Fast Optimizing Control for Non-Convex State Constraints using Homotopy Properties

Damian Kontny[1] and Olaf Stursberg[1]

Abstract— **For optimal point-to-point control of linear systems, this paper investigates how control strategies should be rendered online if non-convex state constraints are suddenly detected during execution. The challenge is to compute a modified strategy with low effort while maintaining a close-to-optimal control performance. The proposed solution is to synthesize offline a range of trajectories which are homotopic to the optimal unconstrained solution. Upon detection of the state constraints (which may represent an obstacle to be circumvented by a mobile vehicle), a sub-optimal yet feasible homotopic trajectory is selected by a fast iteration which avoids to solve a time-demanding constrained optimization problem online.**

I. INTRODUCTION

The scope of this paper is to control linear dynamic systems from an initial state into a target state while minimizing control costs and satisfying time-varying non-convex state constraints. A motivating example is to steer a vehicle or a robot end effector cost-optimally into a target, assuming that – during motion – a sensor detects an obstacle with which a collision must be avoided. The task is then to find a feasible path and control strategy leading to a small increase of motion costs very quickly. The obvious solution is to formulate and solve a state-constrained open-loop optimal control problem. Even for optimization over relatively small time horizons, a timely solution will not be feasible for many problem instances due to the presence of the (often) non-convex constraints. The objective of this paper is thus to explore means to decompose the problem into an online and an offline part such that the computation times for the latter are acceptably small.

Constrained point-to-point optimization problems are commonly solved by direct optimization methods for reasons of speed. One established method based on the solution of constrained optimization problems is *model predictive control* (MPC) (see e.g. [12]). In recent years, considerable effort has been spent on efficient solution of the optimization problems: efficiency may be increased through replacing state constraints by penalty terms or barrier functions [17], [15], or by use of *move-blocking strategies* which fix the inputs over a certain period of time [6]. The exploitation of block diagonal structures in the Hessian and Jacobian matrices, combined with direct multiple shooting can reduce computation time as well [9], [3].

Partial financial support by the European Commission through the project UNCOVERCPS (grant No. 643921) and the DFG through the project ROCS-Grid is gratefully acknowledged.

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

While for convex search spaces significant progress has been made, the situation for the steering scenario with collision avoidance (as mentioned above) is different, as the presence of obstacles leads to non-convexity of the admissible space. The option to solve the problem by techniques of nonlinear programming (such as SQP) often leads to large computation times and trajectories which are far from optimal. An alternative is the formulation of mixed-integer programs (MIP), in which the admissible space is split into sets of convex regions, and a feasible sequence of such regions is mapped into binary variables [1], [4]. 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 of *explicit MPC* where developed [16], [18]. 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 of affine solution. For this method, the reduced effort for online computation is paired with high memory requirements, and thus limited to low system dimensions and/or time horizons.

With respect to trajectory planning in non-convex spaces, a variety of relevant publications originate from the field of human robot interaction. An important and established class of path planning algorithms is that of *potential field methods* [8], which however does not aim at optimal solutions. Cell-decomposition methods compute feasible paths by a sequence of free cells [13], and apply graph search algorithms afterwards. These methods are restricted by the combinatorial complexity arising from the state dimension and from time-varying constraints. For multiple DOF-robots, the problem of mapping obstacles into the configuration space is critical, and algorithms like *rapidly Exploring Random Trees* (RRT) [10] and RRT* [7], [5] were developed. These methods search for collision-free trajectories in the configuration space by sampling, but they commonly do not consider the system dynamics.

In contrast to existing work, this paper approaches the named problem by a method which computes offline linear time-varying quadratic state feedback regulators for the purpose of transitioning between homotopic trajectories. In online execution, values of the homotopy parameters for excluding collision with the obstacle are determined algorithmically with low effort. 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 [14], [11].

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 modification of control strategies (Sec. V), and show numerical results (Sec. VI).

II. HOMOTOPIC FUNCTIONS

The class of systems under consideration in this paper are discrete-time linear systems:

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

with time index $k \in \mathbb{N}_0$, state vector $x_k \in \mathbb{R}^{n_x}$, input vector $u_k \in \mathbb{R}^{n_u}$, as well as state and input matrices A and B. For a finite time domain $T = \{0,1,\dots,N\}$, $N \in \mathbb{N}$, let the state and input trajectories be denoted by $\hat{x} = (x_0,...,x_N)$ and $\hat{u} = (u_0,...,u_{N-1})$. Now consider n_c+1 different pairs of input and state trajectories $(\hat{u}^i,\hat{x}^i)$, $i \in \mathscr{M} := \{0,1,...,n_c\}$ for (1). The different trajectories are collected in a set $\mathscr{X} := \{\hat{x}^0,...,\hat{x}^{n_c}\}$, where $\hat{x}^0$ denotes a trajectory which is optimal with respect to a given performance measure, and the other trajectories are called *base trajectories*. They are chosen to span a region around $\hat{x}^0$, in which the circumvention of an obstacle can take place. The *base trajectories* can be determined by, e.g., an open-loop optimal control problem with different weights of the state vector or by considering a region of possibly occurring obstacle positions . Furthermore, these base trajectories can be readily computed offline with convex constraints like speed or acceleration limits. All trajectories of $\mathscr{X}$ have the same initial and final states $x_0^i = x_s$, $x_N^i = x_f$, $i \in \mathscr{M}$. 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 *homotopic* functions 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:

