2016 IEEE Conference on Computer Aided Control System Design (CACSD)
Part of 2016 IEEE Multi-Conference on Systems and Control
September 19-22, 2016. Buenos Aires, Argentina

Fast Control using Homotopy Properties for Obstacle-Avoidance of Systems with Input Constraints

Damian Kontny[1] and Olaf Stursberg[1]

Abstract— **For the optimal point-to-point control of a discrete-time linear system in a time varying non-convex environment, this paper investigates how such problems can be handled online. An illustrative example is that of steering a robotic end-effector from an initial to a final state, while executing the motion free of collision in the presence of a human worker. The challenge is to obtain a solution with low effort, to be online applicable, while maintaining a close to optimal solution. The proposed method consists of first computing a range of trajectories, homotopic to the optimal unconstrained solution offline. Then, upon detection of an obstacle which may block the currently executed trajectory, an offline synthesized controller steers the system quickly to a desired homotopic trajectory. The resulting trajectory is free of collision and is determined by a fast iterative procedure.**

I. INTRODUCTION

The scope of this paper is to control linear time-discrete systems from an initial state to a target state while avoiding collisions with time-varying obstacles and additionally satisfying input constraints. A motivating example is that of a robotic manipulator which has to accomplish tasks in two different positions and has to move between these points. Assume that, a human worker operates in the same region and may block the manipulator in its motion between the two positions. For human robot cooperation in industrial processes, this scenario is of great interest in order to combine the advantages of human flexibility and robotic precision. If a human moves fast, the robotic manipulator depends on techniques that compute circumventing trajectories quickly, while considering its actuator saturations, and the performance of reaching.

The objective of this paper is to explore means to decompose the problem into an offline and online part, while the latter has acceptably small computation times.

Linear *model predictive control* (MPC) naturally handles constraints by linear and or quadratic optimization techniques [12]. These methods are commonly used in process industry and other areas. In recent years, considerable effort has been spent to speed up MPC by using soft constraints [18], [16], *move blocking* strategies [6], which fix the inputs over a certain period of time, or direct multiple shooting methods [10], which exploit the structure of Hessian and Jacobian matrices. In addition, methods such as *explicit* MPC were developed, which compute control laws offline by partitioning the state space [19]. These methods are limited to low system dimensions and/or time horizons. While for problems with convex search space, significant progress has been made, the fast online computation for the non-convex case remains challenging. In the case of the robot example described above, non-convexity originates from the presence of an obstacle. To solve this problem, nonlinear optimization techniques such as SQP [14] or mixed-integer programming (MIP) [2], [4] can be applied. The well known Big-M formulation [17] can be applied to separate the free space from the obstacle with binary variables and then solve the problem with branch-and-bound or branch-and-cut techniques. When the considered time horizon is additionally chosen large enough to obtain a feasible solution, the method becomes impractical for many applications with real-time requirements.

With respect to planning in non-convex state-spaces, a variety of publications originates from the field of human robot interaction. Important candidates of this field are *potential field methods* [9], *cell-decomposition methods* [13], but also sampling based approaches like *rapidly exploring random trees* (RRT) [11] and RRT* [8], [5]. However, these methods have in common that they do not consider the system dynamics.

In contrast to the existing work, the approach presented in this paper achieves low computation times for the named problem by using homotopy properties. A finite set of different trajectories is computed offline, which spans a region for circumventing the human worker in operation. Then, a homotopic trajectory inside the spanned region is determined online such that it passes the obstacle optimally without collision. Thus, the robotic manipulator drives from its actual position to the homotopic trajectory by means of offline computed time-varying state feedback controllers. Since dynamic systems, like the one described here, are usually subject to actuator limitations, the offline computed state feedback controllers are synthesized with respect to this input saturations. Formulations for consideration of input constraints in controller synthesis can be found in [3] and [7]. In literature, the use of homotopy properties in optimization is limited to unconstrained cases and solution by model approximation, i.e. to cases which are quite distinct from the setting considered here, see e.g. [15].

The following parts of the paper first introduce the considered type of homotopic functions (Sec. II), then formulate the problem (Sec. III), describe the offline computations (Sec. IV), explain the online collision avoidance by controlling the system to homotopic trajectories (Sec. V), and show numerical results (Sec. VI).

[1]Control and System Theory, Dept. of Electrical Engineering and Computer Science, University of Kassel (Germany). Email: {dkontny, stursberg}@uni-kassel.de

II. HOMOTOPIC FUNCTIONS

The dynamics considered is that of discrete-time linear systems:

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

with states $x_k \in \mathbb{R}^{n_x}$, inputs $u_k \in \mathbb{R}^{n_u}$, $A \in \mathscr{R}^{n_x \times n_x}$, $B \in \mathscr{R}^{n_x \times n_u}$, and the time index $k \in \mathbb{N}_0$. Given a finite time domain $T = \{0, 1, \ldots, N\}$, $N \in \mathbb{N}$, let the state and input trajectory be denoted by $\hat{x} = (x_0, ..., x_N)$ and $\hat{u} = (u_0, ..., u_{N-1})$.

Consider $n_c + 1$ pairs of trajectories $(\hat{u}^i, \hat{x}^i)$, indexed by i, with $i \in \mathscr{M} := \{0, 1, ..., n_c\}$. Let $\mathscr{X} := \{\hat{x}^0, \ldots, \hat{x}^{n_c}\}$ denote the set of state trajectories, assuming that for any pair $\hat{x}^i$, $\hat{x}^j$, $i \neq j$ in $\mathscr{M}$ the trajectories differ. The initial state x_0^i at time $k = 0$ is chosen to be equal for all trajectories $i \in \mathscr{M}$ and the same applies to the final state x_N^i. Let $\hat{x}^0$ denote the trajectory in $\mathscr{X}$ which is optimal with respect to a given performance criterion. The other trajectories in $\mathscr{X}$ are called *base trajectories* and are chosen to span a region (around $\hat{x}^0$) in which circumvention of an obstacle is possible, if $\hat{x}^0$ intersects with the latter. Now interpret the trajectories $\hat{x}^i$ as the images of functions: $\hat{x}^i = F^i(\hat{u}^i)$. The following definition defines *homotopic* trajectories in between of the trajectories $\hat{x}^i$.

