldsymbol{\lambda}_{:|k})$ and inputs $u_{k+j|k}(\boldsymbol{\lambda}_{:|k})$ are finally obtained for all $j \in \mathcal{J}_{\mathcal{H}}$ by the homotopy equations:

$$x_{k+j|k}(\boldsymbol{\lambda}_{:|k}) := x^0_{k|k} + D_{x,k+j|k}\boldsymbol{\lambda}_{:|k}, \tag{7.23}$$

$$u_{k+j|k}(\boldsymbol{\lambda}_{:|k}) := u^0_{k|k} + D_{u,k+j|k}\boldsymbol{\lambda}_{:|k}. \tag{7.24}$$

A homotopic trajectory is then parametrized by the constant value of $\boldsymbol{\lambda}_{:|k}$ based on the determination at time k.

7.3. Online Control for Nonlinear Dynamics

The online homotopic control algorithm `OnHCA` is based on the `HCA` algorithm developed in Sec. 6.3.3. While the transformation of the obstacle into the homotopy space, the identification of collision times, and the removal of redundant half-planes are the same, only the key points are revisited here and adapted to the receding horizon notation. In addition, compared to Chapter 6, $\mathfrak{r}$-step controllability of the system is no longer important for the `OnHCA`, since the *base trajectories* are determined at each time k from the current state $x_{k|k}$ and therefore all homotopic trajectories also start from the current state $x_{k|k} = x^0_{k|k} = x^1_{k|k} = ... = x^{n_c}_{k|k}$. This implies that the system is always on a homotopic trajectory rather than having to be controlled into one in $\mathfrak{r}$-steps, as in Chapter 6. Thus, the `OnHCA` is a completely feedforward method. For a more detailed explanation the reader is referred to Sec. 6.3.3. By assuming $\mathfrak{r} = 1$ the notation of the time lifted variables from the previous chapter become again the original notation:

$$\mathfrak{t} \to k \tag{7.25}$$

$$j_{\mathfrak{t}} \to j \tag{7.26}$$

$$\boldsymbol{\mu}_{\mathfrak{t}+j_{\mathfrak{t}}} \to \boldsymbol{\lambda}_{k+j} \tag{7.27}$$

$$z_{\mathfrak{t}+j_{\mathfrak{t}}} \to x_{k+j}. \tag{7.28}$$

7.3.1. Optimal Homotopic Trajectory without Obstacles

As described in Sec. 6.3.1, the `OnHCA` starts by determining an optimal homotopic trajectory encoded by $\boldsymbol{\lambda}^*_{:|k}$ over the prediction horizon $j \in \mathcal{J}_{\mathcal{H}}$. Therefore, the

cost function $J(x_{k+j|k}, u_{k+j|k})$ in (7.2) is reformulated into $J(\boldsymbol{\lambda}_{:|k})$, by replacing the states $x_{k+j|k}$ and inputs $u_{k+j|k}$ by the homotopy equations $x_{k+j|k}(\boldsymbol{\lambda}_{:|k})$ and $u_{k+j|k}(\boldsymbol{\lambda}_{:|k})$ according to (7.23) and (7.24). The homotopy equations are obtained as described in the previous section by first generating target states $x^i_{f|k} \in \mathcal{G}_{f|k}$, and then determining the *base trajectories* according to (7.22). By solving the following optimization problem, the best homotopic target trajectory $\boldsymbol{\lambda}^*_{:|k}$ from the current state $x_{k|k}$ to the goal state $x^0_{f|k}$ is considered without the obstacle:

`OnOHT` :

$$\begin{aligned}
\min_{\boldsymbol{\lambda}_{:|k}} \quad & \|x_{k+H|k}(\boldsymbol{\lambda}_{:|k}) - x^0_{f|k}\|^2_{Q_{end}} \\
& + \sum_{j=0}^{H-1} \|x_{k+j|k}(\boldsymbol{\lambda}_{:|k}) - x^0_{f|k}\|^2_Q + \|u_{k+j|k}(\boldsymbol{\lambda}_{:|k}) - u^0_{f|k}\|^2_R && \text{(7.29a)} \\
\text{s.t.} \quad & x_{k+j|k}(\boldsymbol{\lambda}_{:|k}) = x^0_{k+j|k} + D_{x,k+j|k}\boldsymbol{\lambda}_{:|k}, && \text{(7.29b)} \\
& u_{k+j|k}(\boldsymbol{\lambda}_{:|k}) = u^0_{k+j|k} + D_{u,k+j|k}\boldsymbol{\lambda}_{:|k}, \ \forall j \in \mathcal{J}_{\mathcal{H}} && \text{(7.29c)} \\
& \boldsymbol{\lambda}_{:|k} \in [0,1]^{n_c}, \ \sum_{i=1}^{n_c} \lambda^i_{:|k} \leq 1, && \text{(7.29d)}
\end{aligned}$$

with weights $Q_{end} \in \mathbb{S}^{n_x}_{>0}$, $Q \in \mathbb{S}^{n_x}_{>0}$ and $R \in \mathbb{S}^{n_u}_{>0}$. Since the input constraints were considered in the determination of the *base trajectories*, the input constraints are automatically satisfied. Thus, they also hold for all homotopic trajectories. The optimization problem (7.29) is used as a first cheep computational trial to determine a homotopic trajectory. But in the case of collisions with the obstacle $\mathcal{P}_{x,k+j|k}$, the obstacle has to be transformed into the homotopy space to identify possible collisions times for the `OnHCA`, see below.

7.3.2. Obstacle Transformation into the Homotopy Space

According to Sec. 6.3.2, the obstacle:

$$\mathcal{P}_{x,k+j|k} := \{x_{k+j|k} |\ C_{k+j|k}\ x_{k+j|k} \leq d_{k+j|k}\},\ j \in \mathcal{T}_{\mathcal{H}} \tag{7.30}$$

is transformed into the homotopy space similar to (6.25)-(6.27):

$$\mathcal{P}_{\boldsymbol{\lambda},k+j|k} := \{\boldsymbol{\lambda}_{k+j|k} | C_{\boldsymbol{\lambda},k+j|k}\ \boldsymbol{\lambda}_{k+j|k} \leq d_{\boldsymbol{\lambda},k+j|k}\}. \tag{7.31}$$

Then, it is checked whether an intersection of $\mathcal{P}_{x,k+j|k}$ and a homotopic state $x_{k+j|k}(\boldsymbol{\lambda}_{k+j|k})$ occurs at time $k+j$:

$$\mathcal{P}_{x,k+j|k} \cap x_{k+j|k}(\boldsymbol{\lambda}_{k+j|k}) \neq \emptyset. \tag{7.32}$$

If, however, the intersection is empty at any time $k+j$, then no collision between a homotopic state and the obstacle can occur for this time steps. Similar to (6.30),

a successive check of the intersection (7.32) yields the set $\mathcal{T}_{\mathcal{I}}$, containing all intersection times for which (7.32) is nonempty.

$$\mathcal{T}_{\mathcal{I}} := \{j|\ (7.32) \text{ is nonempty}\} \subseteq \mathcal{T}_{\mathcal{H}}. \tag{7.33}$$

The obtained intersection times are used later in `OnHCA` for collision avoidance.

When mapping the polytope from the higher dimensional state space into the lower homotopy space, redundant half-planes may occur. These cause an unnecessary overapproximation of the homotopy obstacle $\mathcal{P}_{\boldsymbol{\lambda},k+j|k}$ and thus should be removed. This reduction (see Sec. 6.3.2), finally leads to a shrinked obstacles $\tilde{\mathcal{P}}_{\boldsymbol{\lambda},k+j|k}$ in the homotopy space:

$$\tilde{\mathcal{P}}_{\boldsymbol{\lambda},k+j|k} := \{\boldsymbol{\lambda}_{k+j|k}|\tilde{C}_{\boldsymbol{\lambda},k+j|k}\ \boldsymbol{\lambda}_{k+j|k} \leq \tilde{d}_{\boldsymbol{\lambda},k+j|k}\}. \tag{7.34}$$

With respect to Def. 6.2, the w-th half-plane of $\tilde{\mathcal{P}}_{\boldsymbol{\lambda},k+j|k}$ is given by $\tilde{C}^w_{\boldsymbol{\lambda},k+j|k}$, and $\tilde{d}^w_{\boldsymbol{\lambda},k+j|k}$, for $w \in \tilde{\mathcal{W}}_{k+j|k}$.

7.3.3. Online Homotopic Control Algorithm

The online homotopic control algorithm `OnHCA`, as shown in Alg. 7.3, differs from `HCA` in determining the *base trajectories* online according to the procedure explained in Sec. 7.2. In each k, the following steps are executed to obtain a circumventing, close to optimal trajectory:

1. Linearization of the system dynamics,
2. Determination of the optimal trajectory without obstacle,
3. Determination of *base trajectories*,
4. Computation of a fallback trajectory,
5. Executing a tree search for circumvention.

Each step is now explained.

Step 1: Linearization of the System Dynamics (line 2 of `OnHCA`)

The `OnHCA` starts from the current state $x_{k|k}$ by linearization and time discretization of the nonlinear dynamics to obtain the system dynamics (7.1).

Step 2: Determination of the Optimal Trajectory without Obstacle (line 3 of `OnHCA`)

According to the linearized model, the optimal trajectory from $x_{k|k}$ to the goal state $x^0_{f|k}$ is determined by solving problem (7.4). This step provides a first computationally cheap way to obtain a homotopic target trajectory $\hat{x}_{:|k}(\boldsymbol{\lambda}^*_{:|k}) := \{x_{k|k}(\boldsymbol{\lambda}^*_{:|k}),$

$x_{k+1|k}(\boldsymbol{\lambda}^*_{:|k}), ..., \ x_{k+H|k}(\boldsymbol{\lambda}^*_{:|k})\}$ towards the goal state. If the trajectory is checked to be free of collisions over the prediction horizon (*line 5*), i.e. $x_{k+j|k}(\boldsymbol{\lambda}^*_{:|k}) \notin \mathcal{P}_{x,k+j|k}$ for all $j \in \mathcal{J}_{\mathcal{H}}$, the algorithm stops and the nonlinear system executes one step of the determined solution (*line 14*). After execution, the time is incremented $k := k + 1$ and the execution of `OnHCA` is repeated with a forward-shifted prediction horizon. In case of collisions, the circumventing procedure is initiated and the following steps are executed.

Step 3: Determination of *Base Trajectories* (line 6 of `OnHCA`)

The algorithm `GenTarS` from Alg. 7.1 is executed to obtain the set of target states $\mathcal{G}_{f|k}$ and the corresponding *base trajectories* $\hat{x}^i_{:|k}$, $i \in \mathcal{M}$, for the problem (7.22) with transformed coordinates.

Step 4: Computation of a Fallback Trajectory (line 11 of `OnHCA`)

The best among all *base trajectories*, encoded by $\boldsymbol{\lambda}^{i^*}_{:|k}$, is finally selected and serves as a fallback solution. This trajectory is obtained by evaluating the costs $J(\boldsymbol{\lambda}^i_{:|k})$ for each *base trajectory* $\hat{x}^i_{:|k}$:

$$J(\boldsymbol{\lambda}^i_{:|k}) = \|x_{k+j|k}(\boldsymbol{\lambda}^i_{:|k}) - x^0_{f|k}\|^2_{Q_{end}} + \sum_{j=0}^{H-1} \|x_{k+j|k}(\boldsymbol{\lambda}^i_{:|k}) - x^0_{f|k}\|^2_Q + \|u_{k+j|k}(\boldsymbol{\lambda}^i_{:|k}) - u^0_{f|k}\|^2_R \tag{7.35a}$$

$$\text{with: } (7.23), \ (7.24), i \in \mathcal{M}. \tag{7.35b}$$

As already described in Sec. 6.3.3, the obtained best *base trajectory* $\boldsymbol{\lambda}^{i^*}_{:|k}$ is obtained by the tree search procedure of *Step. 5*.

Step 5: Executing a Tree Search for Circumvention (line 12 of `OnHCA`)

As in Sec. 6.3.3, the tree search selects an intersection time $j \in \mathcal{T}_{\mathcal{I}}$ and determines the best homotopic target trajectory $\boldsymbol{\lambda}^*_{:|k}$ for a selected half-plane $w \in \tilde{\mathcal{W}}_{k+j|k}$. The obstacle is then circumvented by this half-plane. The search method of selecting the half-plane is based on a tree. Each $j \in \mathcal{T}_{\mathcal{I}}$ refers to a level of the tree, while the number of decisions on a tree levels is given by the number of half-planes $|\tilde{\mathcal{W}}_{k+j|k}|$. According to the tree structure (see Fig. 6.5 for illustration), a node on the tree level j, with half-plane w contains the following information:

$$node_{j,w} = (j, w, \boldsymbol{\lambda}^*_{:|k}, J(\boldsymbol{\lambda}^*_{:|k}), \hat{x}_{:|k}(\boldsymbol{\lambda}^*_{:|k}), \mathcal{O}), \tag{7.36}$$

where $J(\boldsymbol{\lambda}^*_{:|k})$ is the cost of the homotopic target trajectory determined in this node and $\hat{x}_{:|k}(\boldsymbol{\lambda}^*_{:|k})$ is the homotopic target trajectory. All nodes of the tree level j are

denoted by the set:

$$Node_j := \{node_{j,1}, ..., node_{j,|\tilde{\mathcal{W}}_{k+j|k}|}\}. \tag{7.37}$$

The set $\mathcal{O}$ again contains the best node, $node^*_{j,w}$ on level j with minimal costs $J(\boldsymbol{\lambda}^*_{:|k})$, and:

$$\mathcal{O} := \mathcal{O} \cup \{node^*_{j,w}\}. \tag{7.38}$$

The node, $node^*_{j,w}$ is obtained from (6.43). The function $TimePlane$, given in (6.37), again extracts the time j and the half-plane w of node $node_{j,w}$ and assigns these values to the variables s and l:

$$(s, l) = TimePlane(node_{j,w}). \tag{7.39}$$

Thus, an optimization problem can be set up which determines the best homotopic target trajectory $\boldsymbol{\lambda}^*_{:|k}$ for a selected node $node_{j,w}$. The problem is based on the problem `OnOHT` with the extension to obstacle avoidance:

`OnOHTadd` :

$$\text{select a node, } node_{j,w} \text{ and solve} \tag{7.40a}$$

$$\texttt{OnOHT} \tag{7.40b}$$

additionally s.t.

$$\tilde{C}^w_{\boldsymbol{\lambda}_{k+j|k}} \boldsymbol{\lambda}^{i^*}_{:|k} \geq \tilde{d}^w_{\boldsymbol{\lambda}_{k+j|k}} \quad \left.\right\} \text{ ensures that the trajectory from Step 4 is always contained in the solution space} \tag{7.40c}$$

$$\tilde{C}^w_{\boldsymbol{\lambda}_{k+j|k}} \boldsymbol{\lambda}_{:|k} \geq \tilde{d}^w_{\boldsymbol{\lambda}_{k+j|k}} \quad \left.\right\} \text{ collision avoidance constraint in time step } j \tag{7.40d}$$

$$\left.\begin{array}{l} \tilde{C}^l_{\boldsymbol{\lambda}_{k+s|k}} \boldsymbol{\lambda}_{:|k} \geq \tilde{d}^l_{\boldsymbol{\lambda}_{k+s|k}}, \\ (s, l) = TimePlane(node_{j,w}),\ \forall node_{j,w} \in \mathcal{O} \end{array}\right\} \text{ consideration of the trees sequence so far} \tag{7.40e}$$

The constraint (7.40c) guarantees that the fallback trajectory $\boldsymbol{\lambda}^{i^*}_{:|k}$ from *Step 4* is always contained in the solution space. If this is not true, the optimization problem would not be feasible and therefore the selected half-plane would not be suitable for circumvention. The second inequality (7.40d) now contains the optimization variable $\boldsymbol{\lambda}_{:|k}$ and forces a homotopic target trajectory to be selected which is outside of the selected half-plane w of $\tilde{\mathcal{P}}_{\boldsymbol{\lambda},k+j|k}$. Finally, the inequalities in (7.40e) have the same functionality as (7.40d), but consider the half-planes l and time-steps s that have already been identified in the trees sequence so far, stored in the set $\mathcal{O}$.

It is again pointed out, that `OnBfSearch` terminates immediately when a collision-free solution is found during the tree search, see *line 9*. Of course, this can occur before the set $\mathcal{T}_\mathcal{I}$ becomes empty, establishing another advantage of this procedure.

Homotopic Control Algorithm with Online Determined Base Trajectories `OnHCA`

The online homotopic control algorithm `OnHCA` is based on *Step 1* - *Step 5* as described above and is summarized in Alg. 7.3. The algorithm is executed in each

Algorithm 7.2.: Best-first Search of a Homotopic Target Trajectory with Online *Base Trajectories* (`OnBfSearch`)

1: **Given:** $\mathcal{T}_\mathcal{I}$, initial time $k := k_0$, fallback trajectory $\boldsymbol{\lambda}^{i^*}_{:|k}$, $\tilde{\mathcal{P}}_{\boldsymbol{\lambda},k+j|k}$, $\mathcal{O} = \emptyset$
2: **while** $\mathcal{T}_\mathcal{I} \neq \emptyset$ **do**
3: select $j \in \mathcal{T}_\mathcal{I}$
4: determine all nodes $node_{j,w}$, with $w \in \tilde{\mathcal{W}}_{k+j|k}$ by solving
5: `OnOHTadd` according to (7.40)
6: $Node_j := \{node_{j,1}, ..., node_{j,|\tilde{\mathcal{W}}_{k+j|k}|}\}$
7: select best node according to $node^*_{j,w} = Bestnode(Node_j)$
8: $\mathcal{O} := \mathcal{O} \cup node^*_{j,w}$
9: **if** $\exists j \in \mathcal{T}_\mathcal{H} : (x_{k+j|k}(\boldsymbol{\lambda}^*_{:|k})$ from $node^*_{j_\mathfrak{i},w}) \in \mathcal{P}_{x,k+j|k}$ **then**
10: $\mathcal{T}_\mathcal{I} := \mathcal{T}_\mathcal{I} \setminus j$
11: **else**
12: exit while loop
13: **end if**
14: **end while**
15: **Return**$(\boldsymbol{\lambda}^*_{:|k})$

k. Starting from a current state $x_{k|k}$ and the input from the previous iteration $u_{k|k-1}$, the system is first linearized (*line 2*). Then an optimal homotopic target trajectory $\hat{x}_{:|k}(\boldsymbol{\lambda}^*_{:|k})$ is determined, neglecting the obstacle (*line 3*). If this trajectory is free of collision, the algorithm executes the determined input value for the nonlinear system (*line 14*) and the `OnHCA` is again executed at time $k := k+1$. If however, a collision is detected (*line 5*), the algorithm proceeds by generating the set of target states $\mathcal{G}_{f|k}$ for which *base trajectories* are determined (*line 6*). By means of an obstacle transformation into the homotopy space, collision times between homotopic states and the transformed obstacle $\tilde{\mathcal{P}}_{\boldsymbol{\lambda},k+j|k}$ can be identified (*line 6-7*). The collision times are finally used in the tree search procedure (*line 12*), which also requires information on a fallback trajectory $\boldsymbol{\lambda}^{i^*}_{:|k}$ (*line 11*). The result of the tree-search is an optimal homotopic target trajectory $\hat{x}_{:|k}(\boldsymbol{\lambda}^*_{:|k})$ circumventing the obstacle (*line 14*).

Algorithm 7.3.: Online Homotopic Control Algorithm (`OnHCA`)

1: **Given:** current state $x_{k|k}$ and input $u_{k|k-1}$, $\mathcal{J}_{\mathcal{H}}$, nonlinear dynamics (3.1), $\mathcal{P}_{x,k+j|k}$, $x^0_{f|k}$, $x^0_{f|k-1}$, n_c, $\mathcal{O} := \emptyset$
2: linearize and discretize (3.1) for the current state $x_{k|k}$ and input $u_{k|k-1}$ (*Step 1*)
3: compute the optimal homotopic target trajectory $\hat{x}_{:|k}(\boldsymbol{\lambda}^*_{:|k})$ without
4: considering the obstacle (*Step 2*)
5: **if** $\exists j \in \mathcal{J}_{\mathcal{H}} : x_{k+j|k}(\boldsymbol{\lambda}^*_{:|k}) \in \mathcal{P}_{x,k+j|k}$ **then**
6: determine $\mathcal{G}_{f|k}$ by executing `GenTarS`, and determine the
7: corresponding *base trajectories* according to (7.22) (*Step 3*)
8: map the time-varying obstacle into the homotopy space $\mathcal{P}_{x,k+j|k} \rightarrow \tilde{\mathcal{P}}_{\boldsymbol{\lambda},k+j|k}$,
9: as given in Sec. 7.3.2
10: determine intersection times $\mathcal{T}_{\mathcal{I}}$ according to (7.32)
11: determine collision-free fallback trajectory $\boldsymbol{\lambda}^{i^*}_{:|k}$ (*Step 4*)
12: execute the tree search algorithm `OnBfSearch` $\rightarrow \boldsymbol{\lambda}^*_{:|k}$ (*Step 5*)
13: **end if**
14: apply $u_{k|k}(\boldsymbol{\lambda}^*_{:|k})$ to (3.1)
15: $k := k + 1$
16: **goto** line 1.

7.4. Numerical Example

In this section, `OnHCA` is applied to a real world physical model, namely to a robotic manipulator as commonly used in industrial production. The task of `OnHCA` is to solve the problem of moving the manipulator from an initial position to a time-varying target position, while avoiding collisions with an obstacle moving in the same workspace. The goal position is time-varying and the manipulator dynamics is nonlinear. In addition, the problem arises, that the manipulator dynamics is described by a model in configuration space, hence in the space of joint angles and velocities, while on the other side, the obstacle exists in the Cartesian space. In general, it is known that the collision avoidance problem for robotic manipulators is very challenging. Especially when real time capability is of importance. The subsequent sections now show the modeling of the robotic manipulator, followed by simulation results of `OnHCA` for the obstacle avoidance scenario.

7.4.1. Model of a Robotic Manipulator

In this part, a simple model of a 2-D robotic manipulator is introduced, and the steps of linearization and transformation are described. The obtained approximation is then used by the `OnHCA` to provide an optimized solution of the collision avoidance problem in real-time.

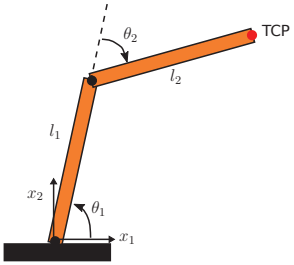


Figure 7.4.: Illustration of a robotic manipulator with two links and joints. The robot is operating in a two-dimensional cartesian space $x = (x_1, x_2)$ under gravitational forces.

The considered model is a robotic manipulator with two joints and two links, existing in a 2-D Cartesian space $x(t) = (x_1(t), x_2(t))^T \in \mathbb{R}^2$, see Fig. 7.4. The manipulator configuration is described by the vector of angles $\theta = (\theta_1(t), \theta_2(t))^T \in \mathbb{R}^2$, and the angular velocities $\dot{\theta}(t) = (\dot{\theta}_1(t), \dot{\theta}_2(t))^T \in \mathbb{R}^2$ over time t. The combined vector is denoted by:

$$\Theta(t) = (\theta_1(t), \theta_2(t), \dot{\theta}_1(t), \dot{\theta}_2(t))^T. \tag{7.41}$$

The two robotic links have the lengths l_1 and l_2. For simplification, the masses m_1 and m_2 are assumed to be mass points which are centered at the end of the first and second links, hence on positions $r_i(t) \in \mathbb{R}^2$, $i = \{1, 2\}$. The end effector position $r_2(t)$, also known as the tool center point (TCP), equals the position of mass m_2. Furthermore, friction is modeled as viscous friction with the coefficients c_1 and c_2, and the gravitational constant is denoted by g. The torques applied to the joints are $u(t) = (\tau_1(t), \tau_2(t))^T \in \mathbb{R}^2$.