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

△

The definition states that homotopic trajectories are linear interpolations between the optimal trajectory $\hat{x}^0$ and the *base trajectories* $\hat{x}^i$, $i \in \mathscr{M}$. If e.g. $\lambda^i = 1$ is chosen, while the other components of $\boldsymbol{\lambda}$ are zero, trajectory $\hat{x}^i$ is obtained by (2). If convex constraints were made on the base trajectories the homotopic trajectories also satisfy these constraints because of the linearity in interpolation. Since (2) relates to complete trajectories, a homotopic state $x_k(\boldsymbol{\lambda}_k)$ at a single point of time k lies in between the states x_k^i, $i \in \mathscr{M}$, and is identified by the homotopy value $\boldsymbol{\lambda}_k$. Therefore the homotopic states as well as the inputs at time k can be written as:

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

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, and matrices $D_{x_k} = (x_k^1 - x_k^0,...,x_k^{n_c} - x_k^0) \in \mathbb{R}^{n_x \times n_c}$, and likewise $D_{u_k} = (u_k^1 - u_k^0,...,u_k^{n_c} - u_k^0) \in \mathbb{R}^{n_u \times n_c}$ for $k \in T$. If $x_k(\boldsymbol{\lambda}_k)$ has a constant homotopy value over time denoted by $\bar{\boldsymbol{\lambda}}$, the resulting trajectories are denoted by $\hat{x}(\bar{\boldsymbol{\lambda}})$, and $\hat{u}(\bar{\boldsymbol{\lambda}})$.

III. PROBLEM DEFINITION

For defining the problem, assume the situation that the system follows the optimal trajectory $\hat{x}^0$ from x_0 to x_N, and that an intermediate time $k^* \in \{1,\dots,N-1\}$ exists at which an obstacle $\mathscr{P}_x$ is detected, preventing to follow $\hat{x}^0$ further (i.e. $x_k^0 \in \mathscr{P}_x$ for at least one $k \in \{k^*+1,\dots,N-1\}$). Let the obstacle be defined as a polytopic region $\mathscr{P}_x := \{x |\ Cx \leq d\} \subseteq \mathbb{R}^{n_x}$, with $C \in \mathbb{R}^{c \times n_x}$ and $d \in \mathbb{R}^c$. Here, $\mathscr{P}_x$ is assumed to be static for $k \in \{k^*,\dots,N\}$. Obviously, $x_{k^*}^0 \notin \mathscr{P}_x$ and $x_N^0 \notin \mathscr{P}_x$ is required to admit a feasible solution. Upon obstacle detection in $x_{k^*} = x_{k^*}^0$, the goal is to determine feasible optimized trajectories $\hat{x}^* = (x_{k^*}^*,...,x_N^*) \in \mathbb{R}^{n_x \times (N+1-k^*)}$ and $\hat{u}^* = (u_{k^*}^*,...,u_N^*) \in \mathbb{R}^{n_u \times (N-k^*)}$, which avoid collision but ensure that the final state is $x_N = x_f$. The selection of the best among the feasible trajectories is based on the following quadratic performance criterion:

$$\begin{aligned} J(x_{k^*+j},\ u_{k^*+j}) = &\sum_{j=0}^{N-1-k^*} (x_{k^*+j} - x_f)^T Q (x_{k^*+j} - x_f) \\ &+ (u_{k^*+j} - u_f)^T R (u_{k^*+j} - u_f), \end{aligned} \tag{4}$$

with positive-definite weighting matrices $Q \in \mathbb{R}^{n_x \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$. With $\mathscr{J} := \{0,...,N-1-k^*\}$, the problem can then be stated as:

$$\begin{aligned} &\min_{x_{k^*+j},\ u_{k^*+j}} \quad J(x_{k^*+j},\ u_{k^*+j}) \\ &\text{s.t.} \quad (1),\ x_{k^*} = x_{k^*}^0,\ x_N = x_f,\ x_{k^*+j} \notin \mathscr{P}_x. \quad \forall j \in \mathscr{J}. \end{aligned} \tag{5}$$

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 non-convex. As remarked in Sec. I and illustrated in Sec. VI for an example, the solution by MIP, or specifically mixed-integer quadratic programming (MIQP), is a common approach. It leads to large computation times, due to formulating collision avoidance by binary variables for any $j \in \mathscr{J}$. The objective of the method presented below is to find a close to optimal solution with significantly lower computational effort as with MIQP. The idea for circumvention of $\mathscr{P}_x$ is to: (i) select a homotopic trajectory $\hat{x}(\bar{\boldsymbol{\lambda}})$, which avoids an intersection with $\mathscr{P}_x$, and (ii) to design controllers offline that realise the transition to the trajectory $\hat{x}(\bar{\boldsymbol{\lambda}})$ from the actual executed one. For enabling that a solution to the problem can be found in step (i), the following assumption is stated.

ASSUMPTION III.1 Let the set of trajectories $\mathscr{X}$ contain at least one trajectory $\hat{x}^i$, $i \in \mathscr{M}$, such that for $\hat{x}^i_{k^*+j} \in \hat{x}^i$ no collision with the obstacle occurs: $x^i_{k^*+j} \notin \mathscr{P}_x$, $\forall j \in \mathscr{J}$.