DEFINITION II.1 For a set of $n_c + 1$ continuous functions $F^i : \mathbb{R}^{n_u \times T} \to \mathbb{R}^{n_x \times T}$, $i \in \mathscr{M}$, a *vectorized homotopy* is defined by $H : \left(\mathbb{R}^{n_u \times T}\right)^{n_c} \times [0,1]^{n_c} \to \mathbb{R}^{n_x \times T}$. The second argument is a vector of homotopy parameters $\boldsymbol{\lambda} = (\lambda^1, ..., \lambda^{n_c})^T$ with $\lambda^i \in [0,1]$. The *linear vectorized homotopy* function is given with $F = (F^1(\hat{u}^1) - F^0(\hat{u}^0), ..., F^{n_c}(\hat{u}^{n_c}) - F^0(\hat{u}^0))^T$ and with $\lambda^0 := 1 - \sum_{i=1}^{n_c} \lambda^i$ according to:

$$\begin{aligned} H(\hat{u}^0, \ldots, \hat{u}_c^n, \boldsymbol{\lambda}) &= \sum_{i=0}^{n_c} \lambda^i \cdot F^i(\hat{u}^i) \\ &= F^0(\hat{u}^0) + \sum_{i=1}^{n_c} (F^i(\hat{u}^i) - F^0(\hat{u}^0)) \cdot \lambda^i = F^0(\hat{u}^0) + F \cdot \boldsymbol{\lambda} \end{aligned} \tag{2}$$

△

Equation (2) shows that the homotopy can be understood as a linear interpolation between the optimal and base trajectories using the weights λ^i. A trajectory $\hat{x}^i \in \mathscr{X}$ is obtained for $\lambda^i = 1$, while the other components of the vector $\boldsymbol{\lambda}$ are set to zero (i.e. the trajectory lies in the boundary of the homotopic space). While (2) formulates the homotopy between complete trajectories, the homotopy of a single point of time k can be specified by introducing matrices $D_{x_k} = [x_k^1 - x_k^0, ..., x_k^{n_c} - x_k^0] \in \mathbb{R}^{n_x \times n_c}$, $k \in T$, and likewise $D_{u_k} = [u_k^1 - u_k^0, ..., u_k^{n_c} - u_k^0] \in \mathbb{R}^{n_u \times n_c}$. The state x_k^i denotes the state of the i-th trajectory at time k. The same applies to u_k^i for the input. Homotopic states and inputs at time k can then be written as:

$$x_k(\boldsymbol{\lambda}_k) := x_k^0 + D_{x_k} \boldsymbol{\lambda}_k, \quad u_k(\boldsymbol{\lambda}_k) := u_k^0 + D_{u_k} \boldsymbol{\lambda}_k, \tag{3}$$

with $\boldsymbol{\lambda}_k := (\lambda_k^1, ..., \lambda_k^{n_c})^T \in \mathbb{R}^{n_c}$ denoting the vector of homotopy parameters at time k. A constant homotopy vector over time is denoted by $\bar{\boldsymbol{\lambda}}$, and the corresponding trajectories are then denoted by $\hat{x}(\bar{\boldsymbol{\lambda}})$ and $\hat{u}(\bar{\boldsymbol{\lambda}})$.

III. PROBLEM DEFINITION

Consider the case, in which system (1) follows the optimal trajectory $\hat{x}^0$ from x_0 towards x_N until an obstacle $\mathscr{P}_x := \{x | \; Cx \leq d\} \subseteq \mathbb{R}^{n_x}$, with $C \in \mathbb{R}^{c \times n_x}$ and $d \in \mathbb{R}^c$, is detected at time $k^* \in \{1, \ldots, N-1\}$, such that $\hat{x}^0$ can not be followed further. Within this paper, $\mathscr{P}_x$ is assumed to be static for all $k \in \{k^*, \ldots, N\}$, i.e. after detection. To allow for a feasible solution, $x_{k^*}^0 \notin \mathscr{P}_x$ and $x_N^0 \notin \mathscr{P}_x$ is required. The goal is to compute an optimized trajectory $\hat{x}^* = (x_{k^*}^*, ..., x_N^*) \in \mathbb{R}^{n_x \times (N+1-k^*)}$ and $\hat{u}^* = (u_{k^*}^*, ..., u_{N-1}^*) \in \mathbb{R}^{n_u \times (N-k^*)}$, while avoiding any collision with the polytope, ensuring $x_N = x_f$, and additionally satisfying the input constraints $u_{k^*+j} \in \mathscr{U} \subseteq \mathbb{R}^{n_u}$ for all $j \in \mathscr{J} := \{0, \ldots, N-1-k^*\}$. Based on the selection of the optimal trajectories by a quadratic performance criterion, the optimization problem can be summarized to:

$$\begin{aligned} \min_{x_{k^*+j},\, u_{k^*+j}} \quad & \sum_{j=0}^{N-1-k^*} (x_{k^*+j} - x_f)^T Q (x_{k^*+j} - x_f) \\ & + (u_{k^*+j} - u_f)^T R (u_{k^*+j} - u_f) && (4) \\ \text{s.t.} \quad & (1),\; u_{k^*+j} \in \mathscr{U},\; x_{k^*+j} \notin \mathscr{P}_x,\; \forall j \in \mathscr{J} && (5) \\ & x_{k^*} = x_{k^*}^0,\; x_N = x_f, && (6) \end{aligned}$$

with suitable positive-definite weighting matrices $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$. The values x_{k^*+j} and u_{k^*+j} denote the states and inputs at time $k^* + j$, with $j \in \mathscr{J}$ denoting the future time steps counting from k^*.

The problem stated here is a non-convex problem. A possible approach is, as remarked in Sec. I and illustrated in Sec. VI for an example, 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 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 by reasons of 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 with MIQP. The idea is to (i) select a desired homotopic trajectory $\hat{x}(\bar{\boldsymbol{\lambda}})$ and (ii) to employ offline computed controllers that realize the transition from the current to the desired trajectory. For enabling that a solution to the problem can be found in step (i), a necessary condition is made in the following assumption.

ASSUMPTION III.1 Let the set of trajectories $\mathscr{X}$ contain at least one trajectory $\hat{x}^i \in \mathscr{X}$ such that for $x_{k^*+j}^i \in \hat{x}^i$ it holds that, $x_{k^*+j}^i \notin \mathscr{P}_x$, $\forall j \in \mathscr{J}$.