Since `OnHCA` works with a linear discrete-time model, model simplification is necessary. An overview of the modeling process is shown in Fig. 7.5, based on which a dynamic model of the manipulator is derived. First (right part of Fig. 7.5), the model describes the motion of the robot depending on the vector of joint configurations, velocities $\Theta(t)$, and inputs $u(t)$ as nonlinear differential equation. The next step is to linearize and discretize the model in time. Parallel to this, the nonlinear forward kinematics is established, shown in the left path of Fig. 7.5. The forward kinematics computes the Cartesian position of a desired point on the robot (e.g. the TCP) from values of the joint parameters. Differentiation with respect to time of the forward kinematics then leads to the velocities of the desired point (TCP) in Cartesian space. These nonlinear equations are also linearized. Finally, the linear model approximation (right path of Fig. 7.5) and the linearized forward kinematics (left path of Fig. 7.5) are combined, such that a linear dynamic model is obtained,

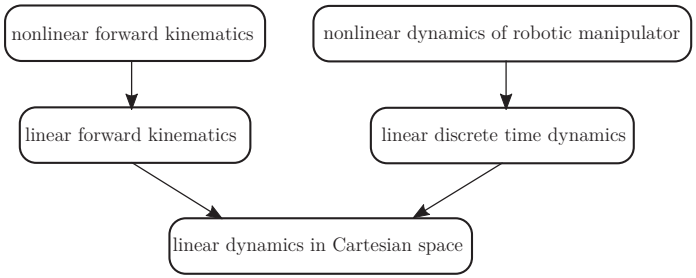


Figure 7.5.: Model transformation scheme with steps carried out for the manipulator dynamics (right side), and transformation from configuration to Cartesian space (left side). Both sides are linearized and combined to a model in the Cartesian space.

describing the position and velocity of a desired point in the Cartesian space. Obviously, a model which is based on two linearizations can lead to imprecise preditions. Therefore, this procedure is executed at every time k to obtain a best as possible model behavior for prediction.

Modeling the Robot Dynamics

By means of Lagrangian mechanics, the nonlinear differential equation can be constructed by the equation:

$$\frac{\partial L(t)}{\partial \theta_i(t)} - \frac{d}{dt}\frac{\partial L(t)}{\partial \dot{\theta}_i(t)} = -\tau_i, \quad i = \{1, 2\}, \tag{7.42}$$

where the Lagrangian $L(t)$ for the system is:

$$L(t) = T_{kin}(t) - V_{pot}(t), \tag{7.43}$$

with $T_{kin}(t)$ as the sum of the kinetic energy of each center of mass:

$$T_{kin}(t) = \sum_{i=1}^{2} \frac{1}{2} m_i ||\dot{r}_i(t)||^2, \tag{7.44}$$

and $V_{pot}(t)$ the potential energy:

$$V_{pot}(t) = \sum_{i=1}^{2} \frac{1}{2} m_i g \dot{r}_{i,2}(t)^2. \tag{7.45}$$

To determine the two energy equations, the positions $r_i(t) \in \mathbb{R}^2$ of each mass point m_i are modeled as functions of the generalized coordinates $\theta_i(t)$ by means of the

Denavit-Hartenberg convention [106]. Using this, each link can be described by a coordinate transformation from the coordinate system of mass m_i originated at i to the previous at $i-1$ (mass m_{i-1}) by a sequence of rotations and translations:

$$^{i-1}T_i = Rot_{i-1}(x_3, \theta_i(t)) \cdot Trans_i(x_1, l_i). \tag{7.46}$$

The transformation is performed by a rotation around the x_3-axis of the coordinate system $i-1$, followed by a translation along the new obtained x_1-axis of coordinate system i. A rotation around the x_3-axis of the current coordinate system $i-1$, with angle $\theta_i(t)$ is given by:

$$Rot_{i-1}(x_3, \theta_i(t)) = \begin{pmatrix} \cos\,\theta_i(t) & -\sin\theta_i(t) & 0 & 0 \\ \sin\,\theta_i(t) & \cos\,\theta_i(t) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{7.47}$$

and a translation along the x_1-axis of the coordinate system i with length l_i follows from:

$$Trans_i(x_1, l_i) = \begin{pmatrix} 1 & 0 & 0 & l_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{7.48}$$

The position $r_1(t)$ of mass m_1 in the global coordinate system is obtained from the matrix:

$$\begin{aligned} ^0T_1 &= Rot_0(x_3, \theta_1(t)) \cdot Trans_1(x_1, l_1) \\ &= \left(\begin{array}{ccc|c} & & & l_1\cos\theta_1(t) \\ & * & & l_1\sin\theta_1(t) \\ & & & 0 \\ \hline 0 & 0 & 1 & 1 \end{array}\right) \end{aligned} \tag{7.49}$$

The upper right part of $^0T_1$ shows the Cartesian coordinates depending on the joint angle $\theta_1(t)$:

$$r_1(\theta(t)) = \begin{pmatrix} l_1\cos\theta_1(t) \\ l_1\sin\theta_1(t) \end{pmatrix}. \tag{7.50}$$

The position $r_2(\theta(t))$ of mass m_2 is determined by a concatenation of rotations and translations. First, a rotation and translation around joint $\theta_1(t)$ is performed, which is then followed by the same procedure around the second joint $\theta_2(t)$. The transformation matrix from $r_2(\theta(t))$ into the Cartesian coordinate system is given

by:

$$
\begin{aligned}
{}^0T_2 &= Rot_0(x_3, \theta_1(t)) \cdot Trans_1(x_1, l_1) \cdot Rot_1(x_3, \theta_2(t)) \cdot Trans_2(x_1, l_2) \\
&= \left(\begin{array}{ccc|c} & * & & \begin{array}{c} l_1 \cos\theta_1(t) + l_2 \cos(\theta_1(t) + \theta_2(t)) \\ l_1 \sin\theta_1(t) + l_2 \sin(\theta_1(t) + \theta_2(t)) \\ 0 \end{array} \\ \hline 0 & 0 & 1 & 1 \end{array}\right),
\end{aligned} \tag{7.51}
$$

leading to the Cartesian position of the tool center point $TCP = r_2(t)$:

$$
TCP = r_2(\theta(t)) = \begin{pmatrix} l_1 \cos\theta_1(t) + l_2 \cos(\theta_1(t) + \theta_2(t)) \\ l_1 \sin\theta_1(t) + l_2 \sin(\theta_1(t) + \theta_2(t)) \end{pmatrix}. \tag{7.52}
$$

With these equations, the determination of the Lagrange mechanics (7.42) yields a system of two differential equations, each of order two, describing the manipulator dynamics:

$$
\begin{aligned}
\tau_1(t) - c_1\dot{\theta}_1 = &\left(l_1^2 m_1 + l_1^2 m_2 + l_2^2 m_2 + 2 l_1 l_2 m_2 \cos\theta_2(t)\right) \ddot{\theta}_1(t) \\
&+ \left(l_2^2 m_2 + l_1 l_2 m_2 \cos\theta_2(t)\right) \ddot{\theta}_2(t) \\
&+ g\left(l_2 m_2 \cos(\theta_1(t) + \theta_2(t)) + l_1 m_1 \cos\theta_1(t) + l_1 m_2 \cos\theta_1(t)\right)
\end{aligned} \tag{7.53}
$$

$$
\begin{aligned}
\tau_2(t) - c_2\dot{\theta}_2 = &\left(l_2^2 m_2 + l_2 m_2 l_1 \cos\theta_2(t)\right) \ddot{\theta}_1(t) + l_2^2 m_2 \ddot{\theta}_2(t) \\
&+ l_2 m_2 l_1 \sin\theta_2(t)\, \dot{\theta}_1(t)\dot{\theta}_1(t) + l_2 m_2 \dot{\theta}_2(t) l_1 \sin\theta_2(t)\, \dot{\theta}_1(t) \\
&+ g l_2 m_2 \cos(\theta_1(t) + \theta_2(t)).
\end{aligned} \tag{7.54}
$$

The two differential equations (7.53) and (7.54) can be represented by the following form:

$$
M(\theta(t))\ddot{\theta}(t) + C(\theta(t), \dot{\theta}(t))\dot{\theta}(t) + G(\theta(t)) - u(t) = 0, \tag{7.55}
$$

with $M(\theta(t))$ representing the inertial forces due to acceleration of the joints, $C(\theta(t), \dot{\theta}(t))$ modeling the Coriolis and centrifugal forces, $G(\theta(t))$ the gravitational forces and $u(t) = (\tau_1(t), \tau_1(t))^T \in \mathbb{R}^2$ the applied torques. The differential equation (7.55) is subsequently modeled as a system of ordinary differential equations (ODEs), with the combined state vector:

$$
\Theta(t) = \left(\theta_1(t), \theta_2(t), \dot{\theta}_1(t), \dot{\theta}_2(t)\right)^T, \tag{7.56}
$$

according to:

$$
\dot{\Theta}(t) = \begin{pmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ \ddot{\theta}_1(t) \\ \ddot{\theta}_2(t) \end{pmatrix} = \begin{pmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \\ (M(\theta_1(t), \theta_2(t)))^{-1} \left(u - C(\theta(t), \dot{\theta}(t)) \begin{pmatrix} \dot{\theta}_1(t) \\ \dot{\theta}_2(t) \end{pmatrix} - G(\theta_1(t), \theta_2(t))\right) \end{pmatrix}. \tag{7.57}
$$

Linearization of this dynamics (7.57) by a first order Taylor series expansion around the current state vector $\bar{\Theta}$ and the current input vector $\bar{u}$, followed by a zero-order hold (ZOH) discretization with step time ΔT leads to the linearized discrete-time dynamics:

$$\Theta_{k+j+1|k} = A_{\Theta,:|k}\Theta_{k+j|k} + B_{\Theta,:|k}u_{k+j|k} + \underbrace{f(\bar{\Theta},\bar{u}) - A_{\Theta,:|k}\bar{\Theta} - B_{\Theta,:|k}\bar{u}}_{r_{\Theta,:|k}}. \tag{7.58}$$

Here, $A_{\Theta,:|k}$, $B_{\Theta,:|k}$, and $r_{\Theta,:|k}$ are the matrices of the linear dynamics in the configuration space denoted by the index Θ, determined at time k. Index Θ is needed to distinguish the dynamics in the configuration space from the later obtained dynamics in the Cartesian space. Constancy over the prediction time is denoted by $:|k$, see Def. 7.1.

Transforming the Robot Dynamics from the Configuration into the Cartesian Space

Since the linearized dynamic (7.58) describes the dynamic behavior of the robotic manipulator in the configuration space, but the obstacle $\mathcal{P}_{x,k+j|k}$ is described in the Cartesian space, two options exist to bring the system and the obstacle into the same space:

1. Using forward kinematics by means of the Denavit-Hartenberg convention to describe a certain point on the manipulator (e.g. the TCP) according to its Cartesian states $x(t)$ and velocities $\dot{x}(t)$.

2. Mapping the obstacle into the configuration space.

From [89] it is known that the nonlinear mapping of the obstacle into the configuration space is very time demanding even without consideration of predicted obstacle positions. The non-convex shapes of the configuration space obstacle do not allow for an intuitive planning procedure. In the presented homotopy control method, this does prevent an intuitive parametrization of the *base trajectories*. Therefore, rather then mapping the obstacle into the configuration space, the linearized dynamics (7.58) are mapped into the Cartesian space by forward kinematics. However, it must be said that the forward kinematics only considers the TCP in this example, limiting the obstacle avoidance problem to the TCP. To start with, a consideration of collisions between the manipulator links and the obstacle are not considered at this point but will be treated in the later chapter.

To determine the TCP position $x(t) \in \mathbb{R}^2$ in the Cartesian space, the forward kinematics is given by the nonlinear function (7.52):

$$x(t) = r_2(\theta(t)) \tag{7.59}$$

and the TCP velocity $\dot{x}(t) \in \mathbb{R}^2$ by the time derivative of (7.52):

$$\dot{x}(t) = \frac{\partial\, r_2(\theta(t))}{\partial\, t}. \tag{7.60}$$

The combination of (7.59) and (7.60) into one state vector $\Psi(t) = (x(t),\ \dot{x}(t))^T \in \mathbb{R}^4$ leads to:

$$\Psi(t) = \begin{pmatrix} r_2(\theta(t)) \\ \\ \frac{\partial\, r_2(\theta(t))}{\partial\, t} \end{pmatrix} = \begin{pmatrix} l_1 \cos\theta_1(t) + l_2 \cos(\theta_1(t) + \theta_2(t)) \\ l_1 \sin\theta_1(t) + l_2 \sin(\theta_1(t) + \theta_2(t)) \\ -l_1 \sin\theta_1(t)\dot{\theta}_1(t) - l_2 \sin(\theta_1(t) + \theta_2(t))(\dot{\theta}_1(t) + \dot{\theta}_2(t)) \\ l_1 \cos\theta_1(t)\dot{\theta}_1(t) + l_2 \cos(\theta_1(t) + \theta_2(t))(\dot{\theta}_1(t) + \dot{\theta}_2(t)) \end{pmatrix}. \tag{7.61}$$

For the sake of clarity, (7.61) is referred to by a nonlinear function $\Xi : \mathbb{R}^4 \to \mathbb{R}^4$, of the joint vector $\Theta(t)$ as follows:

$$\Psi(t) = \Xi(\Theta(t)). \tag{7.62}$$

By evaluating (7.62) for a discrete point of time, one can write:

$$\Psi_{k+j|k} = \Xi(\Theta_{k+j|k}). \tag{7.63}$$

Equivalent to the nonlinear robot dynamics (7.57), the nonlinear forward kinematics (7.63) is linearized at the same point $\bar{\Theta}$:

$$\Psi_{k+j|k} = \underbrace{\left.\frac{\partial\, \Xi(\Theta_{k+j|k})}{\partial\, \Theta_{k+j|k}}\right|_{\bar{\Theta}}}_{\Phi_{:|k}} \Theta_{k+j|k} + \underbrace{\Xi(\bar{\Theta}) - \left.\frac{\partial\, \Xi(\Theta_{k+j|k})}{\partial\, \Theta_{k+j|k}}\right|_{\bar{\Theta}} \bar{\Theta}}_{\Omega_{:|k}}. \tag{7.64}$$

With this linearization, the Cartesian states $\Psi_{k+j|k}$ are determined from the configuration vector $\Theta_{k+j|k}$, with matrices $\Phi_{:|k}$, and $\Omega_{:|k}$. Solving (7.64) for $\Theta_{k+j|k}$ yields:

$$\Theta_{k+j|k} = (\Phi_{:|k})^{-1}\Psi_{k+j|k} - (\Phi_{:|k})^{-1}\Omega_{:|k}. \tag{7.65}$$

The inverse matrix $(\Phi_{:|k})^{-1}$, obtained from linearization, always exists when no singularity in the robots joint configurations is reached. Singularity results for example when all joint configurations reach zero. Then $(\Phi_{:|k})^{-1}$ has no more full rank and can not be inverted. Now, replacing $\Theta_{k+j|k}$ of the linearized robot dynamics (7.58), by (7.65), leads to:

$$\begin{aligned}(\Phi_{:|k})^{-1}\Psi_{k+j+1|k} - (\Phi_{:|k})^{-1}\Omega_{:|k} =& A_{\Theta,:|k}\left((\Phi_{:|k})^{-1}\Psi_{k+j|k} - (\Phi_{:|k})^{-1}\Omega_{:|k}\right) \\ &+ B_{\Theta,:|k}u_{k+j|k} + r_{\Theta,:|k},\end{aligned} \tag{7.66a}$$

which, when solved according to $\Psi_{k+j+1|k}$, becomes:

$$\begin{aligned}(\Phi_{:|k})^{-1}\Psi_{k+j+1|k} =& A_{\Theta,:|k}\left((\Phi_{:|k})^{-1}\Psi_{k+j|k} - (\Phi_{:|k})^{-1}\Omega_{:|k}\right) \\ &+ B_{\Theta,:|k}u_{k+j|k} + r_{\Theta,:|k} + (\Phi_{:|k})^{-1}\Omega_{:|k} \qquad (7.67a)\\ \Psi_{k+j+1|k} =& \Phi_{:|k}A_{\Theta,:|k}(\Phi_{:|k})^{-1}\Psi_{k+j|k} - \Phi_{:|k}A_{\Theta,:|k}(\Phi_{:|k})^{-1}\Omega_{:|k} \\ &+ \Phi_{:|k}B_{\Theta,:|k}u_{k+j|k} + \Phi_{:|k}\ r_{\Theta,:|k} + \Omega_{:|k}. \qquad (7.67b)\end{aligned}$$

A compact notation of the difference equation (7.67b), describing the motion of the TCP in the Cartesian space is now provided by:

$$\Psi_{k+j+1|k} = A_{\Psi,:|k}\Psi_{k+j|k} + B_{\Psi,:|k}u_{k+j|k} + r_{\Psi,:|k}, \qquad (7.68)$$

with the similarity matrices:

$$\begin{aligned}A_{\Psi,:|k} &= \Phi_{:|k}A_{\Theta,:|k}(\Phi_{:|k})^{-1}, &(7.69a)\\ B_{\Psi,:|k} &= \Phi_{:|k}B_{\Theta,:|k}, &(7.69b)\\ r_{\Psi,:|k} &= \Phi_{:|k}\ r_{\Theta,:|k} + \Omega_{:|k} - \Phi_{:|k}A_{\Theta,:|k}(\Phi_{:|k})^{-1}\Omega_{:|k}. &(7.69c)\end{aligned}$$

This model is used for the collision avoidance procedure of the `OnHCA`. Due to the receding horizon scheme, the model is updated at every time k based on the robots current configurations and inputs. The updated model is then used for prediction.

7.4.2. Simulation Results

The link lengths, the masses, and the friction coefficients in (7.55) are chosen to:

$$\begin{aligned}l_1 &= l_2 = 1 &(7.70)\\ m_1 &= m_2 = 7 &(7.71)\\ c_1 &= c_2 = 1. &(7.72)\end{aligned}$$

The ZOH discretization of the linearized robot dynamic is performed with a discretization time of $\Delta t = 50$ms. The task is to bring the TCP from an initial Cartesian state $\Psi_{0|0} = (1.4,\ 0,\ 0,\ 0)^T$ to a time-varying final state, which changes from $\Psi^0_{f|0} = (1.3,\ 1.4,\ 0,\ 0)^T$ at $k = 0$ to $\Psi^0_{f|20} = (1.3,\ 0.85,\ 0,\ 0)^T$ at $k = 20$, see Fig. 7.7. The challenge is to avoid any collision with a moving obstacle $\mathcal{P}_{x,k+j|k}$, while reaching the time-varying final state. The obstacle is considered to constantly move from its initial position towards the left side, hence into the direction of the robot. The movement takes place from $k = 0$ until $k = 20$, whereupon the obstacle remains in this position. Additionally, limitations in the input torques are considered for all times of the prediction horizon: $\underline{u} \leq u_{k+j|k} \leq \bar{u}$ for all $j \in \mathcal{J}_{\mathcal{H}}$. The circumventing trajectory is determined by `OnHCA` (see Alg. 7.3). Without going into detail, parameters like d_{fac}, which describes how far the target states of $\mathcal{G}_{f|k}$ are

spread apart, as well as the weighting matrices Q_B, R_B and Q_{Bend} for the determination of the *base trajectories*, see Sec. 7.2, are chosen such that a sufficiently large space for circumventing the obstacle is spanned.

The simulation results in Fig. 7.7 show the circumvention obtained from `OnHCA` with a chosen prediction horizon of $H = 20$ time steps. In this example, there are four *base trajectories* $\{\hat{\Psi}^1_{:|k}, \hat{\Psi}^2_{:|k}, \hat{\Psi}^3_{:|k}, \hat{\Psi}^4_{:|k}\}$ (dotted in black). Two of them reach the target states $\mathcal{X}^{var}_{f|k}$, which adapt at each time step k and the other two reach their target state $\mathcal{X}^{stat}_{f|k}$. For the case of neglecting the obstacle, the trajectory dotted in red represents the optimal trajectory $\hat{\Psi}^0_{:|k}$ towards the time-varying goal state $\Psi^0_{f|k}$. The solution of `OnHCA` is shown by the trajectory colored in magenta. This trajectory represents the collision-free homotopic target trajectory $\hat{\Psi}(\boldsymbol{\lambda}^*_{:|k})$. The green box shows the obstacle $\mathcal{P}_{x,k|k}$ at time k.

As it can be seen in Fig. 7.7a, the robot starts from its initial position by determining a collision-free homotopic target trajectory (magenta), while taking into account the moving obstacle. The magenta colored trajectory of Fig. 7.7a passes the obstacle a little further on the left side, due to the fact that the predicted obstacle positions are also taken into account. The homotopic target trajectory is followed for one time step and the procedure repeats. Successively, an avoidance maneuver takes place. For $k = 8$, Fig. 7.7c shows that `OnHCA` determines a homotopic target trajectory, which equals the optimal trajectory $\hat{\Psi}(\boldsymbol{\lambda}^*_{:|8}) = \hat{\Psi}^0_{:|8}$, as can be seen by the equality of both trajectories (magenta and red trajectories are equal). This means that `OnHCA` is able to select the cost minimal trajectory to the goal state and must not evade to another homotopic trajectory. In time $k = 20$ (see Fig. 7.7e), the system adapts its motion to the new goal state $\Psi^0_{f|20}$. This causes an online adaption of the set $\mathcal{X}^{stat}_{f|k}$ with the corresponding *base trajectories* and finally leads to a new movement of the robot manipulator, shown in Fig. 7.7f. Hence, `OnHCA` reacts to this change in the goal state and the TCP moves down from the upper reached position.

In Fig. 7.6, the computation times and the computational load of `OnHCA` is shown for each step time k. The yellow bars show the portions of time spent for computing the *base trajectories*, while the blue bars are the corresponding times for the tree search in `OnHCA`. The computation times show that `OnHCA` can solve the obstacle avoidance problem for the linearized system in real-time, since the maximum computation times over all k is less then 30 ms. Especially in the first time steps from $k = 0$ to $k = 8$, a higher computational effort is observed. The reason is that `OnHCA` detects the need for circumvention already at the beginning due to the large prediction horizon of $H = 20$. As the obstacle is passed, `OnHCA` can directly select the optimal trajectory and the computation times decrease to small values.

For comparison, the same scenario is shown in Fig. 7.8 with a short prediction horizon of $H = 2$. As a result of this, the TCP moves straight up until the obstacle is nearly reached, see Fig. 7.8b. Only then, the obstacle is recognized due to the short prediction horizon and `OnHCA` starts its circumvention routine. The algorithm detects that the two *base trajectories*, left-sided from the obstacle, are free of collision

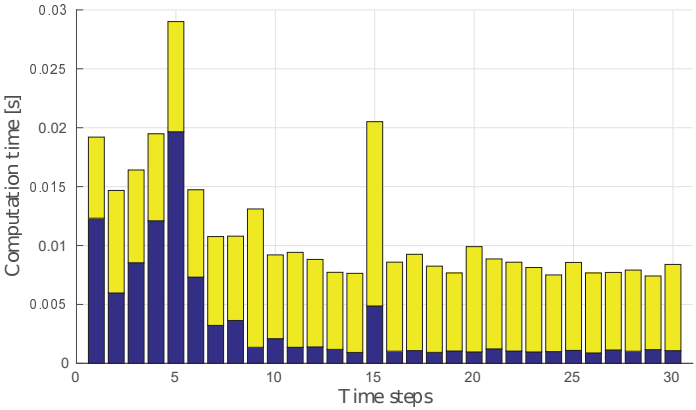
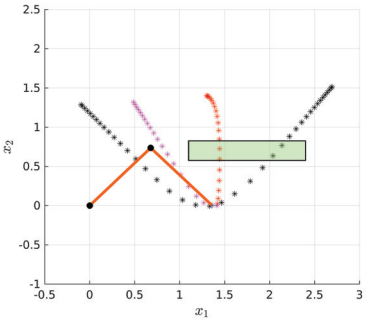


Figure 7.6.: Computation times of `OnHCA` for the circumvention example with prediction horizon $H = 20$, yellow bars: spent time for determination of the *base trajectories*. blue bars: time for the other parts of `OnHCA`.

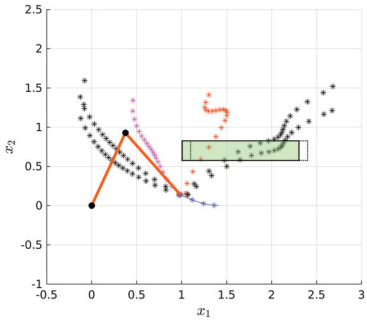
(Fig. 7.8c) and pushes the TCP into this direction by determining an appropriate homotopic target trajectory. This procedure repeats (see Fig 7.8d) until the obstacle is passed. For $k = 20$, again the goal state changes and jumps directly above the obstacle such that the algorithm controls to the new goal state, see Fig. 7.8e and Fig. 7.8f. Comparing e.g. time step $k = 20$ of both scenarios, one can see that the TCP with $H = 20$ could come much closer to the first goal state (before it changed), than in the example with $H = 2$. This is because the circumventing trajectory could be calculated much more efficiently with a larger horizon. One interesting aspect in Fig. 7.8c is, that the developed method is able to pass the obstacle, while a standard MPC implementation would stop at this position in a local minimum. In addition, Fig. 7.8c clearly shows that even though no collision between the TCP and the obstacle exists, one between the link of the robot and the obstacle occurs. The consideration of such collisions is addressed in the next chapter.

