Model Predictive Control for Real-Time Point-to-Point Trajectory Generation

M. Mahdi Ghazaei Ardakani, *Member, IEEE*, Björn Olofsson, Anders Robertsson, *Senior Member, IEEE*, and Rolf Johansson, *Fellow, IEEE*

***Abstract*— The problem of planning a trajectory for robots starting in an initial state and reaching a final state in a desired interval of time is tackled. We propose an approach based on model predictive control to solve the problem of point-to-point trajectory generation for a given final time. We discuss various choices of models, objective functions, and constraints for generating trajectories to transfer the state of the robot, while respecting physical limitations on the motion as well as fulfilling computational real-time requirements. Extensive simulation results illustrate the use of the approach, and experiments on an industrial robot in a challenging ball-catching task show the effectiveness of the approach also in demanding scenarios with real-time constraints on the computation.**

***Note to Practitioners*—This paper was motivated by the problem of generating movements to transfer a robot from its current state to a new position and velocity at a certain time, when the target state and the final time may require correction at a high rate. For example, for picking small objects from a conveyor belt with a variable feed rate, an off-line planning would fail, since the motion has to be adjusted as soon as the speed is changed. Under the assumption that the desired pickup position and velocity and arrival time can be predicted, the approach in this paper is applicable. The movements can be optimized, for example, for energy efficiency or for reduction of vibrations in the robot. We discuss how to mathematically express the desired performance criteria and other requirements on the motion, such as not violating a maximum joint speed. Quick reactions to sensor inputs are computationally demanding. Thus, we limit ourselves to a class of motion-generation problems that lends itself to numerical optimization. Additionally, we save computation power by gradually refining the motion.**

***Index Terms*— Model predictive control (MPC), real-time control, robotics, trajectory generation, trajectory optimization.**

I. Introduction

TRAJECTORY generation is an inherent problem in motion control for robotic systems, such as industrial manipulators and mobile platforms. The motion-planning problem is to define the path and the course of motion as a function of time, namely the trajectory. In many applications, the trajectory generation is desired to be performed such that the time for executing a task or the energy consumed during the motion is minimized. Hence, motion-planning problems are often formulated as optimal control problems [1]. For solving a trajectory-planning problem, it is, in many cases, beneficial to solve the path-planning problem first and then find a trajectory by assigning time to each point along the path [1], [2]. Nevertheless, for the specification of the problem, such separation can be unnecessary in the optimization framework. Therefore, as long as there is no computational constraint, both path and trajectory planning can be solved in an integrated approach. Ultimately, this leads to a motion planning where the full potential of the system is utilized, since the inherent interrelation between these problems is considered.

In the motion-planning procedure, various modeling assumptions have to be made. The major difference with respect to modeling is whether a kinematic or a dynamic model of the system is considered. Although dynamic models are required for achieving the highest possible performance, a purely kinematic model suffices in many applications, given conservative constraints on the kinematic variables, i.e., the position, the velocity, and the acceleration [3]. Such constraints are then chosen based on the assumed robot configurations and load during the motion, which could imply limitations on the geometry of the workspace or the load variations.

A desired characteristic of motion planning in uncertain environments is the ability to react to sensor inputs [4]. In this paper, we provide experimental results for a ball-catching robot [5], where the estimates of the contact position of the ball are delivered online from a vision system. As more vision data become available, the position and predicted arrival time are estimated with lower uncertainty. Thus, a new trajectory needs to be computed when new estimates are delivered. Our approach to trajectory generation is based on the receding-horizon principle offered by model predictive control (MPC) [6], [7]. By using a final-state constraint in the MPC formulation, it is possible to solve fixed-time motion-planning problems where analytic solutions are not available. In contrast to the typical application of MPC for reference-tracking problems (see [8]–[11] for examples regarding path or trajectory tracking for mobile robots and ground vehicles), we adopt a trajectory-generation perspective.