This assumption is immediately justified by the fact that one cannot hope to find a feasible circumvention of $\mathscr{P}_x$ if the whole admissible space (as constructed by the choice of $\mathscr{X}$) is blocked. The assumption is not sufficient for finding a feasible solution, since it must be ensured also in step (2) that the transition to $\hat{x}(\bar{\boldsymbol{\lambda}})$ is achieved without intersecting $\mathscr{P}_x$.

IV. 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 new vectors *bar*$\boldsymbol{\lambda}$. Towards this goal, the dynamic system is first expressed in the space of homotopy parameters, leading to a linear time-varying (LTV) system. Then, semi-definite programming synthesizes state feedback controllers for realizing transitions in the $\boldsymbol{\lambda}$-space.

A. Transformation into the homotopy space

Consider the task of steering (1) from a state x_{k^*} (corresponding to a momentary vector $\boldsymbol{\lambda}_k$) to a future state on a homotopic trajectory referring to $\bar{\boldsymbol{\lambda}}$. While the inputs for the currently executed trajectory and the targeted one are known from the homotopy function (2), the transition between these trajectories requires additional inputs δu_k, leading to:

$$\tilde{u}_k(\boldsymbol{\lambda}_k) := u_k(\boldsymbol{\lambda}_k) + \delta u_k. \tag{6}$$

This superposition may lead to signals $\tilde{u}_k(\boldsymbol{\lambda}_k)$ which are not in the set of homotopic input trajectories according to (3).

With the homotopic states $x_k(\boldsymbol{\lambda}_k)$ in (3) and the inputs $\tilde{u}_k(\boldsymbol{\lambda}_k)$ in (6), the system dynamics is written as:

$$x_{k+1}(\boldsymbol{\lambda}_{k+1}) = Ax_k(\boldsymbol{\lambda}_k) + B\tilde{u}_k(\boldsymbol{\lambda}_k), \tag{7}$$

for $k \in \{0,...,N-1\}$.

ASSUMPTION IV.1 Matrix B is invertible

This assumption ensures that any homotopic state $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ at $k+1$, can always be reached from a homotopic state $x_k(\boldsymbol{\lambda}_k)$ with the system dynamics (1) and the input (6).

For transferring (7) into the $\boldsymbol{\lambda}$-space, the combination of (3), (6), and (7) leads to:

$$D_{x_{k+1}}\boldsymbol{\lambda}_{k+1} + x^0_{k+1} = A(x^0_k + D_{x_k}\boldsymbol{\lambda}_k) + B(u^0_k + D_{u_k}\boldsymbol{\lambda}_k + \delta u_k) \tag{8}$$

$$\Leftrightarrow \quad D_{x_{k+1}}\boldsymbol{\lambda}_{k+1} = (AD_{x_k} + BD_{u_k})\boldsymbol{\lambda}_k + B\delta u_k \tag{9}$$

$$\Leftrightarrow \quad D_{x_{k+1}}\boldsymbol{\lambda}_{k+1} = D_{x_{k+1}}\boldsymbol{\lambda}_k + B\delta u_k. \tag{10}$$

The last equation represents an LTV-system with state $\boldsymbol{\lambda}_k$, input δu_k, and a time-dependent and bounded matrix $D_{x_{k+1}}$ for $k \in \{0,...,N-1\}$.

ASSUMPTION IV.2 The number of *base trajectories* equals the system dimension: $n_c = n_x$.

This assumption[1] is reasonable in order to obtain a full-dimensional space for circumventing $\mathscr{P}_x$. A positive side-effect is that $D_{x_k} \in \mathbb{R}^{n_x \times n_x}$ and its inverse $D^{-1}_{x_k}$ is non-singular (since the trajectories in $\mathscr{X}$ differ, see Sec. II). Thus, $\boldsymbol{\lambda}_{k+1}$ follows explicitly from (10):

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + D^{-1}_{x_{k+1}} B\delta u_k. \tag{11}$$

This equation describes the transition between different homotopic trajectories, and leads to $x_{k+1}(\boldsymbol{\lambda}_{k+1})$ via (3).

B. Optimization-based controller synthesis

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 synthesized offline:

$$\delta u_k := -K_k(\boldsymbol{\lambda}_k - \bar{\boldsymbol{\lambda}}), \tag{12}$$

The control law drives the system to a desired value $\bar{\boldsymbol{\lambda}}$, or respectively a desired offline computed homotopic trajectory $\hat{x}(\bar{\boldsymbol{\lambda}})$. The following part explains the synthesis of K_k for the LTV-system (11) based on Lyapunov stability conditions and a quadratic performance functional. ($\bar{\boldsymbol{\lambda}}$ is assumed to be known here; its computation is covered later in Sec. V).