This assumption is justified by the fact that a circumvention of $\mathscr{P}_x$ is not possible, if the whole admissible space (constructed by $\mathscr{X}$) is blocked. Nevertheless, the assumption is not sufficient for finding a feasible solution, since it must be ensured also in step (ii) that the transition to $\bar{\boldsymbol{\lambda}}$ is achieved without intersecting $\mathscr{P}_x$.

IV. OFFLINE CONTROLLER SYNTHESIS

This sections deals with the offline part of the solution procedure, i.e., the computation of controllers to transfer the system to a new homotopy vector $\bar{\boldsymbol{\lambda}}$. To accomplish this goal, the system (1) has to be transformed into the homotopy space, leading to a linear time-varying (LTV) system. Based on this, semi-definite programming is used to synthesize state feedback controllers for realizing transitions within the $\boldsymbol{\lambda}$-space.

A. Transformation of the system dynamics into the homotopy space

The goal is to steer the system (1) from the state x_{k^*}, which corresponds to the current homotopy vector $\boldsymbol{\lambda}_{k^*}$, to a target trajectory determined by $\bar{\boldsymbol{\lambda}}$. With the offline computation, the corresponding input trajectories of the optimal and base trajectories are known from (3). Aiming to transition from $\boldsymbol{\lambda}_{k^*}$, to $\bar{\boldsymbol{\lambda}}$, additional inputs δu_k are required, leading to:

$$\tilde{u}_k(\boldsymbol{\lambda}_k) := u_k(\boldsymbol{\lambda}_k) + \delta u_k. \tag{7}$$

Because of the additional input term δu_k, the superposition may lead to signals $\tilde{u}_k(\boldsymbol{\lambda}_k)$ which are not in the set of the homotopic input trajectories according to (3). Therefore, the controller synthesis later (Sec. IV-B) addresses this issue.

The system dynamics can be described dependent on $\boldsymbol{\lambda}_k$, by inserting $x_k(\boldsymbol{\lambda}_k)$ according to (3) and $\tilde{u}_k(\boldsymbol{\lambda}_k)$ as in (7), into (1):

$$x_{k+1}(\boldsymbol{\lambda}_{k+1}) := Ax_k(\boldsymbol{\lambda}_k) + B\tilde{u}_k(\boldsymbol{\lambda}_k), \tag{8}$$

With (3), (7), and (8), the system can be further transformed for $k \in \{0, ..., N-1\}$ to:

$$D_{x_{k+1}}\boldsymbol{\lambda}_{k+1} + x^0_{k+1} = A(x^0_k + D_{x_k}\boldsymbol{\lambda}_k) + B(u^0_k + D_{u_k}\boldsymbol{\lambda}_k + \delta u_k) \tag{9}$$

$$\Leftrightarrow \quad D_{x_{k+1}}\boldsymbol{\lambda}_{k+1} = (AD_{x_k} + BD_{u_k})\boldsymbol{\lambda}_k + B\delta u_k \tag{10}$$

$$\Leftrightarrow \quad D_{x_{k+1}}\boldsymbol{\lambda}_{k+1} = D_{x_{k+1}}\boldsymbol{\lambda}_k + B\delta u_k. \tag{11}$$

The last equation is an LTV-system with the homotopy vector $\boldsymbol{\lambda}_{k+1}$ denoting the new homotopy vector, with input δu_k, and the time-dependent and bounded matrix $D_{x_{k+1}}$.

ASSUMPTION IV.1 The number of *base trajectories* equals the system dimension: $n_c = n_x$.

This assumption[1] is introduced in order to obtain a full-dimensional space for the circumvention of $\mathscr{P}_x$. It implies that $D_{x_k} \in \mathbb{R}^{n_x \times n_x}$ is quadratic and its inverse $D^{-1}_{x_k}$ is non-singular (since the trajectories in $\mathscr{X}$ differ, see Sec. II). Thus, $\boldsymbol{\lambda}_{k+1}$ can be represented explicitly by:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + D^{-1}_{x_{k+1}}B\delta u_k, \tag{12}$$

and it describes the transition between trajectories, when the additional input δu_k acts. To ensure that any homotopic future state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$, in $k+1$, can be reached from $x_k(\boldsymbol{\lambda}_k)$, the system has to be fully controllable in one step. This requires $n_u = n_x$ and a full rank of B, leading to a controllability matrix $C_{AB} = [B, AB, ..., A^{n-1}B]$ with full rank.

[1]The case of $n_c > n_x$, or $n_c < n_x$ can be a suitable alternative to enlarge or reduce the homotopy space, but is not considered here for brevity.

B. Fast converging LTV-System Controller Synthesis with Input Constraints

To obtain a fast online control procedure, which additionally satisfies input constraints, an explicit control law with time varying matrices $K_k \in \mathbb{R}^{n_u \times n_x}$ is synthesized offline:

$$\delta u_k := -K_k(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}). \tag{13}$$

The control law enables the transition from a value $\boldsymbol{\lambda}_k$ to a desired value $\bar{\boldsymbol{\lambda}}$, or respectively, to an offline computed homotopic target trajectory $\hat{x}(\bar{\boldsymbol{\lambda}})$. The synthesis of the controller matrices K_k which need to be able to steer the system between any arbitrary combination $\boldsymbol{\lambda}_k$ and $\bar{\boldsymbol{\lambda}}$ is explained next. To make full use of the permissible input range, the controller is designed to perform the transition phase from the current trajectory to a desired homotopic one as fast as possible. The synthesis is based on Lyapunov stability conditions and input saturations, formulated as linear matrix inequalities (LMI's). $\bar{\boldsymbol{\lambda}}$ is assumed to be known here; its computation is covered later in Sec. V.