The computation time of `OnHCA` with the prediction horizon of $H = 2$ for the determination of an optimal homotopic target trajectory (blue bars) are smaller compared to the case with $H = 20$ (Fig. 7.9). The computation times rise from $k = 5$, when the obstacle is first detected with the small prediction horizon, and remain about the same value until $k = 19$. In this period of time, the tree search of `OnHCA` is active to obtain a circumventing homotopic target trajectory. After $k = 19$, the obstacle is passed, and `OnHCA` is able to directly select the optimal trajectory, meaning that the tree search procedure does not need to be activated. Thus, the computation times fall back to small values for the remaining time steps until $k = 30$. In contrast to the case with $H = 20$, it can be seen that the tree

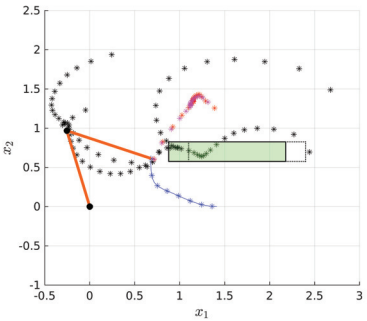
search is active for more time steps as the robot moves for a relatively long time along the obstacle, see Fig. 7.8e.



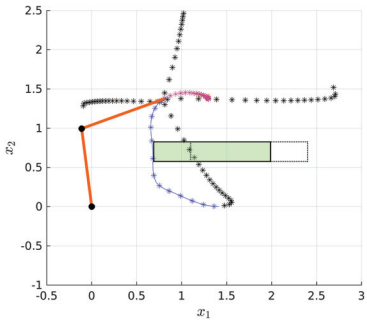
(a) Circumvention at $k = 1$.



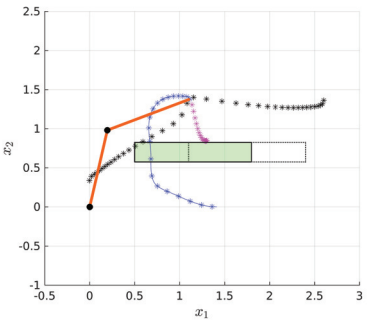
(b) Circumvention at $k = 4$.



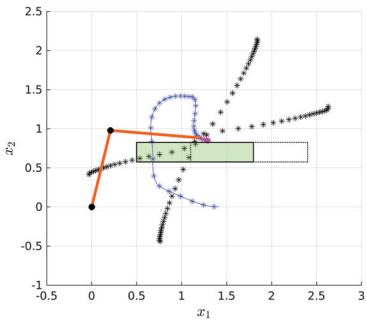
(c) Circumvention at $k = 8$.



(d) Circumvention at $k = 14$.

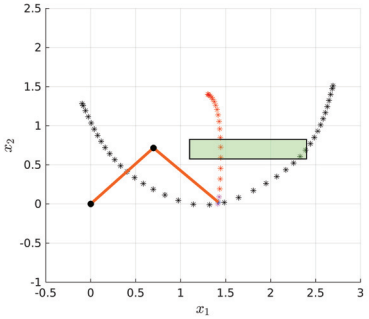


(e) Circumvention at $k = 20$.

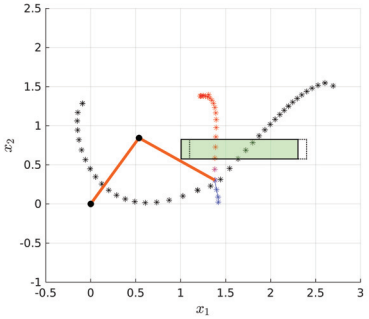


(f) Circumvention at $k = 30$.

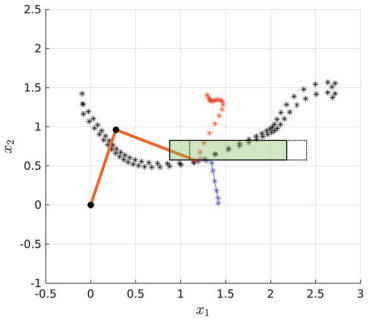
Figure 7.7.: Collision avoidance of a robotic manipulator with a time-varying obstacle (green box), and prediction horizon $H = 20$: *Base trajectories* $\hat{\Psi}^i_{:|k}$, $i = \{1, ..., 4\}$ (black), optimal trajectory without considering the obstacle (red), homotopic target trajectory (magenta), and realized motion of the TCP (blue).



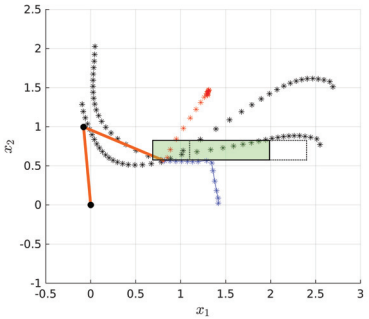
(a) Circumvention at $k = 1$.



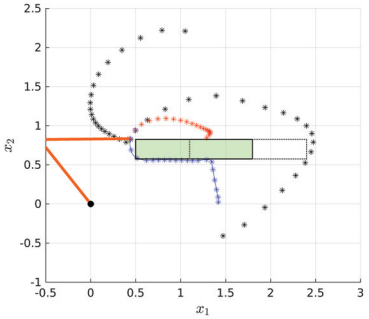
(b) Circumvention at $k = 4$.



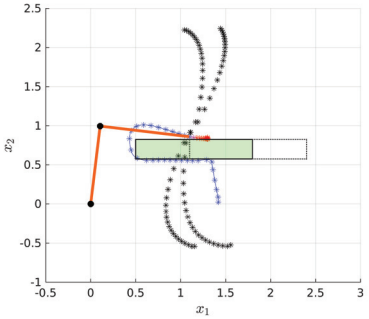
(c) Circumvention at $k = 8$.



(d) Circumvention at $k = 14$.



(e) Circumvention at $k = 20$.



(f) Circumvention at $k = 30$.

Figure 7.8.: Collision avoidance by the `OnHCA`, with prediction horizon $H = 2$.

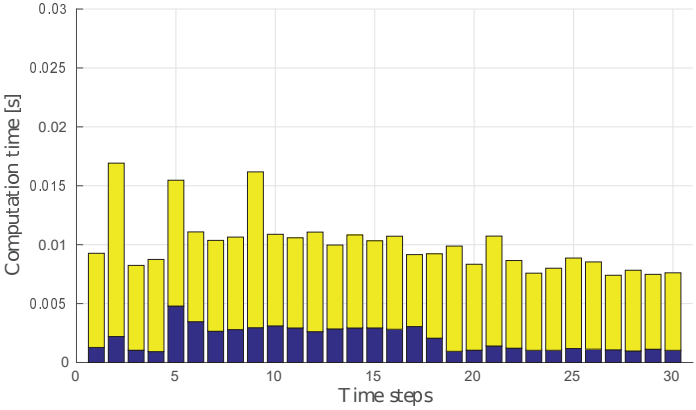


Figure 7.9.: Computation times of the `OnHCA` for the circumvention example with prediction horizon $H = 2$. Yellow bars: time spent for determination of the *base trajectories*. Blue bars: remaining time of the `OnHCA`

7.5. Discussion

The developed homotopic control method HCA of the previous Chapter 6 provides an online capable method for linear systems with fixed initial and goal states. This chapter adapts the HCA by introducing an online approach for the determination of *base trajectories*. The necessity for online determining *base trajectories* follows from the existence of nonlinear dynamics and the consideration of time-varying goal states. The procedure linearizes the dynamics, generates *base trajectories* and finally uses the same procedure as described in the HCA for collision avoidance. Summarizing these steps yields OnHCA.

The linearization is performed in every time step, so that the computations of the OnHCA are always based on the actually best known linear approximation of the system. Of course, the linearization leads to a deviation with increasing time horizon between the linearized model and the nonlinear dynamics. But the effect of this disadvantage is reduced by the receding horizon principle of the OnHCA. Nevertheless, because the planning is based on a linearized model and the receding horizon scheme, collision avoidance can not be guaranteed completely. Therefore, one would combine this real-time planning procedure with mechanisms like a distance based velocity reduction until stopping to guarantee safety.

The procedure for determining online *base trajectories*, with generated target states, shaped as an n-dimensional simplex around the goal state of the system was used here as a simple and efficient method. Certainly, there are many alternatives to simplices for arranging the target states of the *base trajectories* around the goal state. Nevertheless, the advantage of simplices is that it generates a minimum number of *base trajectories* spanning the space for circumvention. Additionally, the strategy of generating a set of static and variable target states for the base trajectories allows the system to pass an obstacle even if the obstacle is almost in front of the current state. A standard MPC implementation of such a problem can easily get stuck, i.e. a circumvention of the obstacle would not be found.

As already described in the discussion of Sec. 6.5, OnHCA has the same advantages as the low dimensional optimization problems solved in each node of the tree search. The identification of collision times in the homotopy space is the main reason of making this procedure so fast. The simulation example of Sec. 7.4.2, clearly shows the strength of this procedure. The highly nonlinear robot manipulator dynamics, which is additionally coupled with the highly nonlinear transformation from the configuration to the state space is solved for a moving obstacle and a time-varying goal state with low computation times. The example shows, that even with four *base trajectories*, a sufficient variety for circumvention is given such that OnHCA easily solves this problem.

However, when taking a closer look at Fig. 7.8c and Fig. 7.8d, one can see, that although the TCP is not colliding with the obstacle at any time, an intersection between one of the robot links and the obstacle occurs. The reason for this that OnHCA only considers the TCP in the planning process, but not the robot links. The

consideration of the robot geometry in collision avoidance and the extension to the cooperation of several robots are addressed in the following chapter.

8. Homotopic Control for Systems with Polytopic Space Occupancy

The previous chapter has motivated the online control of nonlinear systems, using homotopies. For real world problems like a human-robot interaction in an industrial assembling line, the `OnHCA` was developed. It uses homotopy properties for online circumvention of moving obstacles (as the motion of a human can be approximated by one). It was shown that the `OnHCA` controlled the TCP from an initial to a goal state while avoiding collisions. Nevertheless, a collision free trajectory of the TCP does not imply the whole body of the robot to be free of collision with the obstacle. This chapter addresses the named problem and extends the `OnHCA` by a particle approach. Representing the systems geometry with an appropriate number of particles, a collision-free motion of the system is targeted, such that for robotic manipulators a collision between one of the links and the obstacle can thus be avoided.

The chapter begins with a problem definition in Sec. 8.1, followed by an introduction on how particles are used to describe the systems polytope Sec. 8.2. The part in which the developed `OnHCA` from the previous chapter is extended by a particle approach, Sec. 8.3 and finally a simulation example for the robotic manipulator in Sec. 8.4.

8.1. Problem Definition

According to Def. 7.1, a nonlinear system is described by its linearization at the current point of time k around its actual state and the predicted input from the last time step. The task is to bring the system (7.1) from any initial state x_s to a time-varying goal state $x^0_{f|k}$, while avoiding collisions with any state space obstacle, respectively polytope of another agent $\mathcal{P}_{x,k+j|k}$, known within a prediction horizon $j \in \mathcal{J}_{\mathcal{H}} := \{1, ..., H\}$. The subscript $\nu \in \overset{[\nu]}{\Sigma} := \{1, ..., |\Sigma|\}$ denotes the ν-th obstacle/agent of $|\Sigma|$ many. The control problem is solved based on `OnHCA` from Chapter. 7, which quickly determines a homotopic target trajectory $\boldsymbol{\lambda}^*_{\cdot|k}$ for collision avoidance. This homotopic trajectory finally minimizes the given cost

function $J(\boldsymbol{\lambda}_{:|k})$ with weighting matrices $Q \in \mathbb{S}^{n_x}_{>0}$, $Q_{end} \in \mathbb{S}^{n_x}_{>0}$ and $R \in \mathbb{S}^{n_u}_{>0}$:

$$\begin{aligned} J(\boldsymbol{\lambda}_{:|k}) =& \|x_{k+H|k}(\boldsymbol{\lambda}_{:|k}) - x^0_{f|k}\|^2_{Q_{end}} \\ &+ \sum_{j=0}^{H-1} \|x_{k+j|k}(\boldsymbol{\lambda}_{:|k}) - x^0_{f|k}\|^2_Q + \|u_{k+j|k}(\boldsymbol{\lambda}_{:|k}) - u^0_{f|k}\|^2_R \end{aligned} \tag{8.1a}$$

and the well known homotopic states and inputs:

$$x_{k+j|k}(\boldsymbol{\lambda}_{:|k}) = x^0_{k+j|k} + D_{x,k+j|k}\boldsymbol{\lambda}_{:|k},\ \forall j \in \mathcal{J}_{\mathcal{H}} \tag{8.2a}$$

$$u_{k+j|k}(\boldsymbol{\lambda}_{:|k}) = u^0_{k+j|k} + D_{u,k+j|k}\boldsymbol{\lambda}_{:|k},\ \forall j \in \mathcal{J}_{\mathcal{H}}. \tag{8.2b}$$

This basic problem is now extended to the problem of collision avoidance with consideration of polytopes for both, the controlled system and the obstacles/agents.

Collision Avoidance of Polytopes

Consider the case, where the system (7.1) itself occupies a region of the state space $\mathcal{P}_{x,k+j|k}$ at each point of time. For a particular point κ in this polytope, e.g. the polytopes center or the TCP of a robotic manipulator, the state evolution $x_{\substack{k+j|k \\ \{\kappa\}}}$ satisfies:

$$x_{\substack{k+j|k \\ \{\kappa\}}} \in \mathcal{P}_{x,k+j|k}. \tag{8.3}$$

The task of collision avoidance between the state $x_{\substack{k+j|k \\ \{\kappa\}}}$ and any ν-th polytopic obstacle/agent $\mathcal{P}_{\substack{x,k+j|k \\ [\nu]}}$ is thus extended to the task of collision avoidance of polytopes:

$$\bigwedge_{j \in \mathcal{J}_{\mathcal{H}}, \nu \in \Sigma} \mathcal{P}_{x,k+j|k} \cap \mathcal{P}_{\substack{x,k+j|k \\ [\nu]}} = \emptyset. \tag{8.4}$$

The problem which arises is the question how (8.4) can be efficiently included in the planning process to control the dynamic system with its geometric shape $\mathcal{P}_{x,k+j|k}$ without colliding with any other $\mathcal{P}_{\substack{x,k+j|k \\ [\nu]}}$. To satisfy this, an algorithmic extension of `OnHCA` is shown in this chapter.

8.2. Consideration of Polytopes for Collision Avoidance

In general, it is easier to approximate a body by one single point and describe its motion by one state vector, as to consider also its region that the system occupies. Nevertheless, to prevent collision the polytope of the system and the obstacles/agents $\mathcal{P}_{\substack{x,k+j|k \\ [\nu]}}$, $\nu \in \Sigma$ have to be considered either by constraints describing

with binary variables or by other approximations. While the first case again leads to solving a time-consuming mixed integer problem, an approximation by using particles describing the systems polytope or an enlargement of the obstacle by consideration of the system polytope is useful. Enlarging the obstacle at least by the systems polytope ensures that the system can be again approximated by one single state such that when the system circumvents the obstacle, the distance between the system polytope and the obstacle is sufficiently large. When using particles κ, the systems polytope is described by a finite number of points. With respect to the robot example, these points are distributed along the robots links. Each of them has its own linearized model because of their different locations, but the same input would be applied to all. A mixed approach on the other side uses particles approximating the polytope, but also enlarges the obstacle. The obstacle is enlarged at least by the greatest distance that any particle has to a any point of the polytope without being describable by another particle with a smaller distance. This prevents the trajectory of e.g. two neighboring particles from passing the obstacle, but the body between them to intersect with the obstacle.

Obstacle Enlargement Approach

Enlarging each obstacle $\underset{[\nu]}{\mathcal{P}_{x,k+j|k}}$ by the system polytope $\mathcal{P}_{x,k+j|k}$ leads to obstacles $\underset{[\nu]}{\hat{\mathcal{P}}_{x,k+j|k}} \supset \underset{[\nu]}{\mathcal{P}_{x,k+j|k}}$:

$$\underset{[\nu]}{\hat{\mathcal{P}}_{x,k+j|k}} := \{x | x \in \mathcal{P}_{x,k+j|k} \cup \underset{[\nu]}{\mathcal{P}_{x,k+j|k}} \;\; \exists l \in \partial\mathcal{P}_{x,k+j|k} \wedge l \in \underset{[\nu]}{\mathcal{P}_{x,k+j|k}}\} \tag{8.5}$$

As described before, by enlarging the obstacles by the system polytope, the control task can be again projected back to a formulation, in which collision avoidance is in general satisfied by avoiding collisions between the state $\underset{\{\kappa\}}{x_{k+j|k}}$ and the now enlarged obstacle $\underset{[\nu]}{\hat{\mathcal{P}}_{x,k+j|k}}$:

$$\underset{[\kappa]}{x_{k+j|k}} \notin \underset{[\nu]}{\hat{\mathcal{P}}_{x,k+j|k}}, \; \forall \nu \in \Sigma, \; j \in \mathcal{J}_{\mathcal{H}}. \tag{8.6}$$

The advantage of this approach is that the number of constraints and optimization variables remains the same when using (8.6) in an optimization problem. However, the main disadvantage is that this approach is conservative and a solution for the collision avoidance of (8.6) often cannot be found. The reason for this is that the obstacles $\underset{[\nu]}{\hat{\mathcal{P}}_{x,k+j|k}}$ become enlarged by the maximum expansion that the obstacle and the system polytope can reach when they are in contact. Thus solutions are excluded in which e.g. the system still could pass the obstacle by a change of the polytopes orientation. This leads to either non feasible trajectories if the circumventing trajectory is not inside the permissible state space or in the feasible case to solution trajectories with higher costs.

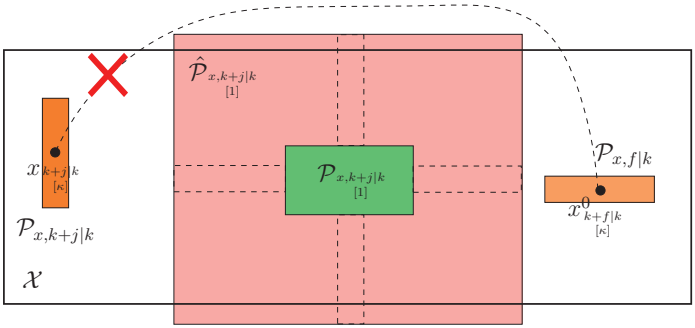


Figure 8.1.: Illustration of the enlarged obstacle $\hat{\mathcal{P}}_{\substack{x,k+j|k\\[1]}}$ of $\mathcal{P}_{\substack{x,k+j|k\\[1]}}$, and the problem of circumvention.

In Fig. 8.1, an obstacle $\mathcal{P}_{\substack{x,k+j|k\\[1]}}$, the occupied space of the system $\mathcal{P}_{x,k+j|k}$, and the resulting enlarged obstacle $\hat{\mathcal{P}}_{\substack{x,k+j|k\\[1]}}$ is shown. The figure shows the situation where no trajectory from the current state $x_{\substack{k+j|k\\[\kappa]}}$ to the goal state $x^0_{\substack{k+f|k\\[\kappa]}}$ can be found which is inside the feasible space $\mathcal{X}$ when the obstacle $\mathcal{P}_{\substack{x,k+j|k\\[1]}}$ is enlarged by the maximal possible expansion with the system polytope $\mathcal{P}_{x,k+j|k}$.

Particle Approach

In contrast to enlarging the obstacle by the system polytope such that the system is describable by one particle, here the system polytope $\mathcal{P}_{x,k+j|k}$ is approximated by a set of particles $\kappa \in \mathcal{K} := \{1, ..., |\mathcal{K}|\}$. The particles are chosen to be uniformly distributed along the system polytope.

Since the motion of the polytope is now described by several particles, a linear model is derived for each particle based on its specific position in the state space:

Definition 8.1. *Given the nonlinear, continuous time system:*

$$\dot{x}_\kappa(t) = f(x_\kappa(t), u(t)), \tag{8.7}$$

as e.g. given by the dynamics of the TCP or any other particle on a robot manipulator (see (7.68)*). The dynamics of a particle* $x_{\substack{k+j|k\\\{\kappa\}}} \in \mathcal{P}_{x,k+j|k}$ *is then given by the linearization and time-discretization of the nonlinear system* (8.7) *at particle* $x_{\substack{k+j|k\\\{\kappa\}}}$*, with* $\kappa \in \mathcal{K}$ *and input value* $u_{k|k-1}$*:*

$$x_{\substack{k+j+1|k\\\{\kappa\}}} = A_{\substack{:|k\\\{\kappa\}}}\ x_{\substack{k+j|k\\\{\kappa\}}} + B_{\substack{:|k\\\{\kappa\}}}\ u_{k+j|k} + r_{\substack{:|k\\\{\kappa\}}}\,. \tag{8.8}$$

To each particle $x_{\substack{k+j|k\\\{\kappa\}}}$*, the same input* $u_{k+j|k}$ *is applied.*

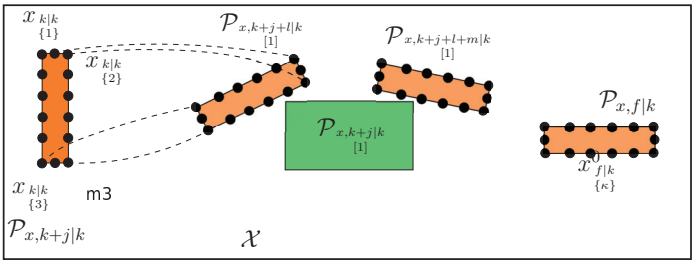


Figure 8.2.: Particle approach for circumventing $\mathcal{P}_{\substack{x,k+j|k\\[1]}}$ with the system polytope $\mathcal{P}_{x,k+j|k}$. The trajectories of all in total 13 particles $x_{\substack{k|k\\\{\kappa\}}}$ have to be free of collisions.

The goal is then to find the control inputs $u_{k+j|k}$ such that the resulting trajectory of each particle does not collide with the obstacles $\mathcal{P}_{\substack{x,k+j|k\\[\nu]}}$ over the prediction horizon. It holds that $\forall \kappa \in \mathcal{K}$ and $\forall j \in \mathcal{J}_{\mathcal{H}}$:

$$x_{\substack{k+j|k\\\{\kappa\}}} \notin \mathcal{P}_{\substack{x,k+j|k\\[\nu]}}, \quad \forall \nu \in \Sigma. \tag{8.9}$$

Fig.8.2 illustrates the particle approach again with the space occupied by the system polytope $\mathcal{P}_{x,k+j|k}$, its particles $x_{\substack{k+j|k\\\{\kappa\}}}$ and an obstacle $\mathcal{P}_{\substack{x,k+j|k\\[1]}}$. The trajectory of each particle is now determined such that no collision for each trajectory exists. This forces the system polytope to pass the obstacle. It can be seen in the example of Fig.8.2 that a solution for circumventing the obstacle and reaching the final position can be found by rotating the system polytope. The disadvantage here is, however, that a large number of particles has to be considered during the optimization to prevent that intermediate parts between the particles collide with the obstacle. This quickly makes the particle approach not applicable in real-time when combining with `OnHCA`, especially when considering higher dimensions.

Mixed Approach

The approach proposed in this work uses a combination of the obstacle enlargement and the particle approach to be integrated in `OnHCA`. Thus, the advantages of both approaches, i.e. low complexity and small conservatism in safe distance between obstacle an system are unified. The obstacles $\bar{\mathcal{P}}_{\substack{x,k+j|k\\[\nu]}}$ are sufficient enlargements of $\mathcal{P}_{\substack{x,k+j|k\\[\nu]}}$:

$$\mathcal{P}_{\substack{x,k+j|k\\[\nu]}} \subset \bar{\mathcal{P}}_{\substack{x,k+j|k\\[\nu]}} \subset \hat{\mathcal{P}}_{\substack{x,k+j|k\\[\nu]}}, \tag{8.10}$$

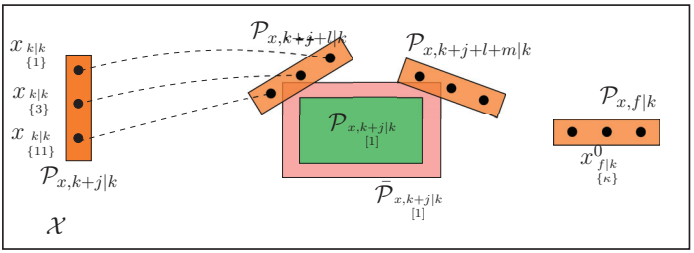


Figure 8.3.: Unification of particle approach and obstacle enlargement. The trajectories of $x_{\substack{k|k\\{\kappa}}}$ are forced to circumvent the enlarged obstacle $\bar{\mathcal{P}}_{\substack{x,k+j|k\\[1]}}$.

such that besides the collision avoidance for any particle κ, the set $\mathcal{P}_{x,k+j|k}$ also becomes free of collisions. The value for enlargement is chosen at least by the distance of neighboring particles. The collision avoidance condition (8.4) is then satisfied if $\forall \kappa \in \mathcal{K}$ and $\forall j \in \mathcal{J}_{\mathcal{H}}$ the following condition holds:

$$x_{\substack{k+j|k\\{\kappa}}} \notin \bar{\mathcal{P}}_{\substack{x,k+j|k\\[\nu]}}, \quad \forall \nu \in \Sigma. \tag{8.11}$$

This implies that the original collision avoidance condition holds:

$$\mathcal{P}_{x,k+j|k} \cap \mathcal{P}_{\substack{x,k+j|k\\[\nu]}} = \emptyset. \tag{8.12}$$

An illustration of this principle is shown in Fig. 8.3.