The *closed-loop delta-system* of (11) with $\delta\boldsymbol{\lambda}_{k+1} := \boldsymbol{\lambda}_{k+1} - \bar{\boldsymbol{\lambda}}$ and the control law (12) becomes:

$$\delta\boldsymbol{\lambda}_{k+1} = (I - D^{-1}_{x_{k+1}} BK_k)\delta\boldsymbol{\lambda}_k. \tag{13}$$

As performance measure for this system, consider the quadratic cost function with symmetric and positive-definite weighting matrices Q_C and R_C:

$$J(\delta\boldsymbol{\lambda}_k, \delta u_k) = \sum_{k=0}^{N-1} \delta\boldsymbol{\lambda}_k{}^T Q_C \delta\boldsymbol{\lambda}_k + \delta u_k{}^T R_C \delta u_k, \tag{14}$$

or with insertion of (12) written as:

$$J(\delta\boldsymbol{\lambda}_k, K_k) = \sum_{k=0}^{N-1} \delta\boldsymbol{\lambda}_k{}^T (Q_C + K_k{}^T R_C K_k)\delta\boldsymbol{\lambda}_k. \tag{15}$$

With a matrix $P = P^T > 0 \in \mathbb{R}^{n_c \times n_c}$, let a Lyapunov function:

$$V_k = \delta\boldsymbol{\lambda}_k^T P \delta\boldsymbol{\lambda}_k, \tag{16}$$

be given, with which the LTV-system (13) is stable, if the following condition holds:

$$V_{k+1} \leq V_k - \delta\boldsymbol{\lambda}_k{}^T (Q_C + K_k{}^T R_C K_k)\delta\boldsymbol{\lambda}_k. \tag{17}$$

LEMMA IV.1 For the LTV-system (13) and a Lyapunov function (16) with $P = P^T > 0$, an upper bound of the finite horizon costs (15) is given by:

$$J(P) = \text{trace}(P). \tag{18}$$

Proof. 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

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

formulated by (15) when time advances from k to $k+1$. Summing up the decrease over all time steps yields:

$$\sum_{k=0}^{N-1} V_{k+1} - V_k \leq \sum_{k=0}^{N-1} -\delta\boldsymbol{\lambda}_k{}^T (Q_C + K_k{}^T R_C K_k)\delta\boldsymbol{\lambda}_k, \quad (19)$$

and thus:

$$V_0 - V_N \geq J(\delta\boldsymbol{\lambda}_k, K_k). \quad (20)$$

Since all $\hat{x}^i$ for $i \in \mathscr{M}$ 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, for all $\boldsymbol{\lambda}_N$, it follows that $\delta\boldsymbol{\lambda}_N = 0$ and $V_N = 0$, such that:

$$V_0 \geq J(\delta\boldsymbol{\lambda}_k, K_k). \quad (21)$$

The upper bound of V_0 can be minimized independently of $\delta\boldsymbol{\lambda}_0$ by minimizing over the trace of matrix P, such that (18) holds. □

As argued in [2], LTV-systems can be interpreted as linear parameter varying (LPV) systems with bounded uncertainties, such that the controller synthesis described there can be applied to LTV-systems. It is known for LPV-systems, that bounded uncertainties of the system matrices can be over-approximated by matrix polytopes. Stability for the controlled LPV-system is then guaranteed for all variations of the uncertain parameters, if the control law is stabilizing for the vertices of the over-approximating matrix polytope. For the system (13) considered here, the matrices D_{x_k} can be interpreted as bounded uncertainties such that techniques for LPV-systems become applicable. However, unlike the LPV case, the matrices D_{x_k} are completely known at every time step (due to the offline computation of the trajectories $\hat{x}^i$), implying that a polytopic approximation of the matrices D_{x_k} over k is unnecessary. Instead, the stability condition (17) has to be satisfied for D_{x_k} in any k. The optimization problem can be written for $k \in \{0,...,N-2\}$ as:

$$\min_{P,\ K_k} \quad \text{trace}(P) \quad (22)$$

$$\text{s.t.} \quad (17),\ (16) \text{ with } P \geq 0. \quad (23)$$

For $k = N-1$, the optimization needs not to be solved, since the offline computed input $u_{N-1}(\boldsymbol{\lambda}_{N-1})$ fixes this input already to the value required to bring the system into the final state $x_N(\boldsymbol{\lambda}_N) = x_f$.

LEMMA IV.2 The non-convex constraint (17) can be transformed into:

$$(I - D_{x_{k+1}}^{-1} BK_k)^T P (I - D_{x_{k+1}}^{-1} BK_k) - P + Q_C + K_k{}^T R_C K_k \leq 0, \quad (24)$$

and holds 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 $k = 0,...,N-2$. Then, also the following LMI holds:

$$\begin{bmatrix} Y & (Y - D_{x_{k+1}}^{-1} BL_k)^T & Y^T & L_k{}^T \\ Y - D_{x_{k+1}}^{-1} BL_k & Y & 0 & 0 \\ Y & 0 & Q_C^{-1} & 0 \\ L_k & 0 & 0 & R_C^{-1} \end{bmatrix} > 0, \quad (25)$$

in which 0 denotes zero matrices of appropriate dimensions.