The *closed-loop delta-system* of (12), with $\delta\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_{k+1} - \bar{\boldsymbol{\lambda}}$ and control law (13) is defined by:

$$\delta\boldsymbol{\lambda}_{k+1} = (I - D^{-1}_{x_{k+1}}BK_k)\delta\boldsymbol{\lambda}_k. \tag{14}$$

Let a quadratic Lyapunov function, with matrix $P = P^T > 0 \in \mathbb{R}^{n_c \times n_c}$, be given by:

$$V_k = \delta\boldsymbol{\lambda}^T_k P \delta\boldsymbol{\lambda}_k. \tag{15}$$

The LTV-system (14) becomes stable with given decreasing rate parametrized by the matrix $Q_c^{-1} > 0 \in \mathbb{R}^{n_c \times n_c}$, if the following condition holds:

$$V_{k+1} - V_k \leq -\delta\boldsymbol{\lambda}^T_k Q_c^{-1} \delta\boldsymbol{\lambda}_k. \tag{16}$$

By defining the vector $\mathbf{q} = (1/q_1, ..., 1/q_{n_c})^T \in \mathbb{R}^{n_c}$, the matrix Q_c^{-1} is given as follows:

$$Q_c^{-1} := diag(\mathbf{q}). \tag{17}$$

Here, $diag(\mathbf{q})$ denotes the diagonal matrix obtained from the vector $\mathbf{q}$. The goal is, to determine the controllers such that the decrease of the homotopy parameter $\delta\boldsymbol{\lambda}$ converges fast to zero, which means that the transition phase between the actual and the desired trajectory becomes short. However, fast convergency of $\delta\boldsymbol{\lambda}$ forces high input signals. During the transition phase, it should be guaranteed that the resulting total input signal (7) is bounded. As being addressed in [3], the norm of each component $\tilde{u}_{k,s}(\boldsymbol{\lambda}_k)$, with index $s \in \{1, ..., n_u\}$ denoting the s-th component of the input vector $\tilde{u}_k(\boldsymbol{\lambda}_k)$, is bounded by an upper bound $\overline{u}_s \in \mathbb{R}_{>0}$. The input constraint is valid for all times k and given by:

$$\|\tilde{u}_{k,s}(\boldsymbol{\lambda}_k)\| \leq \|\overline{u}_s\|. \tag{18}$$

Since the norm of each component is bounded, the constraint bounds the input symmetrically in the positive and negative range. Consider now the ellipsoidal set:

$$\mathscr{E}(P) := \{\delta\boldsymbol{\lambda}_k \in \mathbb{R}^{n_c} | \ \delta\boldsymbol{\lambda}^T_k P \delta\boldsymbol{\lambda}_k \leq 1\}, \tag{19}$$

or equivalently:

$$\mathscr{E}(P) := \{P^{-\frac{1}{2}}z \mid \|z\| \leq 1\}. \tag{20}$$

DEFINITION IV.1 The ellipsoidal set $\mathscr{E}(P)$ denotes the region of asymptotic stability of the LTV closed-loop delta system (14), if (16) holds. △

Introducing a set $\mathscr{C}$ with elements $\boldsymbol{\gamma}_i \in \mathbb{R}^{n_c}$, $i \in \mathscr{M}$. The set $\mathscr{C}$ contains all possible corners of the homotopy vector $\boldsymbol{\lambda}_k$ denoted by the vectors $\boldsymbol{\gamma}_i$. Hence, the region of permissible $\boldsymbol{\lambda}_k$ is given by the convex hull of $\mathscr{C}$, thus $\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathscr{C}\}$. The same holds for the desired homotopy vector $\bar{\boldsymbol{\lambda}} \in \mathbf{Co}\{\mathscr{C}\}$. The feasible region of the difference vector $\delta\boldsymbol{\lambda}_k$ is given by the convex hull $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathscr{D}\}$, with $\mathscr{D} := \{d_w | d_w = \boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}, \boldsymbol{\lambda}_k \in \mathscr{C}, \bar{\boldsymbol{\lambda}} \in \mathscr{C}\}$, where $w \in \{1, ..., |\mathscr{D}|\}$ selects the w-th element d_w of the set $\mathscr{D}$. $|\mathscr{D}|$ denotes the cardinality of $\mathscr{D}$.

As argued in [3], an LTV system is Lyapunov-stable, if the Lyapunov stability condition (16) is satisfied for all $k \in \{0, ..., N-2\}$, with its time dependent, bounded matrices given by D_{x_k}. The input constrained controller synthesis task can be formulated as the following optimization problem for all $k \in \{0, ..., N-2\}$ and input components $s \in \{1, ..., n_u\}$:

$$\max_{P,\ K_k,\ Q_c, \delta\boldsymbol{\lambda}_k} \quad \text{trace}(Q_c^{-1}) + \text{trace}(P) \tag{21}$$

$$\text{s.t.} \quad V_{k+1} - V_k \leq -\delta\boldsymbol{\lambda}_k{}^T Q_c^{-1} \delta\boldsymbol{\lambda}_k \tag{22}$$

$$\|\tilde{u}_{k,s}(\boldsymbol{\lambda}_k)\| \leq \|\overline{u}_s\| \ \forall \boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathscr{C}\} \forall \bar{\boldsymbol{\lambda}} \in \mathbf{Co}\{\mathscr{C}\} \tag{23}$$

$$\delta\boldsymbol{\lambda}_k^T P \delta\boldsymbol{\lambda}_k \leq 1, \forall \delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathscr{D}\} \tag{24}$$

$$P > 0,\ Q_c > 0. \tag{25}$$

The objective of the optimization problem is to determine a fast converging controller, as enforced by (22) and the cost term trace(Q_c^{-1}). Meanwhile, the input constraints (23) have to be satisfied for all combinations of $\boldsymbol{\lambda}_k$, $\bar{\boldsymbol{\lambda}}$ and each component indexed by s of the input vector. Since the input signal depends To make full use of the available input range, the ellipsoidal region of stability (24) has to be shrinked up to the permissible region $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathscr{D}\}$, which is applied by the cost term trace(P). Then, the input signal becomes exhausted by demanding a high decrease in the Lyapunov function as mentioned before. The size and shape of an ellipsoid can generally be minimized by different criteria. Here, the size of $\mathscr{E}(P)$ is minimized by minimizing the sum of the semi-axes, what equals to maximize the trace of the shape matrix trace(P). How fast the controller can drive the system to a desired homotopy value is therefore limited by the input constraints.

Due to the non-convexity in (22), (23), and (24), caused by the multiplication of $\delta\boldsymbol{\lambda}_k$ with other variables, the problem is hard to solve in its original form. It can be shown, though, that the optimization problem can be formulated as a convex semi-definite program.