8.3. Homotopic Control Algorithm with Particles

The mixed approach with the obstacle enlargement and the use of particles κ according to Def. 8.1 is embedded into `OnHCA` of Sec. 7.3.3. The result is the online particle extended homotopic control algorithm `OnPeHCA`. This algorithm determines an optimal homotopy parameter $\boldsymbol{\lambda}^*_{:|k}$ such that each homotopic target trajectory is given by the sequence of states:

$$x_{\substack{k+j|k\\{\kappa}}}(\boldsymbol{\lambda}^*_{:|k}) = x^0_{\substack{k+j|k\\{\kappa}}} + D_{\substack{x,k+j|k\\{\kappa}}} \boldsymbol{\lambda}^*_{:|k}, \ \forall j \in \mathcal{J}_{\mathcal{H}}, \tag{8.13}$$

satisfies the collision avoidance condition (8.11). Before starting `OnPeHCA`, a small set of particles is defined that approximates the system polytope. Obstacles and their enlargements $\bar{\mathcal{P}}_{\substack{x,k+j\\[\nu]}}$ are given over the prediction horizon. The algorithm starts by selecting first an enlarged obstacle $\nu \in \Sigma$ and a particle $\kappa \in \mathcal{I} := \mathcal{K}$ from the set of distributed particles in the system polytope (*line 3-6*). Then the optimal homotopic target trajectory $\hat{x}_{\substack{:|k\\{\kappa}}}(\boldsymbol{\lambda}^*_{:|k})$ is determined by the `OnHCA` (Alg. 7.3) (*line*

7-8). The result of `OnHCA` algorithm is a collision-free trajectory of the selected particle. Nevertheless, the determined homotopy parameter $\boldsymbol{\lambda}^*_{:|k}$ can lead to collisions between any other particle and the selected obstacle, because until now, the input signal was determined based only on the one considered particle. This does not guarantees that the same input leads to collision-free trajectories for the other particles. Thus, the algorithm first checks for such collisions by simply testing if a particle is inside the polytope at any time $j \in \mathcal{J}_{\mathcal{H}}$ and stores all particles that are still in collision with the obstacle in the set $\mathcal{I}$ (*line 10-11*):

$$\mathcal{I} := \{\kappa |\ \exists j \in \mathcal{J}_{\mathcal{H}} : x_{\underset{\{\kappa\}}{k+j|k}}(\boldsymbol{\lambda}^*_{:|k}) \in \bar{\mathcal{P}}_{\underset{[\nu]}{x,k+j}}\}. \tag{8.14}$$

During the execution of `OnHCA` in the `OnPeHCA` algorithm, the set of constraints $\mathcal{O}$ from the tree search (see (7.38) of Sec. 7.3.3) is identified. This set provides constraints on the homotopy parameter such that the trajectory of the considered particle is not colliding with the obstacle. The set $\mathcal{O}$ is recorded (*line 9*) and guarantees a collision free trajectory for the considered particle. With the obtained input, the trajectories for the remaining particles are determined and each of the remaining trajectories is checked against collisions (*line 11-12*). If the set $\mathcal{I}$ is identified to be empty, a circumventing trajectory for all trajectories is found and the system polytope passes the obstacle. The algorithm then proceeds by taking the next obstacle $\nu \in \Sigma$ into account (see for-loop *line 3*). If however $\mathcal{I}$ is not empty, successively the next particle $\kappa \in \mathcal{I}$ is selected and `OnHCA` is again executed. The set $\mathcal{O}$ of homotopy constraints now also contains the constraints from the previous particle such that also the previous particle maintains collision-free (*line 5-13*). The successive consideration of individual particles is carried out until a solution is found in which all trajectories are collision-free. Once all obstacles have been processed, the algorithm yields circumventing homotopic target trajectories to all obstacles. Then the algorithm executes one time step and repeats its computation from the beginning (*line 14-16*).

The execution of one time step k, in Alg. 8.1 is schematically shown in Fig. 8.4. The figures show a stationary obstacle $\mathcal{P}_{\underset{[1]}{x,k+j|k}}$, its enlargement $\bar{\mathcal{P}}_{\underset{[1]}{x,k+j|k}}$ and three particles $x_{\underset{\{1\}}{k|k}}$, $x_{\underset{\{2\}}{k|k}}$ and $x_{\underset{\{3\}}{k|k}}$. In Fig. 8.4(a), `OnPeHCA` first determines a collision-free trajectory for particle $x_{\underset{\{1\}}{k|k}}$. Since collisions between the two particles $x_{\underset{\{2\}}{k|k}}$, $x_{\underset{\{3\}}{k|k}}$ and the obstacle $\bar{\mathcal{P}}_{\underset{[1]}{x,k+j|k}}$ are still present, the while-loop of the algorithm is repeated by considering the colliding particle $x_{\underset{\{2\}}{k|k}}$ and the constraints $\mathcal{O}$ obtained from the previous loops. The result is a collision-free trajectory for the particles $x_{\underset{\{1\}}{k|k}}$ and $x_{\underset{\{2\}}{k|k}}$, see Fig. 8.4(b). Since the trajectory of $x_{\underset{\{3\}}{k|k}}$ is identified to be still in collision, the iteration is repeated such that finally all particles become free of collisions, Fig. 8.4(c).

Algorithm 8.1.: Online Particle Extended Homotopic Control Algorithm (`OnPeHCA`)

Given: $x^0_{f|k}$, $\mathcal{J}_\mathcal{H}$, $\underset{[\nu]}{\mathcal{P}_{x,k+j}}$, $\underset{[\nu]}{\bar{\mathcal{P}}_{x,k+j}}$, $\nu \in \Sigma$, particles $\underset{\{\kappa\}}{x_{k|k}}$, $\kappa \in \mathcal{K}$
set $\mathcal{O} := \emptyset$
for $\nu = 1 : |\Sigma|$ **do**
  set $\mathcal{I} := \mathcal{K}$
  while $\mathcal{I} \neq \emptyset$ **do**
    select a particle $\kappa \in \mathcal{I}$
    determine $\boldsymbol{\lambda}^*_{:|k}$ with the `OnHCA`, see Alg. 7.3, for particle $\underset{\{k\}}{x_{k+j|k}}$,
    and obstacle $\underset{[\nu]}{\bar{\mathcal{P}}_{x,k+j}}$
    store the identified constraints for $\boldsymbol{\lambda}^*_{:|k}$ in $\mathcal{O}$
    determine the trajectories of all particles $\underset{\{\kappa\}}{\hat{x}_{:|k}}(\boldsymbol{\lambda}^*_{:|k})$ according to (8.13)
    identify particles in collision $\mathcal{I} := \{\kappa |\ \exists j \in \mathcal{J}_\mathcal{H} : \underset{\{\kappa\}}{x_{k+j|k}}(\boldsymbol{\lambda}^*_{:|k}) \in \underset{[\nu]}{\bar{\mathcal{P}}_{x,k+j}}\}$
  end while
end for
execute determined $u_{k|k}(\boldsymbol{\lambda}^*_{:|k})$ (8.2b) by the n.l. system (8.7)
set $k := k + 1$
goto line 1.

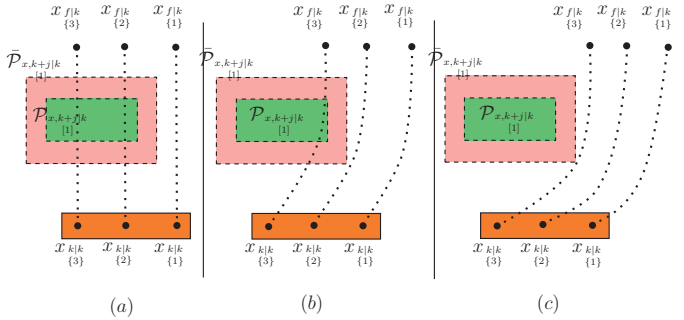
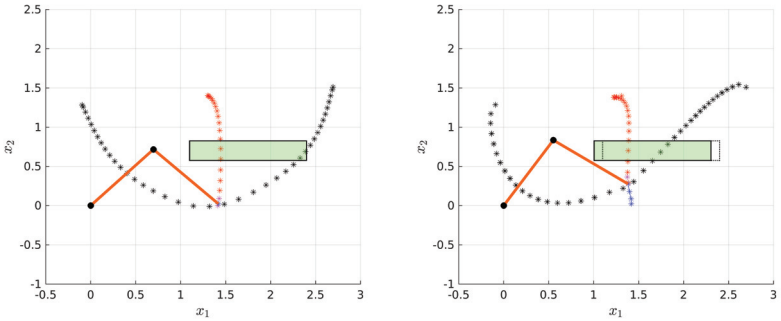


Figure 8.4.: Shows the trajectories of the `OnPeHCA`, for each iteration of the while-loop. In (a), only particle $\underset{\{1\}}{x_{k|k}}$ is considered, in (b) additionally particle $\underset{\{2\}}{x_{k|k}}$, and finally all particles in (c). The green box is the obstacle $\underset{[1]}{\mathcal{P}_{x,k+j|k}}$, and the red its enlargement $\underset{[1]}{\bar{\mathcal{P}}_{x,k+j|k}}$.

8.4. Simulation Results for Body Circumvention

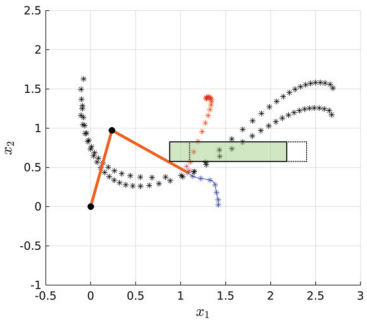
The simulation considered here is based on the manipulator dynamics from Sec. 7.4.1. In order to have a comparison between `OnPeHCA` and `OnHCA`, the scenario equals the one described in Sec. 7.4.2, where the robotic manipulator starts from the initial state $\Psi_{0|0} = (1.4,\ 0,\ 0,\ 0)^T$ on the bottom side of the moving obstacle $\mathcal{P}_{x,k+j|k}$ and moves towards the time-varying final state $\Psi^0_{f|0} = (1.3,\ 1.4,\ 0,\ 0)^T$ for $k = 0$ and eventually up to $\Psi^0_{f|20} = (1.3,\ 0.85,\ 0,\ 0)^T$ for $k = 20$. With a selected prediction horizon of $H = 2$, the simulation results in Sec. 7.4.2 show a collision-free movement of the TCP, see Fig. 7.8. Nevertheless, it can be seen that a collision between one of the robots links and the obstacle occurs, see Fig. 7.8c. The simulation result of this section again shows the circumvention procedure for $H = 2$, but now solved by `OnPeHCA`. It considers a set of $\mathcal{K} = \{1, ..., 10\}$ particles equally distributed along the second links of the robot. Despite the small prediction horizon, one can see in Fig. 8.5 that `OnPeHCA` solves the problem such that no collisions between the robotic link and the obstacle occur. Especially when comparing the time step $k = 8$ of Fig. 8.5c with the solution in Fig. 7.8c from Sec. 7.4.2, the robot starts moving earlier to the left side, such that the link stays free of collisions.

Unfortunately, the consideration of particles leads to an increase of the computation times, see Fig. 8.6. It is can be noted that some points in time ($k = \{4, 8, 13, 14\}$) have significantly larger computation times. The reason for this is that `OnPeHCA` has to perform its optimization for several particles being identified in collision with the obstacle. However, the computation times in MATLAB are still acceptable when the robot dynamics is discretized for a time step $\Delta t = 50$ms.

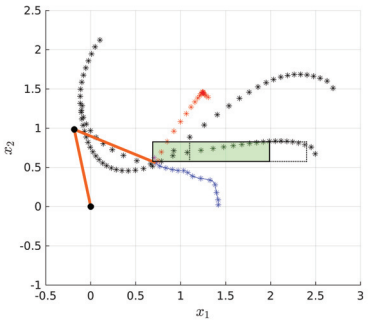


(a) Circumvention at $k = 1$.

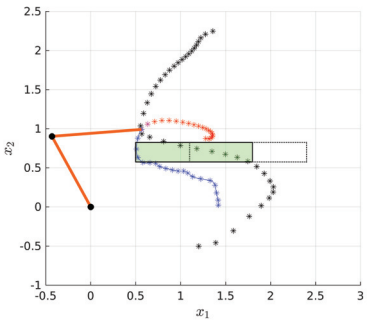
(b) Circumvention at $k = 4$.



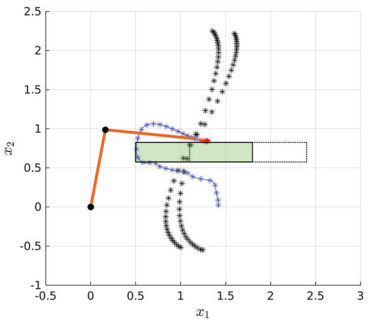
(c) Circumvention at $k = 8$.



(d) Circumvention at $k = 14$.



(e) Circumvention at $k = 20$.



(f) Circumvention at $k = 30$.

Figure 8.5.: Collision avoidance by `OnPeHCA`, with prediction horizon $H = 2$, time-varying obstacle (green box), *base trajectories* $\hat{\Psi}^i_{:|k}$, $i = \{1, ..., 4\}$ (black), optimal trajectory without considering the obstacle (red), homotopic target trajectory (magenta), realized motion of the TCP (blue).

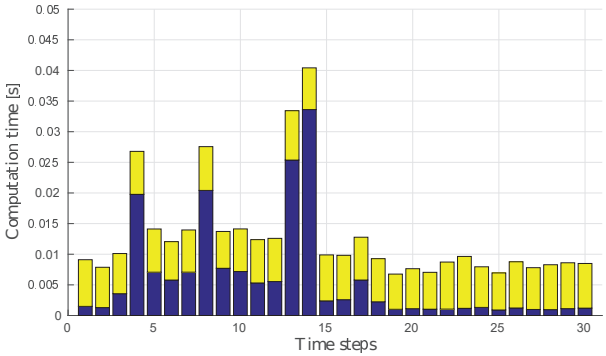


Figure 8.6.: Computation times of `OnPeHCA` with prediction horizon $H = 2$; yellow bars: time spent for determination of the *base trajectories*; blue bars: time for the remaining steps of `OnPeHCA`.

8.5. Discussions of the Particle Approach

Since all obstacle avoidance problems are in a strict sense collision avoidance problems between geometric bodies, the extension of `OnHCA` to `OnPeHCA` provides an important contribution to real world problems. The consideration of particles is easily embedded into the homotopic control approach by successively considering one by one particle, until a collision-free motion of all particles is found. The advantage of this procedure is that termination of `OnPeHCA` can already occur for the first particle, namely if all other particles are also free of collisions for the obtained trajectories. Thus, the algorithm must not process all particles, which in turn improves the speed of the algorithm. Another advantage is the combination between the particle approach and the obstacle enlargement. By unifying these two approaches, the number of particles can be kept small while at the same time, the occupied spaces by the obstacles do not have to be approximated too conservatively to guarantee a save circumvention when approximated by single particle. Especially when thinking of a higher-dimensional circumvention problem, the number of required particles can be kept small.

The investigation of particles in `OnPeHCA` is sequentially triggered each time a collision exists. Instead of executing the algorithm by considering the particles in an ordered sequence, a more comprehensive selection principle could be chosen. If e.g. the algorithm does not receive a collision-free solution until the last particle has been taken into account, it would be more efficient to start `OnPeHCA` with this particle. This would further speed up `OnPeHCA` since it could terminate already with the first particle. One idea could be to evaluate the trajectories of the particles in advance based on how long and how ”deep” they cut the obstacle. With this information, the set of particles could then be sorted and considered in the algorithm. Nevertheless, with respect to the obtained trajectories and the computation times, the developed procedure efficiently solves the collision avoidance problem for polytopes.

9. Cooperative and Distributed Homotopic Control

The contribution of this chapter focuses on the control of multi-agent systems by embedding `OnHCA` and the particle approach into a distributed and cooperative control setting. Especially for manufacturing plants with many robots, an organized and cooperative work of the robots is a challenging question in the field of digitalization and automation. Based on linearizations of nonlinear systems, the introduced method shows how the `OnHCA` can locally operate on each agent and can be coupled in a distributed way to cooperate. The goal is that each agent determines its homotopic target trajectory by a local `OnHCA` while considering a global cost function representing the costs of all agents. An important aspect in the development of the cooperative homotopic control algorithm is the question on how each agent is represented in the global cost function, and how this representation is transferred into the selection of the homotopic target trajectory for each agent. To achieve better closed-loop performance, some level of communication between the agents has to be established. The challenge is to continuously improve the homotopic target trajectories of each agent when communication between them proceeds. The obtained homotopic target trajectories have to be feasible and free of collision for all agents. Since a distributed framework is known to strongly rely on a good and feasible initialization of the trajectory of each agent, a tree search approach is introduced to select appropriate *base trajectories*.

The chapter begins with a literature review on distributed MPC (DMPC) Sec. 9.1. A problem definition is given in Sec. 9.2, followed by introducing `COnPeHCA` for a distributed and cooperative control setting, see Sec. 9.3. A simulation scenario for two cooperating robotic manipulators is given in Sec. 9.4, while the chapter concludes with a discussion in Sec. 9.5.

9.1. Literature Review

Faced with the requirement of safety, communication with the environment, and system performance, model-based control methods like model predictive control (MPC) received significant attention in the last decades. Since MPC is well established with respect to stability and optimality properties, a main advantage of using MPC is the possibility to consider constraints when determining the control actions.

In modern times, assembling in industrial processes represents a highly interlinked process in which many steps are dependent on each other. The control of such coupled subsystems receives more and more attention, since networking and additional sensor devices become cheap and already found their way into industrial machines. The existence of these datasets early motivated not only to control each subsystem separately [5], but also to consider the influence of a subsystem to another subsystems in the control process. To directly account for interactions and constraints, centralized model-based control approaches like MPC can be used. However, centralized approaches are intractable for some applications, since the number of decision variables becomes high for large systems which significantly increases the computation times. Besides that, a centralized approach causes organization and maintenance problems. This means that all data of each subsystem has to be merged in one centralized control unit while at the same time a high availability of the system must be guaranteed.

These considerations motivate the development of distributed control methods, in which each subsystem carries out the calculations on a separate processor. The controller adapts its control strategy by considering interaction among (all) other agents. A communication layer shares the information among the local controllers. Shareable information are e.g. constraints, states, or other objectives. Thus, distributed control methods can be found in fields such as power grids [116], vehicle control [26], or process industry [76]. In the past decades, a significant amount of research on distributed control has been carried out, such that many approaches to distributed control have been developed. Good overviews on distributed control and on the categorization of the various approaches are given in [87], [21] and [16].

One of the main criteria is whether the subsystems calculate their control actions in a cooperative or a non-cooperative way. In a non-cooperative setting, an agent pays attention only to the improvement of its own costs. The optimal solution of one agent can, however, force the remaining agents to worse solutions , see [8]. The results are poor common costs for the sum over all agents.

On the other side, a controller is termed "cooperative", if it minimizes not only its own local costs, but considers system-wide costs. In [109], a cooperative distributed control strategy for linear systems is shown. The algorithm operates iteratively and it was shown that the closed-loop performance converges to the optimal solution of a centralized formulation, a system-wide convex control objective is used. In the work of [85], [84], [56] et.al., a distributed control algorithm for a consensus problem is introduced. The procedure optimizes local cost functions and the communication is limited to neighboring systems. A tracking strategy for changing setpoints is given in [30]. By means of a warm start solution and derived sets of feasible solutions, the cooperative and DMPC method ensures feasibility and a large domain of attraction. In [98], cooperating Uninhabited Areal Vehicles (UAVs) are coupled by constraints. The constraints are given by collision avoidance conditions between the UAVs. A DMPC approach is used which uses mixed integer programming and a global objective to satisfy the constraints and minimize global costs.

DMPC algorithms can be further separated into iterative and non-iterative architectures. In non-iterative architectures, see e.g. [80], a global optimum is not in the focus. The subsystems change their optimized trajectories only by a certain amount compared to the initial candidate input. By using robustness considerations like robust MPC, the trajectory changes are considered in the neighbor predictions as disturbances without iterative communication. In [75], a non-iterative DMPC with Lyapunov stability conditions is proposed. A platooning example for vehicles with non-iterative DMPC is given in [77]. Compared to that, an iterative architecture allows the controllers to communicate their control actions several times during one sampling time. Simultaneously, the controllers can fully communicate and exchange information with each other. The iterative architecture is applied to a coupled power system in [116].

Besides the class of DMPC for linear systems, nonlinear dynamics have also been considered in DMPC. A dual decomposition approach is presented in [36], which reformulates a centralized into a distributed problem by linearization of the nonlinear dynamics. In [110] a nonlinear, non-convex algorithm is proposed that improves the system-wide objective and guarantees feasible solutions during the algorithms iterations.

In the following, the work considers iterative DMPC due to several limitations of non-iterative DMPC, such as the difficulty of finding a feasible initialization and the conservative performance. While iterative DMPC is very costly due to the large amount of communication and optimization steps, this chapter combines DMPC with the homotopy control method i.e. `OnPeHCA` from the previous chapter. Thus, while iterating, we can take advantage of finding a global optimal solution on the one side, while on the other by low computational effort in each iteration due to `OnPeHCA`.

9.2. Problem Definition

In a multi-agent scenario, collision avoidance is achieved, if each agent plans its trajectory in such a way, that the occupied space of the other agents is taken into account during the planning procedure. While a centralized formulation of such large optimization problems can quickly become intractable, especially for collision avoidance problems, a separation into smaller, more tractable problems is a promising approach. Thus, a decentralized control framework, in which each agent solves its collision avoidance problem with its local circumvention procedure, is in the focus of this chapter.

Consider $\nu \in \Sigma := \{1, ..., |\Sigma|\}$ controllable agents. Each agent ν occupies a polytopic space $\underset{[\nu]}{\mathcal{P}}_{x,k+j|k}$. As described in Sec. 8.2, each polytope is approximated by a finite set of particles $\kappa \in \mathcal{K} := \{1, ..., |\mathcal{K}|\}$ positioned equally to each other.

From the perspective of an agent ν, all other polytopes are enlarged $\underset{[\eta]}{\bar{\mathcal{P}}}_{x,k+j|k}$,

$\{\nu, \eta\} \in \Sigma, \nu \neq \eta$. The enlargement is described in Sec. 8.2 and is done in order to avoid collisions in the space between the polytopes not covered by the particles.

The system dynamics is given in the following definition:

Definition 9.1. *Given the nonlinear, continuous time system of particle κ of agent ν:*

$$\dot{x}_{\{\kappa\},[\nu]}(t) = f(x_{\{\kappa\},[\nu]}(t), u_\nu(t)), \tag{9.1}$$

as e.g. given by the dynamics of the TCP or any other particle on a robot manipulator $\nu \in \Sigma$ (see (7.68)). The dynamics of a particle $x_{\substack{k+j|k \\ \{\kappa\},[\nu]}} \in \mathcal{P}_{\substack{x,k+j|k \\ [\nu]}}$ is then given by the linearization and time-discretization of the nonlinear system (9.1) at particle $x_{\substack{k+j|k \\ \{\kappa\},[\nu]}}$, with $\kappa \in \mathcal{K}$, $\nu \in \Sigma$, and input value $u_{\substack{k|k-1 \\ [\nu]}}$:

$$x_{\substack{k+j+1|k \\ \{\kappa\},[\nu]}} = A_{\substack{:|k \\ \{\kappa\},[\nu]}} \, x_{\substack{k+j|k \\ \{\kappa\},[\nu]}} + B_{\substack{:|k \\ \{\kappa\},[\nu]}} \, u_{\substack{k+j|k \\ [\nu]}} + r_{\substack{:|k \\ \{\kappa\},[\nu]}}. \tag{9.2}$$

To each particle $x_{\substack{k+j|k \\ \{\kappa\},[\nu]}}$ of agent ν the same input $u_{\substack{k+j|k \\ [\nu]}}$ is applied.