Manuscript received November 24, 2016; revised March 18, 2017, July 5, 2017, December 6, 2017, and March 20, 2018; accepted November 3, 2018. This paper was recommended for publication by Associate Editor Y. Li and Editor J. Wen upon evaluation of the reviewers' comments. This work was supported in part by the European Union's Seventh Framework Program (FP7/2007–2013) under Grant PRACE 285380 and Grant SMErobotics 287787 and in part by the project Smart System 2015 RIT15-0038 supported by the Swedish Foundation for Strategic Research. *(Corresponding author: M. Mahdi Ghazaei Ardakani.)*

M. M. Ghazaei Ardakani is with the Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT), 16163 Genoa, Italy, and also with the Department of Automatic Control, LTH, Lund University, 221 00 Lund, Sweden (e-mail: mahdi.ghazaei@control.lth.se).

B. Olofsson, A. Robertsson, and R. Johansson are with the Department of Automatic Control, LTH, Lund University, 221 00 Lund, Sweden.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2882764

In our experiments, we use the aforementioned strategy to compute reference values (joint position and velocity) for an underlying motion-control system.

A. Previous Research

An overview of trajectory-generation methods for robots is provided in [4]. There are many methods for online trajectory generation with the objective of time optimality based on analytic expressions, parameterized in the initial and final states of the desired motion and certain constraints [12]–[15]. The main difference between these methods is in their generality with respect to the initial and final states and constraints. Numerical optimization methods based on MPC were also suggested for obtaining time-optimal solutions for point-to-point motion control. In [16], a two-layer approach to time-optimal MPC for linear systems was suggested. The time-optimal point-to-point open-loop control of robot manipulators was translated to a nonlinear two-point boundary value problem and solved iteratively using an indirect method in [17]. Based on the concept of timed elastic bands, [18] proposed an iterative method for time-optimal MPC for point-to-point transitions of nonlinear dynamic systems.

The major benefit of the existing analytic solutions is that nearly time-optimal or time-optimal trajectories can be computed extremely fast. While the minimum-time trajectories are of interest for defining an upper bound for the productivity of a robotic system, they put the system under maximal stress and rely on accurate model and constraints. Hence, the solution to fixed-time problems with a cost function motivated by the application may prove valuable by reducing the wear of the robot. A suboptimal solution to the fixed-time trajectory planning for robot manipulators was proposed in [19]. In another approach, the fixed-time optimal solution in the space of parameterized trajectories was found in [20]. An off-line method for finding energy-efficient trajectories given a path and a maximum time based on nonlinear optimization was suggested in [21].

Trajectory generation based on optimization using a kinematic model has been proposed in [22] for ball-catching robotic systems. Methods for catching flying objects with robots were also considered in [23] and [24]. In the latter, a dynamic model was employed and machine-learning algorithms were used in the online execution. Nonlinear optimization techniques to solve finite-horizon MPC problems with complex dynamical models have been proposed and used for controlling humanoids [25] and for hand manipulation [26].

A preliminary version of the research in this paper with partial results was presented in a conference contribution [27]. Here, we extend the previous results by providing more details on alternative modeling strategies for trajectory generation for flexible robots, the design of the objective function, and possible inclusion of geometric constraints in task space in the optimization. Extensive simulation results are provided to illustrate the developed approaches. We also discuss the differences between the trajectories obtained with our approach as compared to well-known approaches previously proposed in the literature.

B. Contributions

The major contribution of this paper is a method for fast online motion planning, given an initial state of a robot and a desired final state at a given time. The approach does not rely on any predefined trajectory or path. To this purpose, we have adapted the MPC framework to fixed-time point-to-point trajectory generation. In our approach, it is possible to solve the trajectory-planning problem online with application-specific cost functions and a broad class of constraints on inputs and states. The algorithm relies on convex optimization, such that the real-time computational constraints for robot control systems can be satisfied. Various choices of models and constraints are illustrated with simulation results. Further contributions of this paper are a complete implementation of one instance of the approach to motion planning and subsequent experimental evaluations in a challenging scenario of ball catching.

C. Outline

The structure of this paper is as follows. A problem formulation is provided in Section II, where a generic motion-planning problem using MPC is defined. Following this section, the methods that lead to a concrete implementation of a real-time trajectory generator are presented in Section III. Extensive simulation results illustrating different aspects of the approach are presented in Section IV. In Section V-A, the experimental setup is detailed and implementation aspects are discussed. The execution of the proposed motion-planning method on a robot is evaluated and the results for a demanding task of ball catching are provided in Section V. Finally, the results of this paper and the methods are discussed in Section VI, and conclusions are drawn in Section VII.