LEMMA IV.1 The non-convex constraint (22) can be transformed into:

$$(I - D_{x_{k+1}}^{-1} BK_k)^T P (I - D_{x_{k+1}}^{-1} BK_k) - P + Q_c^{-1} \leq 0, \tag{26}$$

with $P = Y^{-1}$ and $K_k = L_k Y^{-1}$, if $Y = Y^T > 0 \in \mathbb{R}^{n_c \times n_c}$ and $L_k \in \mathbb{R}^{n_u \times n_c}$ exist for all $k \in \{0, ..., N-2\}$. Then, the LMI:

$$\begin{bmatrix} Y & (Y - D_{x_{k+1}}^{-1} BL_k)^T & Y^T \\ Y - D_{x_{k+1}}^{-1} BL_k & Y & 0 \\ Y & 0 & Q_c \end{bmatrix} > 0, \tag{27}$$

holds with 0 denoting zero matrices of appropriate dimension.

Proof. By inserting the Lyapunov functions (15) for times k and $k+1$ into (22), and by substituting $\delta\boldsymbol{\lambda}_{k+1}$ according to the difference equation (14), the inequalities (26) are obtained. Now, using the substitution $Y = P^{-1}$ and multiplying (26) from the left and right with Y leads to:

$$\begin{aligned}(Y - D_{x_{k+1}}^{-1} BK_k Y)^T Y^{-1} (Y - D_{x_{k+1}}^{-1} BK_k Y) - Y \\ + Y^T Q_C^{-1} Y < 0.\end{aligned} \tag{28}$$

Applying the Schur complement to (28) yields:

$$\begin{bmatrix} Y & (Y - D_{x_{k+1}}^{-1} BK_k Y)^T & Y^T \\ Y - D_{x_{k+1}}^{-1} BK_k Y & Y & 0 \\ Y & 0 & Q_c \end{bmatrix} > 0, \tag{29}$$

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

LEMMA IV.2 The input constraint (23) can be conservatively approximated by the following LMI:

$$\begin{bmatrix} Y & L_{k,s}^T \\ L_{k,s} & (\|\overline{u}_s\| - \|u_{k,s}^0 + D_{u_{k,s}} \boldsymbol{\lambda}_k\|)^2, \end{bmatrix} \geq 0, \quad \forall \boldsymbol{\lambda}_k \in \mathscr{C},\ s \in \{1, ..., n_u\} \tag{30}$$

which is valid for every component s of the input vector and for all corners of the permissible region of $\boldsymbol{\lambda}_k$ given by the set $\mathscr{C}$. The index in $D_{u_{k,s}}$ and $L_{k,s}$ denotes the s-th row of the corresponding matrix.

Proof. By inserting (13) and (3) into (7), the input inequality (18) can be written as:

$$\|u_{k,s}^0 + D_{u_{k,s}} \boldsymbol{\lambda}_k - K_{k,s} \delta\boldsymbol{\lambda}_k\| \leq \|\overline{u}_s\|, \tag{31}$$

for each input component. Matrix $K_{k,s}$ denotes the s-th row of matrix K_k. With the triangle inequality $\|a+b\| \leq \|a\| + \|b\|$, (31) becomes:

$$\|u_{k,s}^0 + D_{u_{k,s}} \boldsymbol{\lambda}_k\| + \|K_{k,s} \delta\boldsymbol{\lambda}_k\| \leq \|\overline{u}_s\|. \tag{32}$$

Now, to satisfy these constraints for all $\delta\boldsymbol{\lambda}_k \in \mathbf{Co}\{\mathscr{D}\}$, $\delta\boldsymbol{\lambda}_k$ is replaced by the stabilizing ellipsoid (20), which is a superset of the permissible region of $\delta\boldsymbol{\lambda}_k$. Hence, $\mathscr{E}(P_k) \supseteq \mathbf{Co}\{\mathscr{D}\}$, and (32) becomes:

$$\|u_{k,s}^0 + D_{u_{k,i}} \boldsymbol{\lambda}_k\| + \|K_{k,s} P^{-\frac{1}{2}} z\| \leq \|\overline{u}_s\|. \tag{33}$$

Replacing $P = Y^{-1}$, assigning $K_{k,s} = L_{k,s} Y^{-1}$ and squaring the inequality on both sides yields:

$$||L_{k,s} Y^{-\frac{1}{2}} z||^2 \leq (\|\overline{u}_s\| - \|u_{k,s}^0 + D_{u_{k,s}} \boldsymbol{\lambda}_k\|)^2, \tag{34}$$

$$||L_{k,s} Y^{-\frac{1}{2}}||^2 ||z||^2 \leq (\|\overline{u}_s\| - \|u_{k,s}^0 + D_{u_{k,s}} \boldsymbol{\lambda}_k\|)^2, \tag{35}$$

$$L_{k,s} Y^{-1} L_{k,s}^T \leq (\|\overline{u}_s\| - \|u_{k,s}^0 + D_{u_{k,s}} \boldsymbol{\lambda}_k\|)^2, \tag{36}$$

what results in (30) by applying the Schur complement. □

LEMMA IV.3 Inequality (24) holds for all $\delta\boldsymbol{\lambda}_k$, when the following LMI holds:

$$\begin{bmatrix} Y & d_w \\ d_w^T & 1 \end{bmatrix} \geq 0, \ \forall d_w \in \mathscr{D}, \ w \in \{1,...,|\mathscr{D}|\}. \tag{37}$$

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

$$\delta\boldsymbol{\lambda}_k = \sum_w \mu_{k,w} d_w, \tag{38}$$

with weight $\mu_{k,w}$ for time k, and $\sum_w \mu_{k,w} = 1$, $\mu_{k,w} \geq 0$. Thus:

$$(\sum_w \mu_{k,w} d_w)^T P (\sum_w \mu_{k,w} d_w) \leq 1, \tag{39}$$

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

$$d_w^T P d_w \leq 1, \ \forall d_w \in \mathscr{D}, \ w \in \{1,...,|\mathscr{D}|\}. \tag{40}$$

Replacing $P = Y^{-1}$ and applying the Schur complement, (40) is converted into (37). It follows that $\mathscr{E}(P)$ is a stabilizing region of the saturated system for all permissible $\delta\boldsymbol{\lambda}_k$. □

The convex reformulation of the original constraints (22)-(24) leads to the LMI's (27), (30), (37), with variables L_k, Q_c and Y. On the other hand, the cost function (21) still contains the inverse variables $P = Y^{-1}$ and Q_c^{-1}, i.e. it has to be suitably transformed to obtain a solution at all.