Collision Avoidance Condition for Multi-Agents

In a distributed control framework, communication between the agents is established to achieve a better closed-loop performance than for decentralized control. Thus each controller plans its own solution with respect to the information obtained from the other agents and finally communicates its solution. While the determination of an agents trajectory depends on the occupied space of the other agents, the control problem is said to be coupled through the collision avoidance constraints. Hence, the goal of an agent $\nu \in \Sigma$ is to determine an optimized homotopic target trajectory in consideration of the spaces $\mathcal{P}_{\substack{x,k+j|k \\ [\eta]}}$ occupied by the other agents $\eta \in \Sigma$ with $\eta \neq \nu$. The task of collision avoidance for all $j \in \mathcal{J}_\mathcal{H}$ is thus:

$$\bigwedge_{\{\nu,\eta\}\in\Sigma,\nu\neq\eta} \mathcal{P}_{\substack{x,k+j|k \\ [\nu]}} \cap \mathcal{P}_{\substack{x,k+j|k \\ [\eta]}} = \emptyset. \tag{9.3}$$

With respected to the particle approximation and the obstacle enlargement (see Sec. 8.2), the collision avoidance condition (9.3) for an agent $\nu \in \Sigma$ is satisfied if for all $j \in \mathcal{J}_\mathcal{H}$, and all $\kappa \in \mathcal{K}$ the following condition holds:

$$x_{\substack{k+j|k \\ \{\kappa\},[\nu]}} \notin \bar{\mathcal{P}}_{\substack{x,k+j|k \\ [\eta]}}, \quad \forall \eta \in \Sigma \setminus \nu. \tag{9.4}$$

Global Objective for Cooperative Control

A distributed control setting can either be implemented non-cooperatively or cooperatively. In the non-cooperative setting, the goal of each agent is to optimize its local objective function. The key feature of cooperative control is that `OnPeHCA`

of each agent optimizes the same global cost function. This global cost function is built from local costs of each agent. A local cost function of an agent $\nu \in \Sigma$, and a reference particle $\kappa \in \mathcal{K}$ (e.g. a particle that describes the TCP of a robotic manipulator) depends on the homotopy parameter $\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k}$ of agent ν. The homotopy states and inputs of agent ν are given by:

$$\underset{\{\kappa\},[\nu]}{x}_{k+j|k}(\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k}) = \underset{\{\kappa\},[\nu]}{x^0}_{k+j|k} + \underset{\{\kappa\},[\nu]}{D}_{x,k+j|k}\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k},\ \forall j \in \mathcal{J}_{\mathcal{H}} \tag{9.5}$$

$$\underset{[\nu]}{u}_{k+j|k}(\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k}) = \underset{[\nu]}{u^0}_{k+j|k} + \underset{[\nu]}{D}_{u,k+j|k}\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k},\ \forall j \in \mathcal{J}_{\mathcal{H}}. \tag{9.6}$$

The costs for a particle κ of agent ν with goal state $\underset{\{\kappa\},[\nu]}{x^0}_{f|k}$ and input $\underset{[\nu]}{u^0}_{f|k}$ depends on the homotopy parameter $\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k}$:

$$\begin{aligned}\underset{\{\kappa\},[\nu]}{\phi}_{:|k} =& ||\underset{\{\kappa\},[\nu]}{x}_{k+H|k}(\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k}) - \underset{\{\kappa\},[\nu]}{x^0}_{f|k}||_{Q_{end}} \\ &+ \sum_{j=0}^{H-1} ||\underset{\{\kappa\},[\nu]}{x}_{k+j|k}(\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k}) - \underset{\{\kappa\},[\nu]}{x^0}_{f|k}||_Q + ||\underset{[\nu]}{u}_{k+j|k}(\underset{[\nu]}{\boldsymbol{\lambda}}_{:|k}) - \underset{[\nu]}{u^0}_{f|k}||_R.\end{aligned} \tag{9.7}$$

The global costs are evaluated for one particle of each agent (e.g. the particle describing the TCP of a robotic manipulator) and are given by the weighted sum of local costs with weights $\omega_{[\nu]} \in \mathbb{R}_{>0}$ satisfying the following condition:

$$\sum_{\nu\in\Sigma} \omega_{[\nu]} = 1. \tag{9.8}$$

The global cost function is given by:

$$\underset{\{\kappa\}}{\phi}_{:|k} = \sum_{\nu\in\Sigma} \omega_{[\nu]} \underset{\{\kappa\},[\nu]}{\phi}_{:|k}, \tag{9.9}$$

Thus, `OnPeHCA` of agent ν determines a circumventing homotopic target trajectory $\underset{[\nu]}{\boldsymbol{\lambda}^*}_{:|k}$, with " $*$ " denoting the solution by minimizing the its local costs in (9.9). Due to communication and iteration, the `OnPeHCA` algorithm minimizes the global costs in (9.9). The information about the other agents trajectories $\underset{[\eta]}{\boldsymbol{\lambda}}_{:|k}$, $\eta \neq \nu$, and thus their occupied spaces $\underset{[\eta]}{\bar{\mathcal{P}}}_{x,k+j|k}$, is obtained from communication.

9.3. Cooperative Homotopic Control Algorithm

The here considered iterative procedure for cooperative control lets each agent first determine an optimal homotopic target trajectory based on the information of occupied spaces. The results are communicated to all agents and the optimization is triggered again based on new obstacle information. This procedure is done for several *iterations* p during one sampling time k. The number of communication p during one sample time k has an effect on the decrease of the global costs per iteration.

Remark 9.1. *The agents only share their determined trajectories. No dynamic models of the other agents are available to one agent. This leads to the fact that an agent can only control its own trajectory in the optimization since the other obstacle trajectories are regarded as given.*

The remark necessarily leads to the question: If an agent can only control his own trajectory to evade from other obstacles, how do cooperation takes place? Even if the agent would lower the global costs by optimizing his own trajectory, the global costs could however be decreased more if another agent instead could improve his trajectory. Let the following definitions be given:

Definition 9.2. *The optimal homotopic trajectory for agent ν obtained in iteration p is denoted by $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}})$. This solution minimizes the global cost with obstacle information known from all agents at $p-1$.*

Definition 9.3. *The final homotopic trajectory obtained in iteration p and agreed among the agents is denoted by $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}})$. This trajectory is communicated between the agents and is used as starting point for the calculation in $p+1$. An agreed homotopic trajectory is a trajectory with a certain step width towards to the optimal solution $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}})$ starting from the solution at iteration $p-1$.*

The answer to the question is to let first each agent determine an optimal trajectory by `OnPeHCA` in iteration p denoted by $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}})$ and then to shift the known solution from $p-1$, $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}})$ by the corresponding weighting factor $\omega_{[\nu]}$ of the global cost function. The shifting is done in direction of the optimal trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}})$. The equation for shifting describes a convex combination:

$$\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k\\ [\nu]}}) = \omega_{[\nu]}\ \hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}}) + (1-\omega_{[\nu]})\ \hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}}). \tag{9.10}$$

The index "#" describes that $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k\\ [\nu]}})$ is an intermediate trajectory which is convexly combined by the weight. This intermediate solution again is convexly combined by a weighting factor $\varsigma \in [0,1]$ that controls the step width between the intermediate solution $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k\\ [\nu]}})$ and the known solution $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}})$ from $p-1$:

$$\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}}) = \varsigma\ \hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k\\ [\nu]}}) + (1-\varsigma)\ \hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}}). \tag{9.11}$$

The factor ς controls the iteration progress. If the step width ς is too small, the number of iterations p increases until the optimal cooperative solution is reached. A proof will be shown in the later. If the step width is to large on the other

hand, a suboptimal solution can be reached. This occurs if the last step leads to an infeasible solution (collision between agents) and thus the algorithm terminates with the feasible solution from the iteration before. To achieve termination, a maximum number of iterations is specified:

$$p \leq p_{max}. \tag{9.12}$$

Simultaneously, the iterations terminate if the cost change $\Delta\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}$ does not exceed a given threshold $\epsilon \in [0,1]$:

$$\Delta\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}} = \frac{|\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}} - \phi^{(p-1)}_{\substack{:|k\\ \{\kappa\}}}|}{\phi^{(p-1)}_{\substack{:|k\\ \{\kappa\}}}} \geq \epsilon. \tag{9.13}$$

The following example demonstrates the problem if the procedure e.g. iterates with a too large ς: Consider an agent ν that determines its optimal homotopic target trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}})$, based on the trajectories of the other agents $\eta \in \Sigma \setminus \nu$ from the last iteration $p-1$. From these trajectories the occupied spaces are also known $x_{\substack{k+j|k\\ \{\kappa\},[\eta]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\eta]}}) \in \bar{\mathcal{P}}^{*(p-1)}_{\substack{x,k+j|k\\ [\eta]}}$, since the polytope encloses all particles of an agent. The situation can occur that trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}})$ of agent ν collides with the simultaneously determined trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\eta]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\eta]}})$ of an other agents. The reason is, that all determinations are based on informations from $p-1$. Thus a full step toward the optimal solution leads to collisions between the agents. This circumstance is shown in Fig. 9.1 for two robotic manipulators. The left robot (Agent 1) determines its homotopic target trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[1]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [1]}})$ (blue) with the information of the second agent at $p-1$ (which is the black trajectory of Agent 2). The second agent vice versa does the same and determines $\hat{x}_{\substack{:|k\\ \{\kappa\},[2]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [2]}})$ (red). With respect to the informations of the trajectories from the previous iteration, both trajectories would be free of collisions. But it can be seen that both resulting trajectories are in conflict for the current iteration p. The determination of *agreed homotopic trajectories* with a smaller step width ς now solves this conflict.

Ensuring Feasibility during Iterations

With iteration progress the problem arises that an iteration p will be reached, in which it is no longer possible to determine *agreed homotopic trajectories* $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}})$, (9.11). The reason is that with an advanced iteration p, the agents trajectories come so close to each other, that a collision between the agents will occur in the next iteration. In this case, all agents jointly decide which agent reduces the costs the

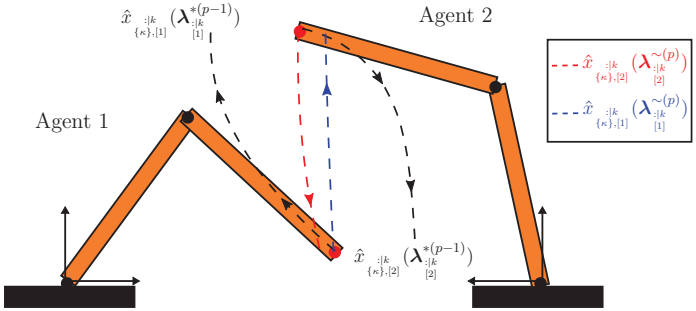


Figure 9.1.: Conflicts arising during the computation of homotopic target trajectories in iteration p, based on information from $p-1$.

most and give priority to this agent, while the remaining agents keep their old trajectories.

To achieve this, a set $\mathcal{Z}$ of cost values $\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}$ is determined. This set evaluates all combinations in which one agent $\nu \in \Sigma$ is selected that uses its *agreed homotopic trajectory* $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}})$, while the other agents $\eta \in \Sigma \setminus \nu$ are forced to use their previously agreed trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\eta]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\eta]}})$:

$$\mathcal{Z} := \{\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}} | \phi^{(p)}_{\substack{:|k\\ \{\kappa\}}} = \sum_{\nu\in\Sigma} \omega_{[\nu]} \phi^{(p)}_{\substack{:|k\\ \{\kappa\},[\nu]}},\ \nu : \boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}},\ \forall \eta \in \Sigma \setminus \nu : \boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\eta]}}\}. \tag{9.14}$$

Selecting from (9.14) the agent that reduces the cost most is denoted by ν^*:

$$\nu^* := \arg(\min \mathcal{Z}). \tag{9.15}$$

Thus, ν^* uses the trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu^*]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu^*]}})$, while the other agents $\eta \in \Sigma \setminus \nu^*$ carry out $\hat{x}_{\substack{:|k\\ \{\kappa\},[\eta]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\eta]}})$. These points are realized in Alg. 9.1.

Algorithmic Procedure - `COnPeHCA`

Important for collision avoidance in a distributed setting are feasible, collision-free initial trajectories $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(0)}_{\substack{:|k\\ [\nu]}})$ to each agent $\nu \in \Sigma$. For the explanation of the algorithm we assume that initial solutions are known here. The exact determination of these initial trajectories for iteration $p = 0$ is explained later.

Alg. 9.1 is executed in step k and returns the agreed final homotopic target trajectories $\hat{x}^*_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}})$ of each agent. These trajectories are executed for one time

step, corresponding to a receding horizon scheme. With the given number of maximum iterations p_{max} and the threshold $\Delta\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}} \geq \varepsilon$ for the cost change (9.13), the number of executions in the while-loop of Alg. 9.1 corresponds to the number of communications between the agents. The intermediate solutions $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k\\ [\nu]}})$ are first determined for each agent. Therefore the local `OnPeHCA` (with global objective (9.9)) of an agent determines first the homotopic target trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}})$ in consideration of the other agents (*line 4-5*) and then the intermediate trajectory $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k\\ [\nu]}})$ according to (9.10) (*line 6*). Subsequently, the determined intermediate trajectories with their occupied spaces $x_{\substack{k+j|k\\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k\\ [\nu]}}) \in \bar{\mathcal{P}}^{\#(p)}_{\substack{x,k+j|k\\ [\nu]}}$ (resulting from the convex hull of particles) are checked against any collisions with other agents (*line 7*). If no collisions exist, the intermediate solutions are the *agreed homotopic trajectories* with $\varsigma = 1$ determined by (9.11) (*line 9*). In the case of collisions, the step width is set to a value $\varsigma = \underline{\varsigma} < 1$ and the *agreed homotopic trajectories* are determined with this value according to (9.11) (*line 13*).

Finally the trajectories are again checked against collisions because of the trajectory shifting by ς (*line 14*). If no collisions exist, the determined trajectories are executed for k. In the case of collisions after shifting with ς, the conflict is solved by preferring one agent according to (9.14) and (9.15), such that the preferred agent executes its *agreed homotopic trajectory* $\hat{x}_{\substack{:|k\\ \{\kappa\},[\nu^*]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu^*]}})$. Meanwhile the other agents $\eta \in \Sigma \setminus \nu^*$ execute their *agreed homotopic trajectory* $\hat{x}_{\substack{:|k\\ \{\kappa\},[\eta]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\eta]}})$ from the previous iteration (*line 15-18*). Finally, after termination of the while-loop (*line 3-21*), each agent executes the first step of its *agreed homotopic trajectory* and the procedure repeats for $k+1$ (*line 22*).

Algorithm 9.1.: Cooperative Online Particle Extended Homotopic Control Algorithm (`COnPeHCA`)

Given: initial, feasible solutions $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(0)}_{\substack{:|k \\ [\nu]}})$, $\forall \nu \in \Sigma$, p_{max}, k
Set: $\Delta\phi^{(0)}_{\substack{:|k \\ \{\kappa\}}} = \infty$, $\varepsilon = 0.1$, $\underline{\varsigma} = 0.1$, $p = 0$
while $p \leq p_{max}$ & $\Delta\phi^{(p)}_{\substack{:|k \\ \{\kappa\}}} \geq \varepsilon$ **do**
    communicate/exchange trajectories $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k \\ [\nu]}})$ between all agents
    determine the homotopic target trajectories $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{(p)}_{\substack{:|k \\ [\nu]}})$ of all agents with their local `OnPeHCA`'s
    determine the intermediate solutions $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k \\ [\nu]}})$ acc. to (9.10)
    if collisions between agents with intermediate solutions $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:|k \\ [\nu]}})$
      exist **then**
        $\varsigma = \underline{\varsigma}$
    else
        $\varsigma = 1$
    end if
    determine the *agreed homotopic trajectories* $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k \\ [\nu]}})$ with ς
    if collisions between agents with solutions $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k \\ [\nu]}})$ exist **then**
        solve collision conflict by identifying a preferred agent ν^* acc. to (9.14) and (9.15), and set:
          (a) for agent $\nu^* : \hat{x}_{\substack{:|k \\ \{\kappa\},[\nu^*]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k \\ [\nu^*]}})$
          (b) and agents $\eta \in \Sigma \setminus \nu^* : \hat{x}_{\substack{:|k \\ \{\kappa\},[\eta]}}(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k \\ [\eta]}})$
    end if
    compute $\Delta\phi^{(p)}_{\substack{:|k \\ \{\kappa\}}}$ acc. to (9.13)
    set $p := p + 1$
end while
execute first step of the determined trajectories $\hat{x}^*_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k \\ [\nu]}})$ by the nonlinear system (3.1)

Initial Trajectories for `COnPeHCA`

Alg. 9.1 determines *agreed homotopic trajectories* by using feasible, collision-free trajectories of all agents from the previous iteration. Thus the algorithm has to be initialized with feasible trajectories $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(0)}_{\substack{:|k \\ [\nu]}})$ for all agents ν. From the `OnPeHCA`

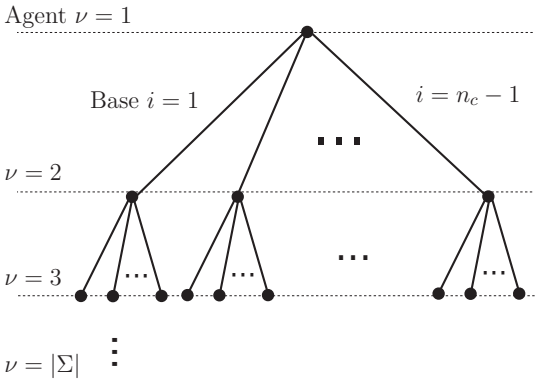


Figure 9.2.: All combinations of *base trajectories* of the agents represented as tree: A feasible solution of the tree provides the initial solutions $\hat{x}_{\substack{:|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{*(0)}_{\substack{:|k \\ [\nu]}})$, $\forall \nu \in \Sigma$

algorithm of an agent, it is known that the algorithm starts by determining a set of *base trajectories* $\boldsymbol{\lambda}^{i}_{\substack{:|k \\ [\nu]}}$, $i = \{1, ..., n_c\}$. The initialization of the algorithm is carried out by selecting a *base trajectory* $\boldsymbol{\lambda}^{i}_{\substack{:|k \\ [\nu]}}$ for each agent $\nu \in \Sigma$, such that no collisions occur or respectively that (9.3) holds for the linearized dynamics (9.2).

With the number $|\Sigma|$ of agents and the number of *base trajectories* n_c, the total number possible combinations for selection is $|\Sigma|^{n_c-1}$. The question that arises is which combination of initial, collision-free *base trajectories* is a reasonable choice. Therefore, different criteria can be used. e.g. minimal global costs for the initial trajectory, or simply collision-free trajectories. Therefore, all possible combinations of *base trajectories* are modeled as a tree, see Fig. 9.2. Each tree-level represents an agent ν. An edge outgoing from a vertex describes the decision for a *base trajectory* i. Hence, each vertex of the tree has $n_c - 1$ edges, and the tree has $|\Sigma|$ levels. A vertex of the tree represents a *base trajectory* corresponding to an agent. In order to quickly obtain an initialization of *base trajectories* in `COnPeHCA`, the tree is searched by a depth-first search for a simple collision-free initialization. The result is an assignment of a *base trajectory* to each agents initial solution.

$$\boldsymbol{\lambda}^{i}_{\substack{:|k \\ [\nu]}} \rightarrow \boldsymbol{\lambda}^{*(0)}_{\substack{:|k \\ [\nu]}} \tag{9.16}$$

Convergence to Local Optimum in `COnPeHCA`

With respect to the convergence results for convex problems in iterative DMPC [116], this part shows that `COnPeHCA` converges to a locally optimal solution by

iterative communication. The global cost function (9.9), at iteration p is a function depending on variables $\underset{[\nu]}{\boldsymbol{\lambda}^{*(p)}_{:|k}}$ of the *agreed homotopic trajectories*, such that one can write:

$$\underset{\{\kappa\}}{\phi^{(p)}_{:|k}}\left(\underset{[1]}{\boldsymbol{\lambda}^{(p)}_{:|k}},...,\underset{[\nu]}{\boldsymbol{\lambda}^{(p)}_{:|k}},...,\underset{[|\Sigma|]}{\boldsymbol{\lambda}^{(p)}_{:|k}}\right). \tag{9.17}$$

The convergence to a local optimum of the algorithm is given by the proof of the following lemma.

Lemma 9.1. *Given* `COmPeHCA`*, the sequence of the global cost functions, describing the costs towards the target states* (9.17)*, is for all* $\varsigma \in]0,1]$ *decreasing over* p *if no collisions between the agents occur.*

Proof. Since the global cost function (9.17) is a convex combination of local cost functions, when an agent ν optimizes its trajectory by determining a homotopic target trajectory $\underset{[\nu]}{\boldsymbol{\lambda}^{\sim(p)}_{:|k}}$, the costs are lower than if the agent would execute its *agreed homotopic trajectory* $\underset{[\nu]}{\boldsymbol{\lambda}^{*(p)}_{:|k}}$. The cost of the latter is, in turn, lower than the costs from the previous iteration with homotopy parameter $\underset{[\nu]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}$:

$$\begin{aligned}\underset{\{\kappa\}}{\phi^{(p)}_{:|k}}\left(\underset{[1]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}},...,\underset{[\nu]}{\boldsymbol{\lambda}^{\sim(p)}_{:|k}},...,\underset{[|\Sigma|]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}\right) &\leq \underset{\{\kappa\}}{\phi^{(p)}_{:|k}}\left(\underset{[1]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}},...,\underset{[\nu]}{\boldsymbol{\lambda}^{*(p)}_{:|k}},...,\underset{[|\Sigma|]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}\right)\\ &\leq \underset{\{\kappa\}}{\phi^{(p-1)}_{:|k}}\left(\underset{[1]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}},...,\underset{[\nu]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}},...,\underset{[|\Sigma|]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}\right).\end{aligned} \tag{9.18}$$

The objective (9.17) is a function of homotopy parameters. Setting (9.10) into (9.11) yields:

$$\underset{\{\kappa\},[\nu]}{\hat{x}_{:|k}}(\underset{[\nu]}{\boldsymbol{\lambda}^{*(p)}_{:|k}}) = \varsigma\omega_{[\nu]}\underset{\{\kappa\},[\nu]}{\hat{x}_{:|k}}(\underset{[\nu]}{\boldsymbol{\lambda}^{\sim(p)}_{:|k}}) + \varsigma(1-\omega_{[\nu]})\underset{\{\kappa\},[\nu]}{\hat{x}_{:|k}}(\underset{[1]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}) + (1-\varsigma)\underset{\{\kappa\},[\nu]}{\hat{x}_{:|k}}(\underset{[\nu]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}). \tag{9.19}$$

For a discrete point of time k of the convexly combined trajectory (9.19) and by inserting the homotopy equation (9.5), one obtains:

$$x_{\underset{\{\kappa\},[\nu]}{:|k}}(\underset{[\nu]}{\boldsymbol{\lambda}^{*(p)}_{:|k}}) = \varsigma\omega_{[\nu]}(\underset{\{\kappa\},[\nu]}{x^0_{k+j|k}} + \underset{\{\kappa\},[\nu]}{D_{x,k+j|k}}\underset{[\nu]}{\boldsymbol{\lambda}^{\sim(p)}_{:|k}}) + \varsigma(1-\omega_{[\nu]})(\underset{\{\kappa\},[\nu]}{x^0_{k+j|k}} + \underset{\{\kappa\},[\nu]}{D_{x,k+j|k}}\underset{[\nu]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}) \tag{9.20}$$

$$+ (1-\varsigma)(\underset{\{\kappa\},[\nu]}{x^0_{k+j|k}} + \underset{\{\kappa\},[\nu]}{D_{x,k+j|k}}\underset{[\nu]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}). \tag{9.21}$$

$$= \underset{\{\kappa\},[\nu]}{x^0_{k+j|k}} + \underset{\{\kappa\},[\nu]}{D_{x,k+j|k}}(\varsigma\omega_{[\nu]}\underset{[\nu]}{\boldsymbol{\lambda}^{\sim(p)}_{:|k}} + \varsigma(1-\omega_{[\nu]})\underset{[\nu]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}} + (1-\varsigma)\underset{[\nu]}{\boldsymbol{\lambda}^{*(p-1)}_{:|k}}). \tag{9.22}$$

Since the cost function only changes by variations in the homotopy parameter $\boldsymbol{\lambda}$, the affine terms $x^0_{\substack{k+j|k\\ \{\kappa\},[\nu]}}$ and $D_{\substack{x,k+j|k\\ \{\kappa\},[\nu]}}$ of (9.22) can be omitted when describing the costs:

$$\begin{aligned}
&\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [|\Sigma|]}}\right)\\
&\quad=\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\varsigma\omega_{[1]}\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [1]}}+\varsigma(1-\omega_{[1]})\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}}+(1-\varsigma)\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}},...,\right.\\
&\qquad\left.\varsigma\omega_{[|\Sigma|]}\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [|\Sigma|]}}+\varsigma(1-\omega_{[|\Sigma|]})\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}+(1-\varsigma)\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}\right),
\end{aligned}\tag{9.23}$$

and with $\sum_{\nu\in\Sigma}\omega_{[\nu]}=1$ (9.23) equals:

$$\begin{aligned}
&\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [|\Sigma|]}}\right)\\
&\quad=\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\varsigma\omega_{[1]}\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [1]}}+\varsigma\omega_{[2]}\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}}+...+\varsigma\omega_{[|\Sigma|]}\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}}+(1-\varsigma)\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}},\right.\\
&\qquad\varsigma\omega_{[1]}\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [2]}}+\varsigma\omega_{[2]}\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [2]}}+...+\varsigma\omega_{[|\Sigma|]}\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [2]}}+(1-\varsigma)\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [2]}},...,\\
&\qquad\left.\varsigma\omega_{[1]}\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}+\varsigma\omega_{[2]}\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}+...+\varsigma\omega_{[|\Sigma|]}\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [|\Sigma|]}}+(1-\varsigma)\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}\right).
\end{aligned}\tag{9.24}$$

Since the global cost function is convex, (9.24) can be rewritten to:

$$\begin{aligned}
&\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [|\Sigma|]}}\right)\\
=&\varsigma\sum_{\nu=1}^{|\Sigma|}\omega_{[\nu]}\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}\right)\\
&+(1-\varsigma)\phi^{(p-1)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}\right) &\text{(9.25a)}\\
&=\varsigma\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [|\Sigma|]}}\right)+(1-\varsigma)\phi^{(p-1)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}\right). &\text{(9.25b)}
\end{aligned}$$

The last equation shows that the costs of the new iteration p is a convex combination of the costs from the last iteration with the *agreed homotopic trajectories* $\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}}$ and the costs with homotopic target trajectories $\boldsymbol{\lambda}^{\sim(p)}_{\substack{:|k\\ [\nu]}}$. Since the left term of (9.25b) has always lowest costs (since all trajectories are choosen as the optimal), and $\varsigma\in]0,1]$, the costs for the new iteration p with *agreed homotopic trajectories* are always decreasing:

$$\phi^{(p)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p)}_{\substack{:|k\\ [|\Sigma|]}}\right)<\phi^{(p-1)}_{\substack{:|k\\ \{\kappa\}}}\left(\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [1]}},...,\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [\nu]}},...,\boldsymbol{\lambda}^{*(p-1)}_{\substack{:|k\\ [|\Sigma|]}}\right).\tag{9.26}$$

Thus (9.26) completes the proof. □

It has to be said, that the local convergence shown for the linearized model in this proof makes no statement about the convergence of the costs or stability when considering the nonlinear dynamics. The functionality and performance of `COnPeHCA` is evaluation by means of simulations.