Proof. Equation (17) can be reformulated by inserting $V_k = \delta\boldsymbol{\lambda}_k{}^T P \delta\boldsymbol{\lambda}_k$ and likewise V_{k+1} into (17), and then by replacing $\delta\boldsymbol{\lambda}_{k+1}$ according to (13), to obtain (24). If the latter is multiplied from the left and right by Y, and if the substitution $Y = P^{-1}$ is used, then a strict formulation of the inequality (24) is obtained to:

$$\begin{aligned}(Y - D_{x_{k+1}}^{-1} BK_k Y)^T Y^{-1} (Y - D_{x_{k+1}}^{-1} BK_k Y) - Y \\ + Y^T Q_C Y + Y^T K_k{}^T R_C K_k Y < 0.\end{aligned} \quad (26)$$

Applying the Schur complement to (26) yields:

$$\begin{bmatrix} Y & (Y - D_{x_{k+1}}^{-1} BK_k Y)^T & Y^T & K_k Y^T \\ Y - D_{x_{k+1}}^{-1} BK_k Y & Y & 0 & 0 \\ Y & 0 & Q_C^{-1} & 0 \\ K_k Y & 0 & 0 & R_C^{-1} \end{bmatrix} > 0, \quad (27)$$

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

The cost function (18) depends on P and the matrix inequality (25) on Y. A closed upper bound $\tilde{P}$ on P can be obtained from the non-strict inequality $\tilde{P} \geq Y^{-1}$. Applying the Schur complement, this inequality is equivalent to:

$$\begin{bmatrix} \tilde{P} & I \\ I & Y \end{bmatrix} \geq 0. \quad (28)$$

The overall optimization problem for the controller synthesis of the LTV-system (13) can now be summarized to:

$$\min_{\tilde{P},\ Y,\ L_k} \quad \text{trace}(\tilde{P}) \quad (29)$$

$$\text{s.t.:} \quad (25),\ (28),\ \tilde{P} \geq 0,\ k \in \{0,...,N-2\}$$

The result of the optimization is a time-varying linear quadratic regulator (LQR) for the LTV-system (13). The offline controller synthesis can be carried out with solvers like MOSEK.

V. ONLINE CONTROL STRATEGY

The online procedure, initiated upon detection of an obstacle at time k^*, consists of two parts: the first is to map 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 $\mathscr{P}_x$ into the homotopy space. The result is denoted by $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$. The motivation for this mapping is to select a desired trajectory $\bar{\boldsymbol{\lambda}}$ that is outside of the convex hull of $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$, denoted by $\mathscr{P}_{\bar{\boldsymbol{\lambda}}} := \text{conv}(\mathscr{P}_{\bar{\boldsymbol{\lambda}},v})$. The second part selects $\bar{\boldsymbol{\lambda}}$ by identifying a trajectory which is free of collision when the system (13) transitions from $\boldsymbol{\lambda}_{k^*}$ to $\bar{\boldsymbol{\lambda}}$.

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 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 $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$. △

Generally, the mapping of a vertex from the state space to the homotopy space is not unique. The reason is, that a point p_l in the state space can be mapped for different times k. This can be seen from (3), if p_l is used on the left side, and the equation is solved for $\boldsymbol{\lambda}_k(p_l)$ for different times k. The region in which the vertex p_l can be described by n_c-1 relevant trajectories is first selected, and $\hat{x}(\bar{\boldsymbol{\lambda}}(p_l))$ is then determined by (3) for the time k, which is closest to p_l. For illustration, Fig. 1 shows the procedure in $\mathbb{R}^2$ for three time steps of three trajectories $\hat{x}^0=[x_0^0,x_1^0,x_2^0]$, $\hat{x}^1=[x_0^1,x_1^1,x_2^1]$, and $\hat{x}^2=[x_0^2,x_1^2,x_2^2]$, and for homotopy parameters $\boldsymbol{\lambda}_k(p_l)=[\lambda_k^1(p_l),\lambda_k^2(p_l)]^T$. The vertex p_l can be described by a homotopy parameter vector at time $k=2$ resulting in values $\boldsymbol{\lambda}_2(p_l)\approx[0.6,\ 0]^T$ or by a vector of $\boldsymbol{\lambda}_3(p_l)\approx[0.7,\ 0.3]^T$. To avoid multiple mapping of p_l to $\boldsymbol{\lambda}_k(p_l)$, the parameter $\lambda_k^2(p_l)=0$ is set to zero for all k, and $\lambda_k^1(p_l)$ is only determined for the k which is closest to p_l, i.e. for $k=2$. The parameter vector thus becomes $\boldsymbol{\lambda}_2(p_l)=[\lambda_2^1(p_l),0]^T\approx[0.6,\ 0]^T$. If instead the other case is chosen with $\lambda_k^1(p_l)$ set to zero for all k, the mapping would be impossible since $\lambda_k^2(p_l)\notin[0,1]$ for any k. The following definition describes the combinatorial possibilities of relevant trajectories that span a region in which p_l can be located.

DEFINITION V.2 Let a set of variables $c_i\in\{0,1\}$, $i\in\{1,\dots,n_c\}$ denote whether the i-th parameter in $\bar{\boldsymbol{\lambda}}(p_l)$ is selected ($c_i=1$) or not. These variables are arranged in a diagonal matrix $C_s=diag(c_1,\dots,c_{n_c})$ to establish the selection by:

$$\bar{\boldsymbol{\lambda}}(p_l,C_s):=C_s\cdot\bar{\boldsymbol{\lambda}}(p_l). \tag{30}$$

Let $\mathscr{G}$ denote the set of matrices C_s, $s\in\{1,\dots,N_p\}$, for all possible combinations of chosen parameters for which $\sum_i c_i=n_c-1$. The cardinality of $\mathscr{G}$ is $N_p=\binom{n_c}{n_c-1}$. $\triangle$