LEMMA IV.4 The objective (21) with the reformulated constraints (27), (30), (37) can be cast into a minimization problem, in which the objective of the problem no longer contains the inverse forms of the variables L_k, Q_c and Y:

$$\min_{Y,\ L_k,\ Q_c} \quad \text{trace}(Q_c) + \text{trace}(Y) \tag{41}$$

$$\text{s.t.} \quad (27),\ (30),\ (37),\ Y > 0,\ Q_c > 0. \tag{42}$$

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

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

$$\text{trace}(W^{-1}) \geq \text{trace}(W)^{-1} \tag{43}$$

holds. With (43) and $P = Y^{-1}$, a lower bound of the cost function (21) can be given by:

$$f(Q_c, Y) := (\text{trace}(Q_c))^{-1} + (\text{trace}(Y))^{-1}. \tag{44}$$

Since (44) is strictly monotonically decreasing, it can be shown that the inverse function:

$$g(Q_c, Y) := \frac{1}{f(Q_c, Y)} = \frac{1}{(\text{trace}(Q_c))^{-1} + (\text{trace}(Y))^{-1}}, \tag{45}$$

is strictly monotonically increasing, and the maximization problem can be cast into the minimization problem:

$$\min_{Y, L_k, Q_c} g(Q_c, Y) \tag{46}$$

$$\text{s.t.} \quad (27),\ (30),\ (37),\ Y > 0,\ Q_c > 0. \tag{47}$$

An upper bound of $g(Q_c, Y)$ is obtained by applying the triangle inequality $\frac{1}{a+b} \leq \frac{1}{a} + \frac{1}{b}$ (for $a, b > 0$) to (45), yielding:

$$g(Q_c, Y) := \frac{1}{\text{trace}(Q_c)^{-1}} + \frac{1}{\text{trace}(Y)^{-1}}, \tag{48}$$

or respectively:

$$g(Q_c, Y) = \text{trace}(Q_c) + \text{trace}(Y), \tag{49}$$

what completes the proof. □

The optimization problem determines time-depending controller matrices $K_k = L_k Y^{-1}$ which guarantee a fast stabilizing transition between the trajectories, while handling the input constraints. The determined controllers are optimal for all $\boldsymbol{\lambda}_k, \bar{\boldsymbol{\lambda}} \in \textbf{Co}\{\mathscr{C}\}$, since they are the optimal solution of the convex semidefinit program. The computation can be performed with standard solvers like MOSEK [1].

V. ONLINE CONTROL STRATEGY

The online procedure starts with obstacle detection at time k^*, and it consists of two parts: In the first part, the n_p vertices p_l of a vertex representation $\mathscr{P}_{x,v} := \{p_l \in \mathbb{R}^{n_x} |\ l \in \mathbb{N} \leq n_p\} = \{p_1, ..., p_{n_p}\}$ of the obstacle polytope $\mathscr{P}_x$ are mapped into the homotopy space, denoted by $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$. This is motivated by selecting a desired trajectory $\bar{\boldsymbol{\lambda}}$, which is outside of the convex hull of $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$, denoted by $\mathscr{P}_{\bar{\boldsymbol{\lambda}}} := \textbf{Co}(\mathscr{P}_{\bar{\boldsymbol{\lambda}},v})$. The second part then selects a suitable $\bar{\boldsymbol{\lambda}}$ which also guarantees that the transitions from $\boldsymbol{\lambda}_{k^*}$ to $\bar{\boldsymbol{\lambda}}$ for the system (14) is free of collision.

A. Transforming $\mathscr{P}_{x,v}$ into the Homotopy Space

DEFINITION V.1 Let $\bar{\boldsymbol{\lambda}}(p_l) \in \mathbb{R}^{n_c}$ denote the vector of constant homotopy parameters which refers to the trajectory $\hat{x}(\bar{\boldsymbol{\lambda}}(p_l))$ that runs from the initial state x_0 through the vertex p_l of $\mathscr{P}_{x,v}$ to the final state x_f, while avoiding any intersection with the interior of $\mathscr{P}_x$. The set of all vertices p_l of $\mathscr{P}_{x,v}$ mapped into the homotopy space is denoted by $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$. △

The mapping of a vertex p_l into $\bar{\boldsymbol{\lambda}}(p_l)$ is performed for every vertex $p_l \in \mathscr{P}_{x,v}$, and is realized by the operation: $\mathsf{mapping}(p_l)$. The operation $\mathsf{mapping}(p_l)$ computes these homotopy values $\bar{\boldsymbol{\lambda}}(p_l)$ in an iterative procedure. Since a vertex p_l can be located between to time steps of the homotopic values $x_k(\boldsymbol{\lambda}_k)$ and $x_{k+1}(\boldsymbol{\lambda}_{k+1})$, this is also considered in the mapping procedure. For the sake of brevity, the detailed algorithmic procedure of $\mathsf{mapping}(p_l)$ is not explain in this work, but only summarized by the abstract Algorithm 1.

Algorithm 1 Vertex mapping

```
1: Given: $\hat{x}^i$, with $i \in \mathscr{M}$ and $\mathscr{P}_{x,v}$
2: for all $p_l \in \mathscr{P}_{x,v}$ do
3:     • $\bar{\boldsymbol{\lambda}}(p_l) \leftarrow \mathsf{mapping}(p_l)$
4:     • $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v} \leftarrow \mathscr{P}_{\bar{\boldsymbol{\lambda}},v} \bigcup \bar{\boldsymbol{\lambda}}(p_l)$
5: end for
```

B. Online trajectory determination

In the online selection, a desired homotopy value $\bar{\boldsymbol{\lambda}} := \bar{\boldsymbol{\lambda}}(p_l) \in \mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$ is selected such that the offline computed controllers K_{k^*+j}, $j \in \mathscr{J}$ drive the system from x_{k^*} to the trajectory $\hat{x}(\bar{\boldsymbol{\lambda}}(p_l))$ which passes the obstacle through the vertex p_l to the final state x_f. With respect to the $\boldsymbol{\lambda}$-space, this equals the transition from $\boldsymbol{\lambda}_{k^*}$ to $\bar{\boldsymbol{\lambda}}(p_l)$. That vertex p_l is chosen which incurs the lowest costs $J(\bar{\boldsymbol{\lambda}})$ for the resulting trajectory, and which is free of collision also for the transient behavior. 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) are replaced by $x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j})$ and $\tilde{u}_{k^*+j}(\boldsymbol{\lambda}_{k^*+j})$ according to (3) and (7). With (13) and (14), the transformed costs result to:

$$J(\bar{\boldsymbol{\lambda}}) = \sum_{j=0}^{N-1-k^*} (x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) - x_f)^T Q (x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) - x_f)$$