9.4. Simulation Results for Multiple Cooperative Robots

The proposed `COnPeHCA` is applied to a system of two robotic manipulators, $\nu \in \Sigma = \{1, 2\}$. Each of them has two joints and two links and they operate in a 2-D Cartesian space. The manipulator dynamics corresponds to the model developed in Sec. 7.4.1 of Chapter 7. Adapted to the notation for a multi-agent scenario, the relevant equations from Sec. 7.4.1 are partially re-used here again. The nonlinear dynamics of an agent ν is given by:

$$M\left(\theta_{[\nu]}(t)\right)\ddot{\theta}_{[\nu]}(t) + C\left(\theta_{[\nu]}(t), \dot{\theta}_{[\nu]}(t)\right)\dot{\theta}_{[\nu]}(t) + G\left(\theta_{[\nu]}(t)\right) - u_{[\nu]}(t) = 0 \tag{9.27}$$

with configurations $\theta_{[\nu]} = (\theta_{\underset{[\nu]}{1}}(t), \theta_{\underset{[\nu]}{2}}(t))^T \in \mathbb{R}^2$ and inputs $u_{[\nu]}(t)$. According to (7.58) of Sec. 7.4.1, the dynamics of an agent ν linearized around the current state vector $\bar{\Theta}_{[\nu]}$ and input vector $\bar{u}_{[\nu]}$ provides with ZOH discretization:

$$\underset{[\nu]}{\Theta}_{k+j+1|k} = \underset{[\nu]}{A}_{\Theta,:|k}\,\underset{[\nu]}{\Theta}_{k+j|k} + \underset{[\nu]}{B}_{\Theta,:|k}\,\underset{[\nu]}{u}_{k+j|k} + \underbrace{f(\bar{\Theta}_{[\nu]}, \bar{u}_{[\nu]}) - \underset{[\nu]}{A}_{\Theta,:|k}\,\bar{\Theta} - \underset{[\nu]}{B}_{\Theta,:|k}\,\bar{u}_{[\nu]}}_{\underset{[\nu]}{r}_{\Theta,:|k}}. \tag{9.28}$$

As already described in Sec. 7.4.1, the model is then mapped from the configuration space into the Cartesian space by (7.59)-(7.69). This is done in Sec. 7.4.1 for one special point/particle of the robot, namely the tool center point (TCP), such that the resulting dynamics (7.68)-(7.69) describes the motion of the TCP in the Cartesian space. Here, the transformation into the Cartesian space is performed for different particles $\kappa \in \mathcal{K}$ distributed along the second link of each robot. Thus, the dynamics of an particle κ of an agent ν in Cartesian space is given by:

$$\underset{\{\kappa\}[\nu]}{\Psi}_{k+j+1|k} = \underset{\{\kappa\},[\nu]}{A}_{\Psi,:|k}\,\underset{\{\kappa\},[\nu]}{\Psi}_{k+j|k} + \underset{\{\kappa\},[\nu]}{B}_{\Psi,:|k}\,\underset{[\nu]}{u}_{k+j|k} + \underset{\{\kappa\},[\nu]}{r}_{\Psi,:|k}, \tag{9.29}$$

with the similarity transformations:

$$\underset{\{\kappa\},[\nu]}{A}_{\Psi,:|k} = \underset{\{\kappa\},[\nu]}{\Phi}_{:|k}\,\underset{[\nu]}{A}_{\Theta,:|k}\,(\underset{\{\kappa\},[\nu]}{\Phi}_{:|k})^{-1}, \tag{9.30}$$

$$\underset{\{\kappa\},[\nu]}{B}_{\Psi,:|k} = \underset{\{\kappa\},[\nu]}{\Phi}_{:|k}\,\underset{[\nu]}{B}_{\Theta,:|k}, \tag{9.31}$$

$$\underset{\{\kappa\},[\nu]}{r}_{\Psi,:|k} = \underset{\{\kappa\},[\nu]}{\Phi}_{:|k}\,\underset{[\nu]}{R}_{\Theta,:|k} + \underset{\{\kappa\},[\nu]}{\Omega}_{:|k} - \underset{\{\kappa\},[\nu]}{\Phi}_{:|k}\,\underset{[\nu]}{A}_{\Theta,:|k}\,(\underset{\{\kappa\},[\nu]}{\Phi}_{:|k})^{-1}\,\underset{\{\kappa\},[\nu]}{\Omega}_{:|k}, \tag{9.32}$$

with matrices $\Phi_{\substack{:|k\\{\kappa\},[\nu]}}$ and $\Omega_{\substack{:|k\\{\kappa\},[\nu]}}$ originating from (7.64). The number of particles is chosen to $|\mathcal{K}| = 10$ for each agent distributed along the second link of each robot. The task is to bring the two robotic manipulators from their initial to their final positions. The robots are placed as shown in Fig. 9.3. Since they may collide during their motion, `COnPeHCA` has to provide a cooperative solution that controlls the two robots free of collisions towards their final states. The initial states of the TCP's (denoted by particle $\{\kappa\} = 1$) are $\Psi^0{}_{\substack{0|0\\{1\},[1]}} = (1.4,\ 0,\ 0,\ 0)^T$ for the first agent, and $\Psi^0{}_{\substack{0|0\\{1\},[2]}} = (1.08,\ 1.4,\ 0,\ 0)^T$ for the second. The final states remain constant at $\Psi^0{}_{\substack{f|0\\{1\},[1]}} = (1.4,\ 1.4,\ 0,\ 0)^T$, and $\Psi^0{}_{\substack{f|0\\1,[2]}} = (1.08,\ 0,\ 0,\ 0)^T$. Additionally, limitations of the input torques are considered:

$$\underline{u}_{[\nu]} \leq u_{\substack{k+j|k\\[\nu]}} \leq \bar{u}_{[\nu]},\ \forall j \in \mathcal{J}.\forall \nu \in \Sigma \tag{9.33}$$

The maximum number of iterations in one time step k of `COnPeHCA`, is $p_{max} = 30$. The termination criterion for the cost change (9.13) is $\epsilon = 0.01$, and the step width each agent is allowed to move towards the homotopic target trajectory in one iteration, is chosen to be $\varsigma = 0.2$. The horizon is $H = 20$ time steps for each agent.

All trajectories are shown for the TCP of each agent, hence $\kappa = 1$. For clarity, only the online adapted *base trajectories* of agent $\nu = 1$ are shown in black dotted lines. They are determined as described in Sec. 7.2. In this case, there are totally four *base trajectories* per agent, which are determined based on the linearized dynamics (9.28) at step k. The red colored trajectories are the optimal homotopic target trajectories to the final state $\Psi^0{}_{\substack{f|k\\{1\},[\nu]}}$ of each agent for the case of neglecting the occupied space of the other agent. The magenta colored ones are the *agreed homotopic trajectories*, resulting from `COnPeHCA`. The blue trajectories show the entire motions of the TCP's, when applying the resulting inputs of `COnPeHCA` to the nonlinear robot dynamics (9.28).

It can be seen that the robots cooperate and closely pass each other to minimize the global cost function. The consideration of link collisions is realized by the particle approach which is included in `COnPeHCA`. When taking a closer look at Fig. 9.3a, it can be seen that the homotopic target trajectories of each agent (red) can not be followed free of collision. Thus, `COnPeHCA` provides iteratively solutions (magenta colored trajectories), which are free of collision. Especially at the beginning when the desired motions of the robots are in conflict with each other, the iterative optimization process takes place. As the movement of the robots progresses, the determined homotopic target trajectories (red) are no longer in conflict and can be selected directly, see $k = 6$, $k = 8$, $k = 10$, and $k = 30$. For the example of $k = 6$ (Fig. 9.3c), the magenta colored trajectories equal the red trajectories.

In Fig. 9.4a and Fig. 9.4a, the computation times of each manipulator are shown with respect to k. The yellow bars represent the portions of time spent for computing the base trajectories, while the blue bars refer to the iterative determination

Table 9.1.: Comparison of Costs for the Cooperative Control Problem

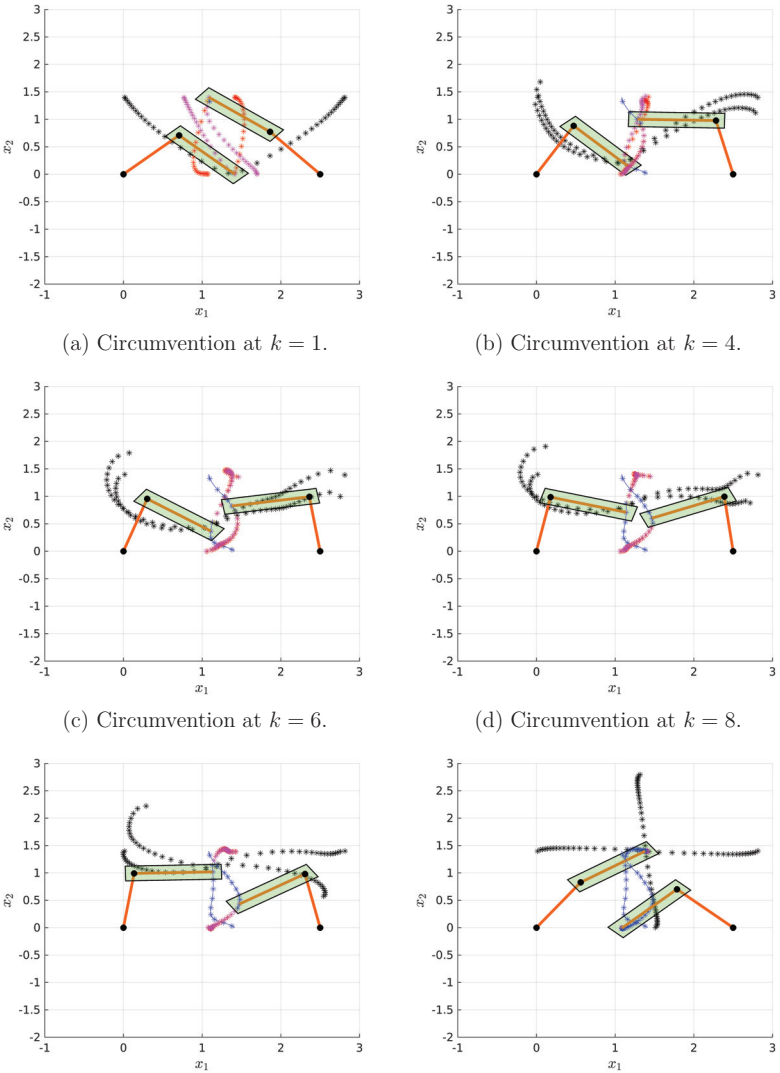
	Local Costs		
	Left Manipulator	Right Manipulator	Global Costs
`COnPeHCA` with $H = 20$	210	175	385
`COnPeHCA` with $H = 5$	229	258	487

of the *agreed homotopic trajectory*. It can be observed that the computation times of agent $\nu = 1$ are in general lower compared to that of the second agent. The reason is that the initialization of the trajectories by the deep-first search is not the optimal initialization. In this case, the right robot can not make a full step, with $\varsigma = 1$ as the left robot does. The computation times show that `COnPeHCA` can solve the cooperative control problem with consideration of collision avoidance quickly.

The number of iterations p performed in `COnPeHCA` for k are shown in Fig. 9.5. It can be seen, especially at the beginning, that a large number of communications between the manipulator controllers is performed until `COnPeHCA` terminates. In the later time steps, when a collision is not possible, the number of iterations decreases.

The separate costs of the two robotic manipulator, as well as the global costs are shown in Table. 9.1. For comparison, the costs for the same scenario but with a prediction horizon of $H = 5$ are also shown. The reduction of the prediction horizon obviously raises the costs because both robots detect each other later and therefore they have to preform a more rapid evasive movement. On the other hand, lower computation times are obtained when reducing the prediction horizon.

All simulations are performed in Matlab.



(a) Circumvention at $k = 1$.

(b) Circumvention at $k = 4$.

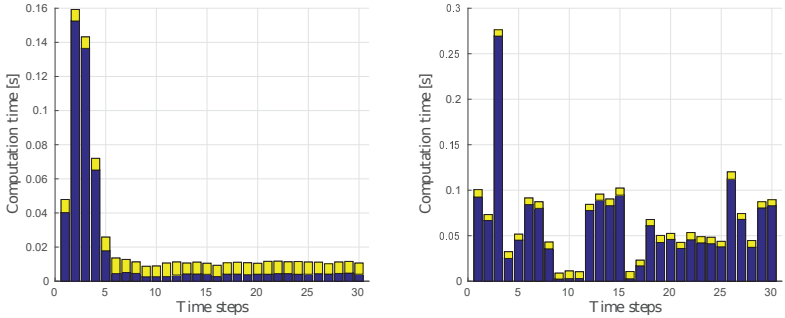
(c) Circumvention at $k = 6$.

(d) Circumvention at $k = 8$.

(e) Circumvention at $k = 10$.

(f) Circumvention at $k = 30$.

Figure 9.3.: Cooperative control of two robotic manipulators by the `COnPeHCA`, with prediction horizon $H = 20$. (black) *base trajectories* of the left robot, (magenta) the *agreed homotopic trajectories*, (red) homotopic target trajectories without considering the other robot, (green box) robot body.



(a) Left manipulator (b) Right manipulator

Figure 9.4.: Computation times for the iterative determination of the *agreed homotopic trajectory* of each manipulator, according to `COnPeHCA`: (yellow) time spent for *base trajectories*; (blue) time spent for the iterative process.

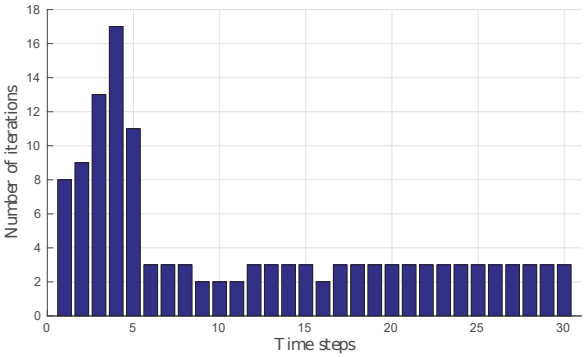


Figure 9.5.: Number of iterations p during each time step k of `COnPeHCA`.

9.5. Discussion of the Cooperative Control Method

The developed COnPeHCA solves a cooperative control problem by an iterative algorithmic procedure in which all agents first communicate their trajectories and subsequently optimize their trajectories based on the communicated new information. The goal of obtaining circumventing trajectories for each agent is quickly achieved by COnPeHCA. Inside COnPeHCA, the OnPeHCA algorithm is called in every iteration p and for each agent $\nu \in \Sigma$. It provides the basis for a quick determination of the trajectories. The computation time depends on two important factors: The first one is the computation time of OnPeHCA, which depends on the number of particles needed to describe the geometry and on the prediction horizon, hence, identified collision times. The second one is the number of iterations p spent in k. This can be strongly controlled by parameters like the step width ς for the determination of *agreed homotopic trajectories*, the threshold ϵ for the reduction of the cost rate or the maximum number of allowed iterations p_{max}. All of these parameters affect the quality of the solution and the computation times. Thus, it is important to find a good compromise in the parametrization such that the number of iterations is kept being small, while on the other hand the obtained solutions have low costs.

When iterations advance, the point comes in which no further iteration p can be made while the robots would collide in $p + 1$ with fixed ς. At this point the approach prefers one agent which reduces the costs most. The initialization of the trajectories for COnPeHCA also plays an important role for the computation times and the quality of the solutions. The proposed procedure of finding initial trajectories for each agent from the set of *base trajectories* is based on a depth-first search. This method has the advantage that initial trajectories for COnPeHCA can be found quickly. Unfortunately, this procedure can select initial trajectories that do not represent a good starting point for the iterative process where both agents can iterate with full step width. This circumstance can lead to increased computation times of single agents. This phenomena is observed for the left and right manipulator in Fig. 9.4a and Fig. 9.4b. The left manipulator determines its solutions much faster compared to the right one.

Since the dynamics of the considered example is highly nonlinear, problems arise when linearizing the dynamics at the current state like the increasing uncertainty of state prediction. This is because the nonlinearity depends on the state and the prediction is based on the linearization at the current robot configuration. This certainly leads to the fact that the planned trajectories can show significant deviations from those trajectories obtained with the computed inputs and the nonlinear dynamics. Nevertheless, by means of the receding horizon scheme in COnPeHCA with its repeated update of the linearization, the algorithm shows good results for the example with this highly nonlinear dynamics.

Part IV.

Conclusion

10. Conclusion and Future Research

10.1. Conclusion

This thesis provides a set of control procedures for point-to-point control problems by using homotopy properties to efficiently reach target states with low computational effort. Throughout this thesis, the individual investigations share the same property of solving a non-convex state space problem. The non-convexity can be understood to arise, e.g., from an obstacle that moves in the state space and prevents the system from directly moving to its target. While a possible approach is to solve such problems by techniques like MPC with embedded MIP, the computational effort is often impractical. The core idea of this thesis is to optimize over complete trajectories, parametrized by a homotopy function, rather than considering each point of time separately as done in MPC. Thus, this approach can save an enormous amount of computation time. The set of homotopic trajectories originate from a convex combination of so called *base trajectories*. The presented methods are applicable to linear and nonlinear systems.

The targeted system classes mainly affects the procedure of how *base trajectories* are specified. In case of linear systems with known initial states and a time-invariant target states, *base trajectories* can be already calculated offline. The method is characterized by first explicitly considering the obstacle in the online procedure and not already during the determination of the *base trajectories*. It was shown that controllers for the realization of transitions between homotopic trajectories can be synthesized offline by an LMI formulation. The separation of the problem into an offline and online part thus allows to reduce the online computation to a simple selection of a homotopic target trajectory that circumvents the obstacle. The execution of such a trajectory is realized by offline determined controllers. The online selection of a homotopic target trajectory is obtained from an algorithmic procedure which maps characteristic *passing points* of the obstacle from the state into the homotopy space. The method selects one *passing point* for which the overall trajectory, running through this *passing point*, has lowest costs. In the case where the dimension of the homotopy space is higher than the state space dimension $n_c > n_x$, the developed mapping algorithm `MapPas` avoids the multiple mapping of a point from the lower dimensional state space into the higher dimensional homotopy space. Nevertheless, the computation time depends on the number of *passing points*. Thus, a trade-of between computation time and quality of the circumventing trajectory exists. The more *passing points* exist, the more likely it is to find a better trajectory.

Compared to MPC using MIQP, the possible solution variety is always limited to those homotopic trajectories that can be obtained from the *base trajectories*. However, the simulation results show that the proposed method provides trajectories with comparably good results, but in considerably shorter computation time.

Since the developed method assumed $n_c > n_x$, due to inverse matrix conditions when transforming the dynamics into the homotopy space, it is obvious to extend the method to $n_x > n_c$. Particularly, $n_x > n_c$ covers a wide range of problems where a system model has a high state dimension, but the obstacle is only considered in a lower-dimension such that only a few *base trajectories* are needed. The associated problem, with $n_x > n_c$, is that the controller synthesized in the homotopy space becomes ambiguous when using in the "real" state dimension. Therefore this thesis derived orthogonality conditions, formulated as LMI's, such that the synthesized controllers force the system to only move on homotopic states. While a movement on this states is guaranteed by the controllers the collision avoidance, which was computed in the homotopy space, also hold in the state space where the system obviously exists in real. The derived conditions on the other side, require the system to be controllable in one step. Furthermore, input constraints were considered already in the controller synthesis by a new formulation of the cost function such that transitions between homotopic trajectories are quickly completed while using the input limitation. However, it has to be mentioned that the formulation may yields conservative controllers that are designed to guarantee stability for the transition between the most distant homotopic trajectories. This conservatism has the effect that for transition between near homotopic trajectories, the inputs do not use their full range. As a remedy, one can partition the delta homotopy system and switch between the controllers when the system enters a new partition. This pushes the input signal closer to its limits such that a circumvention becomes more likely when the obstacle is detected very late.

For the above-mentioned assumption of one-step controllability, it has to be mentioned that a large system class (e.g. the robot manipulator dynamics) does not meet this property. The controller synthesis for the homotopic system then would become impossible, since it is not possible to force the system to only move on homotopic states. The presented work-around is to force the state only for every second or third time step (depending on the rank of controllability), to be pushed on a homotopic state. Therefore an auxiliary system has been introduced which operates on a larger time scale. By designing the auxiliary system offline, the transition behavior is determined such that, input constraints can also be satisfied during online obstacle circumvention. The result of this part shows that the homotopy approach is able to provide circumventing trajectories for linear systems with different system properties like the level of controllability or the relation between the number of state dimension and homotopic base trajectories. To this different problem formulations, controllers are synthesized that allow to separate the problem into an online and offline part for a computationally efficient circumvention with low costs. Nevertheless, the algorithm and the controllers always react to the current known

position of the obstacle. Thus situations can occur in which a circumvention is not possible, e.g., in cases where the obstacle is detected very late and thus may be to close to the current system position.

A useful extension is to include predicted obstacle information into the approach of circumvention with homotopic trajectories to avoid collisions at an early stage. Therefore `HCA` was developed. `HCA` transforms the predicted obstacle location into the homotopy space (which was spanned by the base trajectories) and identifies possible times at which collisions may take place for all homotopic trajectories. Thus, the procedure condenses the circumvention problem to this critical times. `HCA` selects a homotopic trajectory by iteratively determining a non-colliding homotopic trajectory such that all critical times become free of collision. This is done by a tree-search procedure where on each level of the tree (a level represents a critical point of time) the homotopic trajectory is pushed out of the obstacle. Since a tree search can generally run into an infeasible branch, a mechanism was introduced that only known to be feasible branches are further explored by `HCA`. This is implemented by considering a collision-free *base trajectory* as a fallback strategy during the optimization in each tree node. This guarantees to find a circumventing trajectory if at least one collision-free *base trajectory* exists. The solution guarantee is an important aspect, especially for human-robot cooperation requiring save interaction. With respect to optimality of the solution provided by the tree-search, two main aspects of this procedure attract attention: The first is the choice of base trajectories. Since all homotopic trajectories, that are possible solution candidates are a product of the convex combination of the base trajectories, the solution can only be as good as the variety of homotopic trajectories allows. There is no special rule on how to choose these base trajectories. They are given by the designer and have to be chosen such that they span a sufficiently large space for circumvention. Thus, designing this trajectories strongly depends on the problem scenario, hence the obstacle size and the system dynamics. Secondly, the tree search procedure is implemented as a best-first search, which means that when the tree-search decides at an early stage to circumvent the obstacle from a certain direction. Then, the trajectory is further optimized in this direction. This may not provide the best possible solution. Alternatively, to find the best homotopic trajectory, the tree has to be explored completely or a cost-to-go heuristic has to be included in this method. The advantage is however, that a guaranteed collision-free and optimized solution can be found with low effort by `HCA`.