A vertex p_l can be located in between the subspaces reachable in two adjacent points of time, as illustrated in Fig. 2 for a case related to the one addressed in Fig. 1: Consider the case that only the upper trajectory in Fig. 1 is selected by the matrix $C_1\in\mathscr{G}$. Thus, the optimal trajectory $\hat{x}^0$ and the one *base trajectory* $\hat{x}^1$ remain relevant, and $\bar{\boldsymbol{\lambda}}(p_l,C_1)$ has only one valid entry. Fig. 2 shows that the trajectories have a common initial state $x_0=x_s$ and final state $x_N=x_f$. Now to determine the value of $\bar{\boldsymbol{\lambda}}(p_l,C_1)$, the vertex $p_l\in\mathbb{R}^2$ of $\mathscr{P}_{x,v}$ can be described in the space of two linearly independent

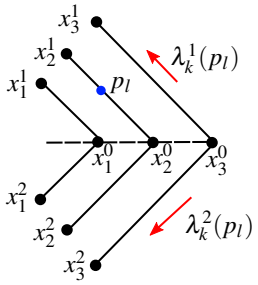

Fig. 1. Mapping of $p_l\in\mathbb{R}^2$ into the homotopy space: multiple mapping is here avoided with $\lambda_k^2=0$ for all k.

straight lines $g^0(\tau)$ and $g^1(\tau)$, colored orange in Fig. 2 with continuous variable $\tau\in[0,1]$ as:

$$g^0(\tau)=x_{LB}^0+(x_{UB}^0-x_{LB}^0)\tau, \tag{31}$$
$$g^1(\tau)=x_{LB}^1+(x_{UB}^1-x_{LB}^1)\tau. \tag{32}$$

Equation (31) is a linear approximation of the trajectory $\hat{x}^0$ between the state x_{UB}^0 and x_{LB}^0, where UB stands for an upper time bound and LB for a lower bound. Eq. (32) linearly approximates the base trajectory $\hat{x}^1$. A homotopy between the two functions for a vertex p_l (and, of course, for C_1) is:

$$g(\tau,\bar{\boldsymbol{\lambda}}(p_l,C_1))=g^0(\tau)+(g^1(\tau)-g^0(\tau))\bar{\boldsymbol{\lambda}}(p_l,C_1). \tag{33}$$

Inserting (31) and (32) into (33) leads to:

$$g(\tau,\bar{\boldsymbol{\lambda}}(p_l,C_1))-x_{LB}^0=(x_{UB}^0-x_{LB}^0)\tau+(x_{LB}^1-x_{LB}^0)\bar{\boldsymbol{\lambda}}(p_l,C_1)$$
$$+[(x_{UB}^1-x_{LB}^1)-(x_{UB}^0-x_{LB}^0)]\tau\cdot\bar{\boldsymbol{\lambda}}(p_l,C_1). \tag{34}$$

For given upper and lower bounds, (34) is a bilinear function in the variables $\bar{\boldsymbol{\lambda}}(p_l,C_1)$ and τ. An iterative procedure for finding $\bar{\boldsymbol{\lambda}}(p_l,C_1)$ in three steps is as follows:

Step (i): The bilinear equation (34) is conically approximated by setting $x_{LB}^1=x_{LB}^0$, leading to the red cone in Fig. 2, and to values of τ and $\bar{\boldsymbol{\lambda}}(p_l,C_1)$.

Step (ii): The upper and lower bounds are tightened in an iterative procedure until p_l is located between the closest upper and lower bound shown by the green cone. The factor of tightening UB and LB in each iteration is given by τ, originating from step (i).

Step (iii): Since p_l would be located in the green cone by a wrong homotopy value of approximately $\bar{\boldsymbol{\lambda}}(p_l,C_1)\approx 1$, a backtracking procedure is carried out, which reduces the upper bound along a linear interpolation between x_{LB}^1 and the green x_{UB}^1, as well x_{LB}^0 and the green x_{UB}^0, until p_l is located on the front side of the blue triangle. This leads to the solution of the blue conic approximation, which is equivalent to the bilinear equation (34), between the resulting upper bounds UB (green) and LB (black).

Based on this procedure, the mapping results in the set:

$$\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}=\{\bar{\boldsymbol{\lambda}}(p_l,C_s)|p_l\in\mathscr{P}_{x,v},C_s\in\mathscr{G}:\bar{\boldsymbol{\lambda}}(p_l,C_s)\geq[0]^{n_c}\} \tag{35}$$

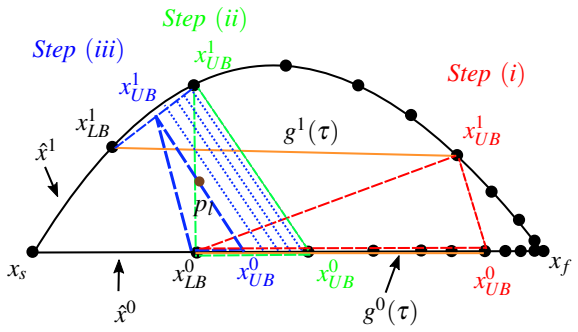

Fig. 2. Procedure for determining $\bar{\boldsymbol{\lambda}}(p_l,C_1)$.

B. Online trajectory determination

The following procedure determines a vector of homotopy parameters which ensures that the interior of the detected obstacle is not crossed. The control laws determined offline in Sec. IV-B are then used to control the states to the corresponding homotopic trajectory.