$$+ (\tilde{u}_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) - u_f)^T R (\tilde{u}_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) - u_f) \quad (50)$$

$$\text{s.t.} \quad x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) = x^0_{k^*+j} + D_{x_{k^*+j}} \boldsymbol{\lambda}_{k^*+j} \quad (51)$$

$$\tilde{u}_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) = u_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) + \delta u_{k^*+j} \quad (52)$$

$$\delta u_{k^*+j} = -K_{k^*+j}(\boldsymbol{\lambda}_{k^*+j} - \bar{\boldsymbol{\lambda}}) \quad (53)$$

$$\delta \boldsymbol{\lambda}_{k^*+1+j} = (I - D^{-1}_{x_{k^*+1+j}} B K_{k^*+j}) \delta \boldsymbol{\lambda}_{k^*+j} \quad (54)$$

$$j \in \mathscr{J}. \quad (55)$$

DEFINITION V.2 Let $\boldsymbol{\Lambda}$ denote the set of vertices $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$ 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 procedure is shown in Algorithm 2: Starting from the optimal trajectory with $\boldsymbol{\lambda}_k = [0]^{n_c}$ at time $k := k^*$, the optimal trajectory is checked against collisions with the state space obstacle $\mathscr{P}_x$ (line 4). If the trajectory intersects with the obstacle, the vertices p_l of $\mathscr{P}_{x,v}$ are mapped into the homotopy space according to Alg. 1 (line 5), followed by sorting the set $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$ in order of increasing costs $J(\bar{\boldsymbol{\lambda}})$, resulting in the set $\boldsymbol{\Lambda}$ (line 6). Now, the desired homotopy value is set to $\bar{\boldsymbol{\lambda}} := \Lambda(1)$, the trajectory $x_{k+j}(\boldsymbol{\lambda}_{k+j})$ is computed according to (3) and (14) (line 8-10) and then checked against collisions with $\mathscr{P}_x$ (line 11). 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 chosen by incrementing i of $\Lambda(i)$. Hence, a new trajectory passing the obstacle along the vertex $\Lambda(2)$ (with higher costs) is computed and checked against collision. This procedure is very quick, since besides of the vertex mapping into the homotopy space, the rest (line 7-end) of Alg. 2 simply checks trajectories successively against collisions with $\mathscr{P}_x$. This has to be done in the worst case for a maximum of $|\boldsymbol{\Lambda}|$ trajectories, where $|\boldsymbol{\Lambda}|$ denotes the cardinality of $\boldsymbol{\Lambda}$. If no feasible $\bar{\boldsymbol{\lambda}}$ can be determined, i.e. no corner of $\mathscr{P}_x$ that can be passed without collision, Alg. 2 terminates with this result. Consequently, an emergency braking routine has to be started which stops the system in a save, collision free position.

VI. NUMERICAL EXAMPLE

In continuation of the robot scenario described at the beginning, the robotic system is realized as a simple position model with the $n_x = 3$ states representing the three position coordinates (x,y,z) of a robotic end-effector, and the $n_u = 3$ inputs can affect the position. The matrices A, B of the discrete-time system are determined to have the following structure:

$$A = 1e^{-3} \cdot \begin{bmatrix} 953 & 24 & 12 \\ 24 & 911 & 9.4 \\ 12 & 9.4 & 965 \end{bmatrix}, \; B = 1e^{-4} \cdot \begin{bmatrix} 967 & -7 & 40 \\ -7 & 1026 & -46 \\ 40 & -46 & 1057 \end{bmatrix}, \quad (56)$$

i.e. the system stable and fully controllable in one step. The weighting matrix Q of the cost function in (4) is chosen as the identity matrix, and $R = 10 \cdot Q$. The upper bound on the norm of each component s of the input vector, as stated in (18), is chosen as $\|\bar{u}_s\| = 15$. The time horizon is selected to $N = 60$, the initial state to $x_0 = x_s = [0,0,0]^T$, and final state to $x_N = x_f = [5,5,5]^T$, realized by $u_N = u_f = [0.5, 2.7, 0.7]^T$. With $n_c = 3$, the set of trajectories is chosen to: $\mathscr{X} = \{(\hat{x}^0,\hat{u}^0),(\hat{x}^1,\hat{u}^1),(\hat{x}^2,\hat{u}^2),(\hat{x}^3,\hat{u}^3)\}$. The pair $(\hat{x}^0,\hat{u}^0)$ denotes the optimal solution of (4) for the case that no obstacle $\mathscr{P}_x$ is present. This trajectory is shown in Fig. 1 and is colored in magenta. The other n_c base trajectories in $\mathscr{X}$ are colored in black, and the polytope $\mathscr{P}_x$ is shwon as green box in Fig. 1. The scenario is to drive the robotic end-effector from x_0 to x_f, while at $k^* = 10$ the obstacle $\mathscr{P}_x$ is detected and remains static until the end time. The green trajectory in Fig. 1 shows that the optimal trajectory (magenta) is followed up to the detection of the obstacle at $k^* = 10$.

For the remaining 50 steps, the blue trajectory (see Fig. 1) is determined by the homotopy approach, passing the obstacle along its lower right vertex. While the trajectory transits to its final value $\bar{\boldsymbol{\lambda}}$, the determined controllers ensure

Algorithm 2 Online control procedure

```
1:  Set: $k := k^*$
2:  Given: $x^0_k$, $\boldsymbol{\lambda}_k = [0]^{n_c}$, $\bar{\boldsymbol{\lambda}} = [0]^{n_c}$, $\mathscr{P}_x$, $\mathscr{P}_{x,v}$,
3:         $j \in \mathscr{J} := \{0,...,N-1-k\}$
4:  if $\exists j \in \mathscr{J} : x_{k+j}(\boldsymbol{\lambda}_{k+j}) \in \mathscr{P}_x$ then
5:      • Map all vertices $p_l$ into the $\boldsymbol{\lambda}$-space by Alg. 1
6:      • Compute the set $\boldsymbol{\Lambda}$ ordered according to $J(\bar{\boldsymbol{\lambda}})$
7:      for $i \in \{1,\dots,|\boldsymbol{\Lambda}|\}$ do
8:          • Compute for all $j \in \mathscr{J}$ the states
9:            $x_{k+j}(\boldsymbol{\lambda}_{k+j})$ according to (3) and (14)
10:           with homotopy value $\bar{\boldsymbol{\lambda}} := \Lambda(i)$.
11:         if $\exists j \in \mathscr{J} : x_{k+j}(\boldsymbol{\lambda}_{k+j}) \in \mathscr{P}_x$ then
12:             Trajectory from $\boldsymbol{\lambda}_{k+j}$ to $\bar{\boldsymbol{\lambda}}$ is not feasible
13:         else
14:             Trajectory from $\boldsymbol{\lambda}_{k+j}$ to $\bar{\boldsymbol{\lambda}}$ is an optimized
15:              feasible trajectory with $\bar{\boldsymbol{\lambda}} = \Lambda(i)$.
16:             break
17:         end if
18:     end for
19: end if
```