When considering nonlinear dynamics, which are locally approximated by linear dynamics, the system underlies a change of the dynamics when it traverses to the target state. As a result, the *base trajectories* have to be updated correspondingly to keep the error small between the linearized and the nonlinear dynamics. The repeated linearization has the consequence that the *base trajectories* also have to be updated at every point of time. The same holds if the target state for example is time-invariant. Then, the *base trajectories* have to direct to the new target state and thus have to be adapted. This leads to `OnHCA` which is a pure online method.

It contains an automated procedure for the determination and adaptation of *base trajectories*. The method places the end states of the *base trajectories* in a simplex around the overall target state and determines the *base trajectories* such, that they span a homotopic space for circumvention. However, a large area covered by the *base trajectories* may cause the system to move forward slowly, while *base trajectories* having a small area allow the system to move faster to the overall target state, but may not have a large enough space for circumvention. Since the obstacle is not taken into account for the calculation of the *base trajectories* but only during the determination of the homotopic target trajectories, the *base trajectories* can be determined with little effort. `OnHCA` was tested on a numerical example of a robotic manipulator dynamics, modeled in the configuration space, such that the robot circumvents a moving obstacle in the Cartesian space. Despite of the high system nonlinearity, the `OnHCA` provides a good and quick solution to this example.

Especially when dealing with collision avoidance between geometric bodies, as it is the case for the human-robot example, the consideration of a single point for the collision avoidance can not guarantee to prevent from collisions between the robotic links and the obstacle. The thesis combines a particle approach with `OnHCA`, resulting in `OnPeHCA`. The determination of circumventing homotopic trajectories is thus carried out for a variety of particles. An interesting aspect is that the homotopy parameter, thus the optimization variable, has the same effect on all particles, so that the dimension of the optimization problem, selecting the best homotopic trajectory, only depends on the number of *base trajectories*. Nevertheless, with an increasing number of particles the computational effort of `OnHCA` increases. The reason for that is that each particle, which is identified to be still in collision with the obstacle, is sequentially included into the optimization problem. Thus, the problem is solved again by considering the collision avoidance constraints from the already considered particles and the actually included one. In order to let the computational effort become not too large, a combination between the number of particles and an obstacle enlargement has been presented. It is further observed that the order with which colliding particles are sequentially added to the optimization problem affects the number of runs `OnPeHCA` has to perform for a collision-free solution over all particles.

The developed homotopic algorithm `OnPeHCA` has further been extended to solve distributed cooperative obstacle avoidance problems, yielding `COnPeHCA`. Consider, e.g, two robotic manipulator sharing a common workspace and fulfill a manufacturing task where they may come into a collision conflict while moving. To let the manipulators continue to move to their target positions, both robots have to evade each other. This can be done in a cooperative fashion where the solution trajectory to both robots separately is not optimal, but with respect to the overall motion cost of both together becomes the best.

`COnPeHCA` consists of an alternating sequence of communicating the homotopic trajectories between the agents, and an optimization of the solution trajectories based on the communicated information. During the optimization process, each

agent is allowed to move its last *agreed homotopic trajectory* a weighted distance toward the actually determined trajectory. In cases of collisions, hence when the weighted distance is not realizable, the algorithm tries to execute a smaller step. This approach is chosen since the weighted distance of each agent is determined with information of the trajectories from the other agents in the last time step. Thus an agent does not know the actual trajectories of the other agents, since all calculations take place simultaneously. The solution is then improved in an iterative procedure, which alternates between communication and optimization. The thesis also shows a mechanism to resolve conflicts when no further improvement is possible for all agents simultaneously in the same iteration of `COnPeHCA`. Then, an agent gets high priority if its changes reduce the costs most. An important factor influencing the quality of the solution as well as the running time of `COnPeHCA` is the choice of each agent's initial solution. The proposed depth-first strategy has the advantage of determining an initial solution fast but unfortunately this procedure can select initial trajectories that do not represent the optimal starting point for the iterative process. For example, one agent may always take full steps in the optimization phase of an iteration of `COnPeHCA`, while the second agent can only make smaller steps to avoid collisions for the given initialization.

Collision avoidance with bodies and also in a cooperative framework is a time consuming problem. The proposed method shows that such problems can be solved in real-time which is especially important when decisions have to be made fast like in robot assembling lines. On each agent of this procedure `OnPeHCA` operates with its strength of finding critical collision times and to optimize over complete trajectories than over each single point in time. Furthermore each `OnPeHCA` still guarantees that a collision free trajectory can be found if at least on base trajectory is free, which was initially found by `COnPeHCA`. `COnPeHCA` then unifies the fast local `OnPeHCA`'s by exchanging their solution trajectories. The advantage of the procedure is that besides the target state only parameters for the circumvention area have to be provided to `COnPeHCA` and the algorithm starts automatically spanning base trajectories and determining a cooperative solution to all agents.

10.2. Future Research

This thesis proposes different control approaches for different instances of non-convex control problems. Remaining open questions and promising research directions are as follows:

- While `OnHCA` uses linearization techniques, the possibility exists to include linearization errors by approximations like the Lagrangian remainder, to determine a collision-free, homotopic target trajectory. These error bounds could be included by enlarging the obstacle when determining circumventing trajectories. An extension of the presented method to linear parameter varying (LPV)

systems is conceivable. Similar to the particle approach in the `OnPeHCA` where the motion of the system is free of collision if all particles are, a collision-free homotopic target trajectory for a LPV system could be obtained by considering all possible motions of the LPV system.

- `OnHCA` evaluates collision-critical times by an algorithmic tree search procedure sequentially. The order in which these points are processed is simply by considering one node after another. If one would, however, find out the points in time which force the homotopic trajectory to be pushed out of the obstacle the most or if it would be possible to reduce the set of collision-critical times further, then the tree search completes faster. One possible approach could be to analyze for which critical times the state is far inside the obstacle and to organize the tree search with respect to these information. In the best case, the tree search could already be finished after one level of the tree.

- A similar task arises in the identification of a more preferable sequence on how identified colliding particles have to be processed in `OnPeHCA`. It would be helpful to reduce the number of particles by considering e.g. n-dimensional balls for the approximation of the system geometry.

- For `COnPeHCA`, a combination of initial collision-free *base trajectories* for each agent is found by depth-first search. Further research effort may be on how to select initial *base trajectories* to reduce the communication.

- An ambitions goal is to reduce the computation time of `COnPeHCA` by allowing as few communication iterations as possible, while still obtaining good solutions. One could think of a type of anytime algorithm in which the circumventing trajectories are optimized during execution by using results from previous time steps and from currently communicated information.

List of Symbols

Abbreviations

`BfSearch`	Best-first Search algorithm
`COnPeHCA`	Cooperative Online Particle Extended Homotopic Control Algorithm
`GenTarS`	Generation of Target States
`HCA`	Homotopic Control Algorithm
`MapPas`	algorithm mapping *passing points* p_l into the homotopy space
`OHTadd`	Optimal Homotopic Trajectory with additional constraints
`OHT`	Optimal Homotopic Trajectory
`OnBfSearch`	Online Best-first Search algorithm
`OnHCA`	Online Homotopic Control Algorithm
`OnOHTadd`	Online Optimal Homotopic Trajectory with additional constraints
`OnOHT`	Online Optimal Homotopic Trajectory
`OnPeHCA`	Online Particle Extended Homotopic Control Algorithm
TCP	Tool Center Point
ZOH	Zero-order hold

Functions

$\Xi(\Theta(t))$	nonlinear function for the mapping the combined vector $\Theta(t)$ in configuration space, into the combined vector $\Psi(t)$ in Cartesian space
$Bestnode(Node_{j_\iota})$	function identifying the best node $node^*_{j_\iota,w}$, from the set $Node_{j_\iota}$

$f : \mathbb{X} \times \mathcal{U} \to \mathbb{X}$	function of the system dynamics
f_{lb}	function representing a lower bound $f_{lb} : \mathbb{S}^{n_c}_{>0} \times \mathbb{S}^{n_c}_{>0} \to \mathbb{R}$
f_{ub}	function representing an upper bound $f_{ub} : \mathbb{S}^{n_c}_{>0} \times \mathbb{S}^{n_c}_{>0} \to \mathbb{R}$
$g^i(\tau)$	function $g^i : [0,1] \to \mathbb{R}^{n_x}$
$h(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s))$	function $h : [0,1] \times \mathbb{R}^{n_c} \to \mathbb{R}^{n_x}$
$h_{con}(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s))$	conic approximation of $h(\tau, \bar{\boldsymbol{\lambda}}(p_l, C_s))$
$J(x_k, u_k)$	performance function
$TCP(\theta(t))$	mapping the tool center point from configuration, into the Cartesian space $TCP : \mathbb{R}^2 \to \mathbb{R}^2$
$TimePlane(node_{j_1,w})$	function identifying time and half-plane of a node
$V_k(\boldsymbol{\lambda}_k)$	quadratic Lyapunov function $V_k : \mathbb{R}^{n_c} \to \mathbb{R}$
$V_k(x_k)$	quadratic Lyapunov function $V_k : \mathbb{R}^{n_x} \to \mathbb{R}$

General

$() \cdot ()$	dot product
$() \cdot ()$	value of particle κ, and agent ν at times $k+j$ determined from information at k
$(\cdot)_k$	value at time k
$\lceil \cdot \rceil$	round up of a value
$\lfloor \cdot \rceil$	determination of nearest integer value
$\lfloor \cdot \rfloor$	round off a value
$\mathbf{Co}\{\cdot\}$	convex hull of a finite set
$\mathbf{E}(\cdot)$	expected value
$\mathbf{rank}(M)$	rank of matrix M
$\mathbf{tr}(\cdot)$	trace of a matrix
$\mathcal{N}(\cdot)$	null space of a matrix or vector
∇	vector differential operator

$\partial\mathcal{P}_x$	boundary of polytope $\mathcal{P}_x$
$\|\cdot\|$	euclidean norm of matrix or vector, e.g.: $\|x\| := \sqrt{x^T x}$
$\|\cdot\|_Q$	weighted norm with weight Q, e.g.: $\|x\|_Q := \sqrt{x^T Q x}$
$node_{j_\mathfrak{t},w}$	node of a tree with tree-level $j_\mathfrak{t}$ and decision w

Scalars and Constants

$(\cdot)_{:\|k}$	value constant over future time determined from information at k
$(\cdot)_{k+j\|k}$	value at time $k+j$ determined from information at k
α_w	$w-th$ weighting value
α_{push}	pushing parameter $\alpha_{push} \in [0,1]$
$\bar{\Theta}$	linearization point for the robots configurations
$\bar{k}$	upper time bound
$\Delta\phi^{(p)}_{\substack{:\|k \\ \{\kappa\}}}$	rate of the cost change at iteration p
Δt	sampling time, i.e. time increment for each time step k
ϵ	threshold for $\Delta\phi^{(p)}_{\substack{:\|k \\ \{\kappa\}}}$
λ^i	i-th entry of vector $\boldsymbol{\lambda}$
$\mathfrak{r}$	indicator for number of time steps the system is controllable in
$\mathfrak{t}$	discrete time index
μ^i	i-th entry of vector $\boldsymbol{\mu}$
ν	agent of the the set of agents $\nu \in \Sigma$
ν^*	most preferable agent identified from the set $\mathcal{Z}$
$\omega_{[\nu]}$	weights for including local costs of agent ν into global cost function
$\phi_{\substack{:\|k \\ \{\kappa\},[\nu]}}$	local costs of an particle κ of agent ν
$\phi_{\substack{:\|k \\ \{\kappa\}}}$	global costs for a particle κ

τ	continuous variable $\tau \in [0, 1]$
τ_i	input torque of robotic manipulator
τ_{int}	integer value $\tau_{int} \in [\underline{k}, \bar{k}]$
$\upsilon,\ \beta,\ \varrho,\ \sigma$	backtracking parameters of algorithm `MapPas`
ς	parameter for convex combination $\varsigma \in [0, 1]$
$\lvert\mathcal{M}\rvert$	cardinality of the set $\mathcal{M}$
$\lvert\mathcal{P}_{pas}\rvert$	cardinality of the set $\mathcal{P}_{pas}$
b_f	binary variable $b \in \{0, 1\}$ indexed by f
c_i	diagonal element of matrix C_s, or friction parameter of robotic manipulator
g	gravitational constant
H	prediction horizon
j	discrete-time j
k	discrete-time k
m_i	mass
N	number of time steps
n_c	number of base trajectories from index set $\mathcal{M} := \{0, ..., n_c\}$
N_p	cardinality of $\mathcal{C}$, hence all possible permutations of C_s
n_u	dimension of the input vector $u \in \mathbb{R}^{n_u}$
n_v	number of vertices of $\mathcal{P}_{x,k}$
n_x	dimension of the state vector $x \in \mathbb{R}^{n_x}$
n_{pas}	number of *passing points* of $\mathcal{P}_{pas}$
n_{stat}	number of base trajectories with static end states
p, p_{max}	iteration p, maximum number of iterations p_{max}
q_i	$i-th$ element of matrix Q_C
t	continuous time t
T_{kin}	kinetic energy

$u_{max,s}$	$s-th$ element of vector $u_{max} \in \mathbb{R}^{n_u}$
V_k	quadratic Lyapunov function
V_{pot}	potential energy
y_i	$i-th$ diagonal element of matrix Q_C
$\underline{k}$	lower time bound

Sets

$\bar{\mathcal{P}}_{\substack{x,k+j\|k \\ [\nu]}}$	enlargement of $\mathcal{P}_{\substack{x,k+j\|k \\ [\nu]}}$
$\bar{\mathcal{P}}^{*(p)}_{\substack{x,k+j\|k \\ [\nu]}}$	occupied polytopic space of an agent ν, at iteration p, obtained from the particles of state $x_{\substack{k+j\|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{(p)}_{\substack{:\|k \\ [\nu]}})$
$\bar{\mathcal{P}}^{\#(p)}_{\substack{x,k+j\|k \\ [\nu]}}$	occupied polytopic space of an agent ν, at iteration p, obtained from the particles of state $x_{\substack{k+j\|k \\ \{\kappa\},[\nu]}}(\boldsymbol{\lambda}^{\#(p)}_{\substack{:\|k \\ [\nu]}})$
$\boldsymbol{\Lambda}$	$\mathcal{P}_{\boldsymbol{\lambda}pas}$ ordered according to costs $J(\boldsymbol{\lambda})$
$\hat{\mathcal{P}}_{\substack{x,k+j\|k \\ [\nu]}}$	enlargement of $\mathcal{P}_{\substack{x,k+j\|k \\ [\nu]}}$
$\Lambda(i)$	i-th element $\Lambda(i) \in \boldsymbol{\Lambda}$
$\mathbb{C}^n$	set of complex vectors with n elements
$\mathbb{N}_{\geq 0}$, $\mathbb{N}_{>0}$	set of non-negative natural numbers, set of positive natural numbers
$\mathbb{R}^n$	set of real vectors with n elements
$\mathbb{R}_{\geq 0}$	set of non-negative real numbers
$\mathbb{S}^n$, $\mathbb{S}^n_{>0}$	set of symmetric matrices of dimension n, set of positive symmetric matrices of dimension n
$\mathbb{X}$	state space constraint: $\mathbb{X} \subseteq \mathbb{R}^{n_x}$
$\mathbb{Z}$	set of natural numbers
$\mathbb{Z}_{>0}$	set of non-negative natural numbers
$\mathcal{C}$	set of permutation matrices C_s

$\mathcal{D}$	set of vertices, with elements $d_w \in \mathbb{R}^{n_c}$ from $\delta\boldsymbol{\lambda}$
$\mathcal{E}(P)$	ellipsoidal set centered at the origin and with shape matrix P
$\mathcal{G}$	set of vertices of $\boldsymbol{\lambda}$, $\mathcal{G} := \{\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_{2^{n_c}}\}$
$\mathcal{G}_{f\|k}$	unified set of target states $\mathcal{G}_{f\|k} = \mathcal{X}_{f\|k}^{stat} \cup \mathcal{X}_{f\|k}^{var}$
$\mathcal{I}$	subset of particles $\kappa \in \mathcal{K}$ which are in collision with an obstacle
$\mathcal{J}_{\mathcal{H}}$	set of time steps $\mathcal{J}_{\mathcal{H}} := \{0, ..., H\}$
$\mathcal{J}_{\mathcal{N}}$	set of time steps $\mathcal{J}_{\mathcal{N}} := \{0, ..., N\}$
$\mathcal{J}_{k^*}$	set of remaining time steps from k^*, $\mathcal{J}_{k^*} := \{0, \ldots, N-1-k^*\}$
$\mathcal{K}$	set of particles $\kappa \in \mathcal{K} = \{1, ..., \|\mathcal{K}\|\}$
$\mathcal{L}$	set of base trajectories $\mathcal{L} := \{\bar{\boldsymbol{\mu}}^0, ..., \bar{\boldsymbol{\mu}}^{n_c}\}$
$\mathcal{M}$	index set of base trajectories $\mathcal{M} := \{0, ..., n_c\}$
$\mathcal{O}$	set of identified optimal nodes: $node^*_{j_\mathfrak{t},w} \in Node_{j_\mathfrak{t}}$
$\mathcal{P}_x$	polytopic set in state space
$\mathcal{P}_{\boldsymbol{\lambda}pas}$	set of mapped *passing points* into the homotopy space
$\mathcal{P}_{\lambda,k+j}$	polytopic set in homotopy space at time $k+j$
$\mathcal{P}_{\lambda,k}$	polytopic set in homotopy space at time k
$\mathcal{P}_{\mu,\mathfrak{t}+j_\mathfrak{t}}$	polytopic set in homotopy space at time $\mathfrak{t}+j_\mathfrak{t}$
$\mathcal{P}_{\mu,\mathfrak{t}}$	polytopic set in homotopy space at time $\mathfrak{t}$
$\mathcal{P}_{pas}$	set of *passing points* p_l
$\mathcal{P}^{[\nu]}_{x,k+j\|k}$	polytopic obstacle of agent ν
$\mathcal{P}_{x,k+j}$	polytopic set in state space at time $k+j$
$\mathcal{P}_{x,k}$	polytopic set in state space at time k
$\mathcal{P}_{xv,k}$	vertex representation of $\mathcal{P}_{x,k}$
$\mathcal{T}_{\mathcal{H}}$	set of time steps $\mathcal{T}_{\mathcal{H}} := \{0, ..., \lfloor H/\mathfrak{t} \rfloor\}$
$\mathcal{T}_{\mathcal{I}}$	set of possible collision time steps between a homotopic state and an obstacle

$\mathcal{T}_\mathcal{N}$	set of time steps $\mathcal{T}_\mathcal{N} := \{0, ..., \lfloor N/\mathfrak{r} \rfloor\}$
$\mathcal{U}$	input constraint: $\mathcal{U} \subseteq \mathbb{R}^{n_u}$
$\mathcal{U}_b$	set of input base trajectories $\mathcal{U}_b := \{\hat{u}^0, ..., \hat{u}^{n_c}\}$
$\mathcal{W}_{\mathfrak{t}+j_\mathfrak{t}}$	index set of half-planes of polytope $\mathcal{P}_{\mu,\mathfrak{t}+j_\mathfrak{t}}$
$\mathcal{X}_{f\|k}^{stat}$	set of target states for base trajectories $\mathcal{X}_{f\|k}^{stat} := \{x_{f\|k}^i \mid i \in \{0, ..., n_{stat}\}\}$
$\mathcal{X}_{f\|k}^{var}$	set of constantly adapted target states for base trajectories $\mathcal{X}_{f\|k}^{var} := \{x_{f\|k}^i \mid i \in \{n_{stat}+1, ..., n_c\}\}$
$\mathcal{X}_b$	set of state base trajectories $\mathcal{X}_b := \{\hat{x}^0, ..., \hat{x}^{n_c}\}$
$\mathcal{Z}$	set of cost values
$\partial\mathcal{P}_{x,k+j\|k}$	boundary of $\mathcal{P}_{x,k+j\|k}$
ϕ_x^*	optimal state sequence $\phi_x^* = (x_{\mathfrak{t}\mathfrak{r}+0}^*, ..., x_{\mathfrak{t}\mathfrak{r}+H}^*)$
$\phi_x^{\bar{\boldsymbol{\mu}}^i}$	state sequence towards base trajectory $\bar{\boldsymbol{\mu}}^i$
$\phi_x^{\bar{\boldsymbol{\mu}}^{i^*}}$	state sequence towards optimal base trajectory $\bar{\boldsymbol{\mu}}^{i^*}$
Σ	set of agents $\nu \in \Sigma = \{1, ..., \|\Sigma\|\}$
$\tilde{\mathcal{P}}_{\mu,\mathfrak{t}+j_\mathfrak{t}}$, $\tilde{\mathcal{P}}_{\lambda,k+j}$	polytopic set in homotopy space without redundant half-planes
$\tilde{\mathcal{W}}_{\mathfrak{t}+j_\mathfrak{t}}$, $\tilde{\mathcal{W}}_{k+j}$	index set of half-planes of polytope $\tilde{\mathcal{P}}_{\mu,\mathfrak{t}+j_\mathfrak{t}}$, resp. $\tilde{\mathcal{P}}_{\lambda,k+j}$
$\triangle_{f\|k}^{n_{stat}-1}$	$(n_{stat}-1)$-simplex with vertices $\tilde{x}_{f\|k}^i$
$\|\mathcal{P}_{pas}\|$	cardinality of $\mathcal{P}_{pas}$
$Node_{j_\mathfrak{t}}$	set of nodes: $node_{j_\mathfrak{t},w}$

Vectors and Matrices

$\tilde{P}_{f\|k}$	orthogonal basis in reduced dimension, $P_{f\|k} \in \mathbb{R}^{n_{stat} \times n_{stat}}$, with entries $\tilde{p}(i,j)_{f\|k}$
$v_{f\|k}^i$	coordinates-transformed target state of i-th base trajectory, $v_{f\|k}^i \in \mathbb{R}^{n_x}$
$P_{f\|k}$	orthogonal basis $P_{f\|k} \in \mathbb{R}^{n_x \times n_x}$
$(\cdot)^{-1}$	regular inverse of a matrix

$(\cdot)^{\dagger}$	Moore–Penrose pseudoinverse of a vector or matrix
$(\cdot)^{H}$	Hermitian of a vector or matrix
$(\cdot)^{T}$	transposed of a vector or matrix
$\bar{\boldsymbol{\lambda}}$	homotopic target state
$\bar{\boldsymbol{\lambda}}(p_l, C_s)$	like $\bar{\boldsymbol{\lambda}}(p_l)$ but in a subspace C_s $in\mathcal{C}$
$\bar{\boldsymbol{\lambda}}(p_l)$	homotopic target state with state trajectory $\hat{x}(\bar{\boldsymbol{\lambda}}(p_l))$ traversing p_l
$\bar{\boldsymbol{\lambda}}^{*}$, $\bar{\boldsymbol{\mu}}^{*}$	optimal solution of the homotopic target state
$\bar{\boldsymbol{\lambda}}^{i}$	encodes the i-th base trajectory
$\bar{\boldsymbol{\lambda}}^{i^{*}}$	optimal base trajectory
$\bar{\boldsymbol{\lambda}}_{over}$	homotopic target state of the overdetermined system
$\bar{\boldsymbol{\lambda}}_{under}$	homotopic target state of the underdetermined system
$\bar{\boldsymbol{\mu}}$	homotopic target state
$\bar{\boldsymbol{\mu}}^{*}$	homotopic target trajectory
$\bar{\boldsymbol{\mu}}^{i}$	encodes the i-th base trajectory
$\bar{\boldsymbol{\mu}}^{i^{*}}$	optimal base trajectory
$\bar{u}_{add}$	input steady state of the additional input value at time k
$\bar{x}$, $\bar{u}$	steady state and input
$\boldsymbol{\gamma}_i$	i-th vertex of the set $\mathcal{G}$, with $\boldsymbol{\gamma}_i \in \mathbb{R}^{n_c}$
$\boldsymbol{\lambda}$	vector of homotopy parameters: $\boldsymbol{\lambda} = (\lambda^1, ..., \lambda^{n_c})^T$
$\boldsymbol{\lambda}_k$, $\boldsymbol{\lambda}_{:}$	homotopy vector at time k, constant homotopy vector over time
$\boldsymbol{\lambda}^{(p)}_{\substack{:\vert k \\ [\nu]}}$	encodes the homotopy target trajectory of agent ν, determined at iteration p
$\boldsymbol{\lambda}^{*(p)}_{\substack{:\vert k \\ [\nu]}}$	encodes the agreed homotopic trajectory of agent ν, determined at iteration p
$\boldsymbol{\lambda}^{\#(p)}_{\substack{:\vert k \\ [\nu]}}$	encodes the weight-dependent temporal trajectory of agent ν, determined at iteration p