When the obstacle is detected at time k^*, $\bar{\boldsymbol{\lambda}}$ has to be chosen such that the offline computed controllers K_{k^*+j}, $j \in \mathscr{J}$, drive the system from x_{k^*} to x_f, or respectively from $\boldsymbol{\lambda}_{k^*}$ to $\bar{\boldsymbol{\lambda}} := \bar{\boldsymbol{\lambda}}(p_l, C_s) \in \mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$. This should be achieved such that the state trajectory passes the obstacle at the vertex p_l, which incurs the lowest costs $J(\bar{\boldsymbol{\lambda}})$. 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 (6). With (12) and (13), the transformed costs result to:

$$J(\bar{\boldsymbol{\lambda}}) = \sum_{j=0}^{N-1-k^*} (x_{k^*+j} - x_f)^T Q (x_{k^*+j} - x_f) + (u_{k^*+j} - u_f)^T R (u_{k^*+j} - u_f) \tag{36}$$

$$\text{s.t.} \quad x_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) = x^0_{k^*+j} + D_{x_{k^*+j}} \boldsymbol{\lambda}_{k^*+j} \tag{37}$$

$$\tilde{u}_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) = u_{k^*+j}(\boldsymbol{\lambda}_{k^*+j}) + \delta u_{k^*+j} \tag{38}$$

$$\delta u_{k^*+j} = -K_{k^*+j}(\boldsymbol{\lambda}_{k^*+j} - \bar{\boldsymbol{\lambda}}) \tag{39}$$

$$\delta \boldsymbol{\lambda}_{k^*+1+j} = (I - D^{-1}_{x_{k^*+1+j}} B K_{k^*+j}) \delta \boldsymbol{\lambda}_{k^*+j} \tag{40}$$

$$j \in \mathscr{J}. \tag{41}$$

DEFINITION V.3 Let $\boldsymbol{\Lambda}$ denote the set $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$ ordered with increasing cost $J(\bar{\boldsymbol{\lambda}})$. An element of $\boldsymbol{\Lambda}$ is referred to by $\Lambda(i)$, and $\Lambda(1)$ has lowest costs. △

The procedure is shown in Algorithm 1: Starting from the optimal trajectory with $\boldsymbol{\lambda}_{k^*} = [0]^{n_c}$ at time k^*, the optimal trajectory is checked against collisions with $\mathscr{P}_x$ (line 4). If this trajectory collides with the obstacle, the vertices p_l of $\mathscr{P}_{x,v}$ are mapped into the homotopy space, leading to the set $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$ (line 5), which is then sorted according to its costs in order to obtain $\boldsymbol{\Lambda}$ (line 6). Starting with the trajectory with $\Lambda(1)$ (and then continuing in the order of increasing costs), the trajectory is computed using (3) and (13), see (line 8-10). It is then checked against collisions with $\mathscr{P}_x$ (line 11). Finally, the algorithm terminates directly if the trajectory with desired homotopy value $\Lambda(i)$ is feasible. If this is not the case, i is incremented and a new trajectory passing the obstacle along another vertex (and with higher costs) is computed and checked against collision. The algorithm has to be repeated in the worst case for $|\boldsymbol{\Lambda}|$ times. If no solution can be found, no possible trajectory avoids collision , and the algorithm terminates and the system has to be stopped by an emergency routine.

Algorithm 1 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: • Compute the set of vertices $\mathscr{P}_{\bar{\boldsymbol{\lambda}},v}$
6: • Compute the set $\boldsymbol{\Lambda}$ ordered according to $J(\bar{\boldsymbol{\lambda}})$
7: **for** $i \in \{1, ..., |\boldsymbol{\Lambda}|\}$ **do**
8: • Compute for all $j \in \mathscr{J}$ the states
9: $x_{k+j}(\boldsymbol{\lambda}_{k+j})$ according to (3) and (13)
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: • $\forall j \in \mathscr{J} : x^h_{k+j}$ is not feasible
13: **else**
14: • $\forall j \in \mathscr{J} : x_{k+j}(\boldsymbol{\lambda}_{k+j})$ is an optimized
15: and feasible trajectory with $\bar{\boldsymbol{\lambda}} = \Lambda(i)$.
16: **break**
17: **end if**
18: **end for**
19: **end if**

VI. NUMERICAL EXAMPLE

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 matrices of the stable, fully reachable discrete-time system are given by:

$$A = 1e^{-3} \cdot \begin{bmatrix} 953 & 24 & 12 \\ 24 & 911 & 9.4 \\ 12 & 9.4 & 965 \end{bmatrix}, \; B = 1e^{-4} \cdot \begin{bmatrix} 967 & -7 & 40 \\ -7 & 1026 & -46 \\ 40 & -46 & 1057 \end{bmatrix}, \tag{42}$$

The weighting matrices Q of (4) is chosen as identity matrix, and R as $10 \cdot Q$.