that the input constraints are satisfied. For comparison, the example is also solved by applying mixed integer quadratic programming (MIQP). The problem is formulated by the well-known Big-M method to handle the non-convexity, and the solution is shown as the red trajectory in Fig. 1. This trajectory passes the obstacle along the lower edge o $\mathscr{P}_x$. While the cost of the red trajectory is a lower compared to the blue one, a significant reduction of the computation times can be observed to the advantage of the homotopy approach: The homotopy approach provides computation times around 1.7 milliseconds (ms) using a Matlab implementation, while the MIQP method has an average computation time of 800 ms. The MIQP-based method is also implemented in Matlab with embedded solution of the optimization problem by the CPLEX solver. As a third alternative, an implementation as a nonlinear problem with approximating the polytope as an ellipsoid, was used and the solution takes approximately 2.5 sec. All methods are performed on an Intel Core i7 @3.4GHz.

VII. CONCLUSIONS

The paper has shown that homotopy properties can be used to quickly compute optimized trajectories satisfying input constraints, for the nonconvex problem of obstacle avoidance in trajectory planning. The key idea is to separate the problem into an offline part, including controllers synthesized by semi-definite programming, and an online part, including suitable reference trajectories (in terms of desired homotopy values) are selected to pass the obstacle without collision. The proposed method computes an optimized solution significantly faster then methods like MIQP or NLP (as would be used within MPC). It is shown that computation times are nearly 3 orders of magnitude smaller, allowing the method to be applicable to systems where small computation times are crucial.

Future work extends this method to passing the obstacle also across edges, and to further reduce the costs of the circumvented trajectory. The method will also be extended to the case of time varying obstacles and final states.

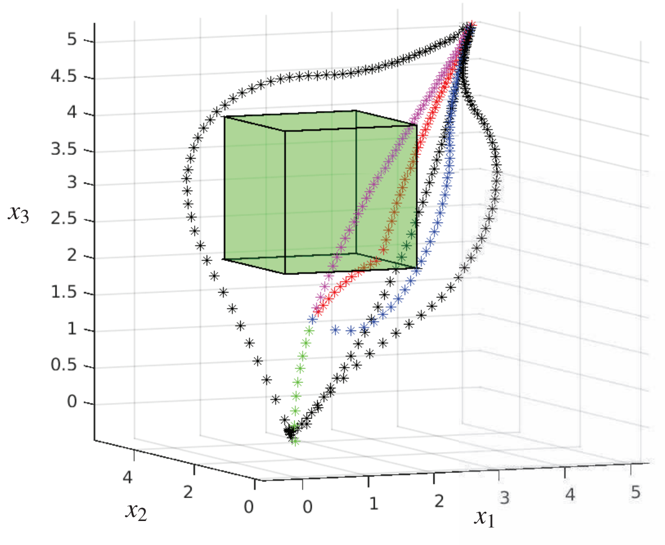

Fig. 1. Simulation of the optimal trajectory $\hat{x}^0$ (magenta), the base trajectories $\hat{x}^1$, $\hat{x}^2$, $\hat{x}^3$ (black), a part of the optimal trajectory (green), up to $k^* = 10$, the trajectory obtained by the homotopy approach $\hat{x}(\boldsymbol{\lambda})$ (blue), and the solution by MIQP (red). The polytope $\mathscr{P}_x$ is the green area.

ACKNOWLEDGMENT

Partial financial support by the European Commission through the project UnCoVerCPS (grant No. 643921) is gratefully acknowledged.

REFERENCES

[1] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 51).*, 2015.

[2] L. Blackmore and B. Williams. Optimal manipulator path planning with obstacles using disjunctive programming. In *American Control Conference*, pages 3200–3202, 2006.

[3] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994.

[4] H. Ding, G. Reissig, and O. Stursberg. Increasing Efficiency of Optimization-based Path Planning for Robotic Manipulators. In 50th *IEEE Conf. on Decision and Control*, pages 1399–1404, 2011.

[5] D. Ferguson and A. Stentz. Anytime, dynamic planning in high-dimensional search space. In *IEEE Conf. on Robotics and Automation*, pages 1310–1315, 2007.

[6] R. Gondhalekar and J.-I. Imura. Least-restrictive move-blocking model predictive control. *Automatica*, 46(7):1234 – 1240, 2010.

[7] Tingshu Hu, Zongli Lin, and Ben M. Chen. Analysis and design for discrete-time linear systems subject to actuator saturation. *Systems & Control Letters*, 45(2):97 – 112, 2002.

[8] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *J. of Robotics Research*, 30(7):846–894, 2011.

[9] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *J. of Robotics Research*, 5:90–98, 1986.

[10] C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schloeder. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *J. of Process Control*, 22(3):540 – 550, 2012.

[11] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE Conf. on Robotics and Automation*, pages 995–1001, 2000.

[12] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, 2000.

[13] W.S. Newman and M.S. Branicky. Real-time configuration space transforms for obstacle avoidance. *J. of Robotics Research*, 6:650–667, 1991.

[14] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[15] K. Reif, K. Weinzierl, A. Zell, and R. Unbehauen. A homotopy approach for nonlinear control synthesis. *IEEE Tr. on Automatic Control*, 43(9):1311–1318, 1998.

[16] A. Richards. Fast model predictive control with soft constraints. In *European Control Conference*, pages 1–6, 2013.

[17] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*, pages 2603–2608, Sept 2001.

[18] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Tr. on Control Systems Technology*, 18(2):267–278, 2010.

[19] M.N. Zeilinger, C.N. Jones, and M. Morari. Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization. *IEEE Tr. on Automatic Control*, 56(7):1524–1534, 2011.