$\boldsymbol{\lambda}^*_{:\|k}$	optimal homotopic target trajectory
$\boldsymbol{\lambda}^i_{:\|k}$	i-th base trajectory
$\boldsymbol{\mu}_{\mathfrak{t}}$	homotopy vector at time $\mathfrak{t}$, for $\mathfrak{r}$-step controllable system
$\delta\boldsymbol{\lambda}_k$	delta signal of the homotopy vector at time k
$\delta u_{add,k}$	delta signal of the additional input value at time k
$\delta\tilde{x}^0_{k\|k}$	direction vector $\delta\tilde{x}^0_{k\|k} \in \mathbb{R}^{n_{stat}}$, from current state $\tilde{x}_{k\|k}$ to target state $\tilde{x}^0_{f\|k}$
$\dot{\theta}(t)$	vector of configuration velocities $\dot{\theta}(t) = (\dot{\theta}_1(t), \dot{\theta}_2(t))^T$
$\hat{u}(\bar{\boldsymbol{\lambda}})$	homotopic target trajectory in input space
$\hat{u}(\boldsymbol{\lambda}_:)$	homotopic input trajectory with constant homotopy parameter over time
$\hat{u}^i$	i-th *base trajectory* $\hat{u}^i = (u^i_0, ..., u^{n_u}_N)$ with $i \in \mathcal{M}$
$\hat{x}(\bar{\boldsymbol{\lambda}})$	homotopic target trajectory in state space
$\hat{x}(\boldsymbol{\lambda}_:)$	homotopic state trajectory with constant homotopy parameter over time
$\hat{x}^i$	i-th *base trajectory* $\hat{x}^i = (x^i_0, ..., x^{n_x}_N)$ with $i \in \mathcal{M}$
$\mathbf{u}_i$	i-th column of matrix $\mathbf{U}_{k+1}$
$\mathbf{U}_{k+1},\ \mathbf{S}_{k+1},\ \mathbf{V}_{k+1}{}^T$	matrices of the singular value decomposition of $D_{x,k+1}$;
$\Omega_{:\|k}$	matrix from linearization of $\Xi(\Theta_{k+j\|k})$, $\Omega_{:\|k} \in \mathbb{R}^{4\times4}$
$\Phi_{:\|k}$	matrix from linearization of $\Xi(\Theta_{k+j\|k})$, $\Phi_{:\|k} \in \mathbb{R}^{4\times4}$
$\Psi(t)$	combined vector of Cartesian positions and velocities, $\Psi(t) = (x(t),\ \dot{x}(t))^T \in \mathbb{R}^4$
$\Theta(t)$	combined vector $\Theta(t) = (\theta_1(t), \theta_2(t),\ \dot{\theta}_1(t), \dot{\theta}_2(t))^T$
$\theta(t)$	vector of robot configurations $\theta(t) = (\theta_1(t), \theta_2(t))^T$
Θ_k	time discretization of $\Theta(t)$
$\tilde{A}_{\mathfrak{t}}$	auxiliary matrix $\tilde{A}_{\mathfrak{t}} \in \mathbb{R}^{n_c\times n_c}$ at time $\mathfrak{t}$
$\tilde{P}$	upper bound of P

$\tilde{x}_k$	reduced state vector $\tilde{x}_k \in \mathbb{R}^{n_{stat}}$ at time k
$\tilde{x}^i_{f\|k}$	dimension reduced target state of the i-th base trajectory, $\tilde{x}^i_{f\|k} \in \mathbb{R}^{n_{stat}}$
A, B, r	matrices defining an affine dynamic system
$A_{\Psi,:\|k}$, $B_{\Psi,:\|k}$, $r_{\Psi,:\|k}$	system matrices of linearized dynamics in Cartesian space
$A_{\Theta,:\|k}$, $B_{\Theta,:\|k}$, $r_{\Theta,:\|k}$	system matrices of linearized dynamics in configuration space
C_s	permutation matrix, $C_s \in \mathcal{C}$
$C_{AB\mathfrak{r}}$	controllability matrix of $\mathfrak{r}$-step controllable system
C_{AB}	controllability matrix
d_w	w-th element of $\mathcal{D}$
$D_{u,k}$	matrix of input differences between the base trajectories at time k, $D_{u,k} \in \mathbb{R}^{n_u \times n_c}$
$D_{x,k}$	matrix of state differences between the base trajectories at time k, $D_{x,k} \in \mathbb{R}^{n_x \times n_c}$
I	identity matrix
K_k	control matrix $K_k \in \mathbb{R}^{n_u \times n_c}$
$K_{k,s}$	s-th row of control matrix K_k
L_k	auxiliary matrix for the controller synthesis, $L_k \in \mathbb{R}^{n_u \times n_c}$
$L_{k,s}$	s-th row of matrix L_k
$n^{\perp}_{k+1}$	orthogonal basis $n^{\perp}_{k+1} \in \mathbb{R}^{n_x \times n_c}$ with respect to $D_{x,k+1}$
P	Lyapunov matrix, $P \in \mathbb{S}^{n_c}_{>0}$
p_l	*passing point* of $\mathcal{P}_{pas}$, and vertices of $\mathcal{P}_{xv,k}$
Q	weighting matrix for the states x_k: $Q = Q^T \geq 0$, $Q \in \mathbb{R}^{n_x \times n_x}$
Q_B, Q_{Bend}	weighting matrix for the determination of base trajectories: $\{Q_B, Q_{Bend}\} \in \mathbb{S}^{n_x}_{>0}$
Q_C	weighting matrix for the homotopic states $\boldsymbol{\lambda}_k$ in the controller synthesis, $Q_C = Q_C^T \geq 0$, $Q_C \in \mathbb{R}^{n_c \times n_c}$
Q_{end}	weighting matrix for end states

R	weighting matrix for the input u_k, $R = R^T \geq 0$, $R \in \mathbb{R}^{n_u \times n_u}$
r	affine linearization term
R_B	weighting matrix for the determination of base trajectories: $R_B \in \mathbb{S}^{n_x}_{>0}$
R_C	weighting matrix for the homotopic inputs $\boldsymbol{\lambda}_k$ in the controller synthesis, $R_C = R_C^T \geq 0$, $R_C \in \mathbb{R}^{n_u \times n_u}$
$Rot_{i-1}(x_3, \theta_i(t))$	rotation along x_3-axis of coordinate system $i-1$, with angle $\theta_i(t)$
s_k	coordinate-transformed state vector at time k, $s_k \in \mathbb{R}^{n_x}$
T_{con}	substitution $T_{con} := \tau\bar{\boldsymbol{\lambda}}(p_l, C_s)$
$Trans_i(x_1, l_i)$	translation along x_1-axis of coordinate system i, with length l_i
u_k	input vector at time k
$u_k(\boldsymbol{\lambda}_k)$	homotopic input at time k
u_k^i	input vector at time k of i-th *base trajectory*
$u_{add,k}$	additional input value at time k
$u_{new,k}(\boldsymbol{\lambda}_k)$	combined input of $u_k(\boldsymbol{\lambda}_k)$ and $u_{add,k}$ at time k
x_k	state vector at time k, $x_k \in \mathbb{R}^{n_x}$
$x_k(\boldsymbol{\lambda}_k)$	homotopic state at time k
x_k^i	state vector at time k of i-th *base trajectory*
$x_L,\ u_L$	linearization points
$x_s,\ x_f,\ u_s,\ u_f$	values of states and input
Y	inverse of the Lyapunov matrix, $Y = P^{-1}$
$z_{\mathfrak{t}}$	state vector at time $\mathfrak{t}$
$z_{\mathfrak{t}}(\boldsymbol{\mu}_{\mathfrak{t}})$	homotopic state at time $\mathfrak{t}$
$x_{f\|k}^i$	target state of i-th base trajectory
${}^{i-1}T_i$	coordinate transformation from the coordinate system i, to the coordinate system $i-1$

References

[1] P. Adolphs and D. Nafziger, "A method for fast computation of collision-free robot movements in configuration-space," in *IEEE Int. Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, 1990, pp. 5–12 vol.1.

[2] E. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*, ser. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 2012.

[3] B. Anderson and J. Moore, *Optimal Control: Linear Quadratic Methods*, ser. Dover Books on Engineering. Dover Publications, 2007.

[4] F. Aurenhammer, "Voronoi diagrams- a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991.

[5] L. Bakule, "Decentralized control: An overview," *Annual Reviews in Control*, vol. 32, no. 1, pp. 87 – 98, 2008.

[6] S. Barnett, *Matrices: Methods and Applications*, ser. Oxford Applied Mathematics and computing science. Clarendon Press, 1990.

[7] E. Behar and J. M. Lien, "Mapping the configuration space of polygons using reduced convolution," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 1242–1248.

[8] G. Betti, M. Farina, and R. Scattolini, *Distributed MPC: A Noncooperative Approach Based on Robustness Concepts*. Springer Netherlands, 2014, pp. 421–435.

[9] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proc. of the AAAI Conf. on Artificial Intelligence*, 2010, pp. 1230–1237.

[10] L. Blackmore and B. Williams, "Optimal manipulator path planning with obstacles using disjunctive programming," in *American Control Conference*, 2006, pp. 3200–3202.

[11] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1999, pp. 1018–1023.

[12] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994.

[13] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[14] M. S. Branicky and W. S. Newman, "Rapid computation of configuration space obstacles," in *Proc., IEEE Int. Conf. on Robotics and Automation*, 1990, pp. 304–310.

[15] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 15, no. 2, pp. 224–233, 1985.

[16] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems*, vol. 22, no. 1, pp. 44–52, 2002.

[17] J. F. P. C. M. Caron, R. J.and McDonald, "A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary," *J. of Optimization Theory and Applications*, vol. 62, no. 2, pp. 225–237, 1989.

[18] Y.-b. Chen, G.-c. Luo, Y.-s. Mei, J.-q. Yu, and X.-l. Su, "Uav path planning using artificial potential field method updated by optimal control theory," *Int. J. Syst. Sci.*, vol. 47, no. 6, pp. 1407–1420, 2016.

[19] H. T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *2015 IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 2347–2354.

[20] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005.

[21] P. D. Christofides, R. Scattolini, D. M. de la Peña, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Computers & Chemical Engineering*, vol. 51, pp. 21 – 41, 2013.

[22] R. Deits and R. Tedrake, "Efficient mixed-integer planning for UAVs in cluttered environments," in *IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 42–49.

[23] M. Diehl, H. Bock, J. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *J. of Process Control*, vol. 12, no. 4, pp. 577 – 585, 2002.

[24] H. Ding, G. Reissig, and O. Stursberg, "Increasing efficiency of optimization-based path planning for robotic manipulators," in *IEEE Conf. on Decision and Control*, 2011, pp. 1399–1404.

[25] G. Duan and H. Yu, *LMIs in Control Systems: Analysis, Design and Applications*. CRC Press, 2013.

[26] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp. 549 – 558, 2006.

[27] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[28] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 1243–1248.

[29] D. Ferguson and A. Stentz, "Anytime RRTs," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 5369–5375.

[30] A. Ferramosca, D. Limon, I. Alvarado, and E. Camacho, "Cooperative distributed MPC for tracking," *Automatica*, vol. 49, no. 4, pp. 906 – 914, 2013.

[31] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Tr. on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[32] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path planning for mobile robot navigation using voronoi diagram and fast marching," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 2376–2381.

[33] P. L. A. V. R. Gasparetto, Alessandroand Boscariol, *Path Planning and Trajectory Planning Algorithms: A General Overview*. Springer Int. Publishing, 2015, pp. 3–27.

[34] R. Gondhalekar and J. Imura, "Least-restrictive move-blocking model predictive control," *Automatica*, vol. 46, no. 7, pp. 1234 – 1240, 2010.

[35] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[36] A. Grancharova and T. A. Johansen, *Distributed MPC of Interconnected Nonlinear Systems by Dynamic Dual Decomposition*. Springer Netherlands, 2014, pp. 293–308.

[37] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[38] A. Hatcher, *Algebraic Topology*. Cambridge University Press, 2002.

[39] E. Hernandez, M. Carreras, and P. Ridao, "A comparison of homotopic path planning algorithms for robotic applications," *Robotics and Autonomous Systems*, vol. 64, pp. 44 – 58, 2015.

[40] E. Hernández, M. Carreras, P. Ridao, J. Antich, and A. Ortiz, "A search-based path planning algorithm with topological constraints. application to an AUV*," *IFAC World Congress*, vol. 44, no. 1, pp. 13 654 – 13 659, 2011.

[41] K. E. Hoff, III, J. Keyser, M. Lin, D. Manocha, and T. Culver, "Fast computation of generalized voronoi diagrams using graphics hardware," in *Conf. on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 277–286.

[42] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *IEEE Int. Conf. on Robotics and Automation ICRA*, 2009, pp. 2587–2592.

[43] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1990.

[44] S. Hrabar, "3d path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 807–814.

[45] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE Int. Conf. on Robotics and Automation*, vol. 3, 2003, pp. 4420–4426.

[46] H.-P. Huang and S.-Y. Chung, "Dynamic visibility graph for path planning," in *Int. Conf. on Intelligent Robots and Systems*, vol. 3, 2004, pp. 2813–2818.

[47] Y. K. Hwang, "Boundary equations of configuration obstacles for manipulators," in *IEEE Int. Conf. on Robotics and Automation*, May 1990, pp. 298–303.

[48] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Inf. Processing Systems. MIT Press*, 2003, pp. 1523–1530.

[49] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A condensed and sparse QP formulation for predictive control," in *IEEE Conf. on Decision and Control and European Control Conf.*, 2011, pp. 5217–5222.

[50] M. Jones, *Artificial Intelligence: A Systems Approach*. Jones & Bartlett Learning, 2015.

[51] K. Kaltsoukalas, S. Makris, and G. Chryssolouris, "On generating the motion of industrial robot manipulators," *Robotics and Computer-Integrated Manufacturing*, vol. 32, pp. 65 – 71, 2015.

[52] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1478–1483.

[53] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[54] L. E. Kavraki, "Computation of configuration-space obstacles using the fast fourier transform," *IEEE Trans. on Robotics and Automation*, vol. 11, no. 3, pp. 408–413, 1995.

[55] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[56] T. Keviczky and J. K. H.

[57] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[58] C. Kirches, L. Wirsching, H. Bock, and J. Schlöder, "Efficient direct multiple shooting for nonlinear model predictive control on long horizons," *J. of Process Control*, vol. 22, no. 3, pp. 540 – 550, 2012.

[59] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Trans. on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[60] D. Kontny and O. Stursberg, "Fast control using homotopy properties for obstacle-avoidance of systems with input constraints," in *IEEE Conf. on Computer Aided Control System Design*, Sept 2016, pp. 654–660.

[61] D. Kontny and O. Stursberg, "Fast optimizing control for non-convex state constraints using homotopy properties," in *IEEE Conf. on Decision and Control*, 2016, pp. 4894–4900.

[62] D. Kontny and O. Stursberg, "Online adaption of motion paths to time-varying constraints using homotopies," *IFAC World Congress*, vol. 50, no. 1, pp. 3331 – 3337, 2017.

[63] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.

[64] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 3713–3719.

[65] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using RRT," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, pp. 1681–1686.

[66] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[67] S. M. LaValle, J. H. Yakey, and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," in *IEEE Int. Conf. on Robotics and Automation*, vol. 3, 1999, pp. 1671–1676.

[68] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[69] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[70] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *Int. J. of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[71] S. Liao, "Notes on the homotopy analysis method: Some definitions and theorems," *Comm. in Nonlinear Science and Numerical Simulation*, vol. 14, no. 4, pp. 983 – 997, 2009.

[72] S. Liao, "On the homotopy analysis method for nonlinear problems," *Applied Mathematics and Computation*, vol. 147, no. 2, pp. 499 – 513, 2004.

[73] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613 – 1643, 2008.

[74] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *In advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[75] J. Liu, X. Chen, D. M. de la Peña, and P. D. Christofides, "Sequential and iterative architectures for distributed model predictive control of nonlinear process systems. part i: Theory," in *American Control Conf.*, 2010, pp. 3148–3155.

[76] J. Liu, D. Muñoz de la Peña, and P. D. Christofides, "Distributed model predictive control of nonlinear process systems," *AIChE J.*, vol. 55, no. 5, pp. 1171–1184, 2009.

[77] P. Liu and U. Ozguner, "Non-iterative distributed model predictive control for flexible vehicle platooning of connected vehicles," in *2017 American Control Conf.*, 2017, pp. 4977–4982.

[78] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. on Computers*, vol. 32, no. 2, pp. 108–120, 1983.

[79] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[80] F. M. and S. R., "Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems," *Automatica*, vol. 48, no. 6, pp. 1088 – 1096, 2012.

[81] A. A. Maciejewski and J. J. Fox, "Path planning and the topology of configuration space," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 4, pp. 444–456, 1993.

[82] F. Matoui, B. Boussaid, B. Metoui, G. Frej, and M. Abdelkrim, "Path planning of a group of robots with potential field approach: Decentralized architecture," *IFAC World Congress*, vol. 50, no. 1, pp. 11 473–11 478, 2017.

[83] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789 – 814, 2000.

[84] M. Müller and F. Allgöwer, "Distributed MPC for consensus and synchronization," *Intelligent Systems, Control and Automation: Science and Engineering*, vol. 69, pp. 89–100, 2014.

[85] M. A. Müller, M. Reble, and F. Allgöwer, "A general distributed mpc framework for cooperative control," *IFAC World Congress*, vol. 44, no. 1, pp. 7987–7992, 2011.

[86] A. Mosek, "The mosek optimization software," *Online at http://www. mosek. com*, vol. 54, pp. 2–1, 2010.

[87] R. R. Negenborn and J. M. Maestre, "Distributed model predictive control: An overview and roadmap of future research opportunities," *IEEE Control Systems*, vol. 34, no. 4, pp. 87–97, 2014.

[88] G. Neumann and W. Maass, "Learning complex motions by sequencing simpler motion templates," in *Conf. on Machine Learning*, 2009.

[89] W. Newman and M. Branicky, "Real-time configuration space transforms for obstacle avoidance," *J. of Robotics Research*, vol. 6, pp. 650–667, 1991.

[90] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[91] F. Nori and R. Frezza, "Nonlinear control by a finite set of motion primitives," in *IFAC Symp. on Nonlinear Control Systems*, 2004, pp. 1–6.

[92] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[93] D. H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *IEEE Conf. on Humanoid Robots*, 2008, pp. 91–98.

[94] M. G. Park, J. H. Jeon, and M. C. Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *IEEE Int. Symposium on Industrial Electronics Proceedings*, vol. 3, 2001, pp. 1530–1535.

[95] Y. Pochet and L. Wolsey, *Production Planning by Mixed Integer Programming*, ser. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[96] K. Reif, K. Weinzierl, A. Zell, and R. Unbehauen, "A homotopy approach for nonlinear control synthesis," *IEEE Tr. on Automatic Control*, vol. 43, no. 9, pp. 1311–1318, 1998.

[97] A. Richards, "Fast model predictive control with soft constraints," in *European Control Conf.*, 2013, pp. 1–6.

[98] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *IEEE Conf. on Decision and Control*, vol. 4, 2004, pp. 4286–4291.

[99] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conf.*, vol. 3, 2002, pp. 1936–1941.

[100] T. Rybus and K. Seweryn, "Application of rapidly-exploring random trees (RRT) algorithm for trajectory planning of free-floating space manipulator," in *Int. Workshop on Robot Motion and Control*, 2015, pp. 91–96.

[101] F. Samaniego, J. Sanchis, S. García-Nieto, and R. Simarro, "UAV motion planning and obstacle avoidance based on adaptive 3d cell decomposition: Continuous space vs discrete space," in *IEEE Second Ecuador Technical Chapters Meeting*, 2017, pp. 1–6.

[102] A. Schrijver, *Theory of Linear and Integer Programming*, ser. Wiley Series in Discrete Mathematics & Optimization. John Wiley & Sons, 1998.

[103] G. Seber and A. Lee, *Linear Regression Analysis*, ser. Wiley Series in Probability and Statistics. Wiley, 2012.

[104] J. Sfeir, M. Saad, and H. Saliah-Hassane, "An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment," in *IEEE Int. Symposium on Robotic and Sensors Environments*, 2011, pp. 208–213.

[105] W. Shu and Z. Zheng, "Computing configuration space obstacles using polynomial transforms," in *IEEE Int. Conf. on Robotics and Automation*, vol. 4, 2004, pp. 3920–3924.

[106] M. Spong, *Robot Modeling and Control*. Hoboken, NJ: John Wiley & Sons, 2006.

[107] M. Steinegger, B. Passenberg, M. Leibold, and M. Buss, "Trajectory planning for manipulators based on the optimal concatenation of LQ control primitives," in *IEEE Conf. on Decision and Control*, 2011, pp. 2837–2842.

[108] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *IEEE Int. Conf. on Robotics and Automation*, vol. 4, 1994, pp. 3310–3317.

[109] B. T. Stewart, A. N. Venkat, J. B. Rawlings, S. J. Wright, and G. Pannocchia, "Cooperative distributed model predictive control," *Systems & Control Letters*, vol. 59, no. 8, pp. 460 – 469, 2010.

[110] B. T. Stewart, S. J. Wright, and J. B. Rawlings, "Cooperative distributed model predictive control for nonlinear systems," *J. of Process Control*, vol. 21, no. 5, pp. 698–704, 2011.

[111] S. Stoneman and R. Lampariello, "Embedding nonlinear optimization in RRT* for optimal kinodynamic planning," in *IEEE Conf. on Decision and Control*, 2014, pp. 3737–3744.

[112] K. Sugihara, "Approximation of generalized voronoi diagrams by ordinary voronoi diagrams," *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 6, pp. 522 – 531, 1993.

[113] X. Sun, W. Yeoh, and S. Koenig, "Moving target D* lite," in *Int. Conf. on Autonomous Agents and Multiagent Systems*, vol. 1, 2010, pp. 67–74.

[114] P. Tøndel, T. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, no. 3, pp. 489 – 497, 2003.

[115] H. Vazquez-Leal, A. Marin-Hernandez, Y. Khan, A. Yıldırım, U. Filobello-Nino, R. Castaneda-Sheissa, and V. Jimenez-Fernandez, "Exploring collision-free path planning by using homotopy continuation methods," *Applied Mathematics and Computation*, vol. 219, no. 14, pp. 7514 – 7532, 2013.

[116] A. N. Venkat, I. A. Hiskens, J. B. Rawlings, and S. J. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Trans. on Control Systems Technology*, vol. 16, no. 6, pp. 1192–1206, 2008.

[117] J. Verschelde, "PHCpack: A general-purpose solver for polynomial systems by homotopy continuation," *ACM Trans. Math. Softw.*, vol. 25, no. 2, pp. 251–276, 1999.

[118] H. Wang, Y. Yu, and Q. Yuan, "Application of Dijkstra algorithm in robot path-planning," in *Int. Conf. on Mechanic Automation and Control Engineering*, 2011, pp. 1067–1069.

[119] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Tr. on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[120] J. Ward and J. Katupitiya, "Free space mapping and motion planning in configuration space for mobile manipulators," in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 4981–4986.

[121] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 5054–5061.

[122] H. P. Williams, *Logic and Integer Programming*. Springer Publishing Company, Incorporated, 2009.

[123] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1999, pp. 1024–1031.

[124] K. D. Wise and A. Bowyer, "A survey of global configuration-space mapping techniques for a single robot in a static environment," *The Int. J. of Robotics Research*, vol. 19, no. 8, pp. 762–779, 2000.

[125] L. Wolsey and G. Nemhauser, *Integer and Combinatorial Optimization*, ser. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2014.

[126] X. Wu, Q. Li, and K. H. Heng, "A new algorithm for construction of discretized configuration space obstacle and collision detection of manipulators," in *Int. Conf. on Advanced Robotics*, 2005, pp. 90–95.

[127] K. Yang and S. Sukkarieh, "Real-time continuous curvature path planning of UAVs in cluttered environments," in *Int. Symposium on Mechatronics and its Applications*, 2008, pp. 1–6.

[128] M. S. Zahedi and H. S. Nik, "On homotopy analysis method applied to linear optimal control problems," *Applied Mathematical Modelling*, vol. 37, no. 23, pp. 9617–9629, 2013.

[129] M. Zeilinger, C. Jones, and M. Morari, "Real-time suboptimal model predictive control using a combination of explicit MPC and online optimization," *IEEE Tr. on Automatic Control*, vol. 56, no. 7, pp. 1524–1534, 2011.

[130] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Int. Conf. on Intelligent Systems Design and Applications*, vol. 2, 2006, pp. 622–627.

[131] M. Zucker, J. Kuffner, and M. Branicky, "Multipartite RRTs for rapid replanning in dynamic environments," in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 1603–1609.

ISBN 978-3-7376-0870-1