The weighting matrices for the controller design are $Q_C = 50 \cdot Q$, $R_C = R$, chosen to design the controller for reaching the desired state trajectory quickly. The time horizon is selected to $N = 60$, the initialization to $x_0 = x_s = [0,0,0]^T$, and the final state and input at the end time $k = 60$ are $x_N = x_f = [5,5,5]^T$, $u_N = u_f = [0.5, 2.7, 0.7]^T$. With $n_c = 3$, the set of offline computed trajectories is chosen to: $\mathscr{X} = \{(\hat{x}^0, \hat{u}^0), (\hat{x}^1, \hat{u}^1), (\hat{x}^2, \hat{u}^2), (\hat{x}^3, \hat{u}^3)\}$, see Fig. 3. The optimal trajectory is $(\hat{x}^0, \hat{u}^0)$, and is determined by solving (5) without the collision avoidance constraint. In Fig. 3, this trajectory is colored magenta. The other n_c base trajectories (black) of $\mathscr{X}$ are specified to obtain the shown area for obstacle avoidance. The polytopic obstacle $\mathscr{P}_x \subset \mathbb{R}^3$ is chosen as visible in Fig. 3; obviously $\hat{x}^0$ intersects with $\mathscr{P}_x$, while this is not true for the three base trajectories.

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 which appears at $k^* = 10$, and then remains static until the end time.

The homotopy-based approach is compared to a mixed-integer quadratic programming (MIQP) approach for the remaining time horizon of 50 steps to guarantee feasibility of the executed trajectory. In the MIQP approach, the forbidden region $\mathscr{P}_x$ is described by binary variables using the well-known Big-M-formulation to handle the non-convexity. As can be seen in Fig. 3, the system moves along the optimal trajectory (magenta) for both methods up to $k^* = 10$ (green),

when the obstacle is detected. The trajectory determined by the homotopy approach (blue) passes the obstacle along its lower right vertex, while the trajectory obtained by the MIQP method (red) passes the obstacle along its lower right edge to minimize the cost.

While the costs for the red trajectory is lower compared to the blue one, a significant advantage of the homotopy-based method compared to the MIQP optimization can be observed with respect to computation time: The homotopy approach determines a feasible and optimized solution in totally 3.2 milliseconds (ms) using a Matlab implementation on a standard PC. For the MIQP-based approach, a Matlab implementation with embedded solution of the optimization problems by the CPLEX-tool used on the same PC solves the problem in approximately 2.5 seconds (s). 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 also to a large number of algebraic constraints involving these variables. Comparing both methods with the formulation of problem (5) as a nonlinear, non-convex program leads to even higher computation times of 5 s (using the Matlab solver fmincon, and by approximating the obstacle by an ellipsoidal region which must not be entered).

VII. CONCLUSIONS

The paper has shown that the problem of online computation of optimized trajectories satisfying non-convex state constraints (in the sense of obstacle avoidance) can be solved by employing homotopy properties. The key idea is to combine the offline computation of feedback control laws by semi-definite programming with the online determination of suitable reference values for these controllers (in terms of desired homotopy parameters) to circumvent the obstacle.

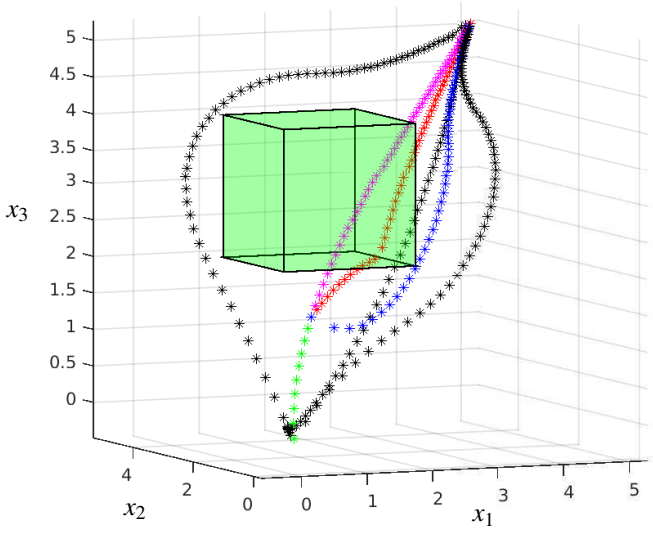

Fig. 3. Simulated state trajectories with optimal trajectory $\hat{x}^0$ (magenta), base trajectories $\hat{x}^1$, $\hat{x}^2$, $\hat{x}^3$ (black), a part of the optimal trajectory (green) up to $k^* = 10$, the trajectory obtained by the homotopy approach $\hat{x}(\bar{\boldsymbol{\lambda}})$ (blue),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

The proposed method computes an approximation of the optimal solution significantly faster compared to methods like MIQP or NLP (as would be used in MPC). For the homotopy approach, the computation times were found to be 3 orders of magnitude smaller, allowing the application to systems with considerably faster dynamics.

Future work will extend the method by passing obstacles not only around its vertices, but across edges. Furthermore, the extension to obstacles moving after detection, and time-varying final states will be considered.

REFERENCES

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

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

[3] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *J. of Process Control*, 12(4):577 – 585, 2002.

[4] H. Ding, G. Reissig, and O. Stursberg. Increasing Efficiency of Optimization-based Path Planning for Robotic Manipulators. In 50^{th} *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. Imura. Least-restrictive move-blocking model predictive control. *Automatica*, 46(7):1234 – 1240, 2010.

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

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

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

[10] 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.

[11] S. Liao. Notes on the homotopy analysis method: Some definitions and theorems. *Comm. in Nonlinear Science and Numerical Simulation*, 14(4):983 – 997, 2009.

[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] 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.

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

[16] P. Tøndel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39(3):489 – 497, 2003.

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

[18] 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.