II. Problem Formulation

In many robotic applications, it is desirable to attain a certain state of the robot at a given time. This will result in a reference-tracking problem if the desired state is specified as a function of time during the entire execution. On the other hand, if there is no specific desired state during certain intervals in time, we need to do planning between the points, i.e., transferring the robot from an initial state to the next state in the given time. In either case, the motion of the robot must fulfill some predefined constraints. Moreover, we wish to specify a desired behavior, which is implicitly characterized by the optimum of an objective function in this paper.

We develop an approach based on model prediction and receding horizon to solve this type of problems. A central notion in MPC is the use of a model to predict the behavior of a system [6], [7]. Accordingly, it is possible to optimize the objective function over a receding horizon, considering the predicted states and outputs. The optimization is usually carried out at each sample. Then, only the first sample in the sequence of computed control inputs over the prediction horizon is applied [6], [7]. Fig. 1 shows a schematic of trajectory planning using MPC.

In the MPC framework, it is required to define a model of the system, an objective function, state and input constraints,

and a prediction horizon. We consider linear models of the form

$$x(k+1) = A_d x(k) + B_d u(k) \tag{1a}$$

$$y(k) = C_d x(k) + D_d u(k) \tag{1b}$$

$$z(k) = \tilde{C} x(k) + \tilde{D} u(k). \tag{1c}$$

Here, $u(k)$, $x(k)$, $y(k)$, and $z(k)$ denote the control signal, the states, the measured outputs, and the controlled variables, respectively [7]. Let us define

$$\mathcal{U} = [u^T(k), u^T(k+1), \ldots, u^T(k+N-1)]^T \tag{2}$$

where N is the prediction horizon. We consider a quadratic cost at time k as

$$V_k(\mathcal{U}) = \sum_{i=k+1}^{k+N} \|z(i) - r(i)\|^2_{Q(i)} + \sum_{i=k}^{k+N-1} \|u(i)\|^2_{R(i)} \tag{3}$$

where $\|a\|^2_W = a^T W a$ for a positive semidefinite weight matrix W and the reference signal is denoted by $r(i)$. Additionally, Q and R are the time-dependent weight matrices. Similar to (2), we define

$$\mathcal{Z} = [z^T(k+1), z^T(k+2), \ldots, z^T(k+N)]^T. \tag{4}$$

Linear constraints on the control and the states are assumed as follows:

$$F\mathcal{U}(k) \leq f \tag{5a}$$

$$G\mathcal{Z}(k) \leq g \tag{5b}$$

where F and G are matrices whose dimensions are determined by the system dynamics (1), the number of constraints, and the prediction horizon.

Given the initial state $x(k)$ of the system, the following optimization problem is solved in the MPC at each sample k [6]:

$$\begin{aligned} &\underset{\mathcal{U}}{\text{minimize}}\ V_k(\mathcal{U}) \\ &\text{subject to (1), (5)} \\ &\quad z(k+N) = r_f(k), \quad \text{(optional)} \end{aligned} \tag{6}$$

where $r_f(k)$ defines for the current sample a constraint on the controlled variable at the end of the prediction horizon.

It is desirable to investigate how the MPC framework is applicable to trajectory generation for point-to-point problems with a desired final time. Furthermore, we wish to find a set of assumptions and methods that allow for real-time solutions.

III. Methods

In this section, we specialize the general linear MPC framework from Section II to point-to-point trajectory generation. We also provide the examples of models and constraints, which later are used in simulations and experiments on a robotic setup.

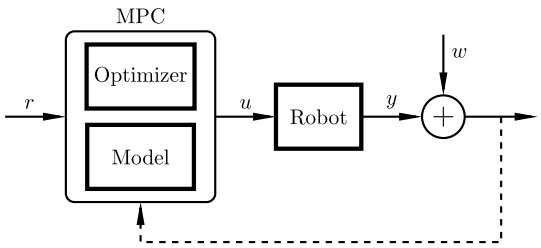

Fig. 1. Schematic of trajectory generation using MPC. r is the reference signal, u is the control signal, y is the output signal, and w is the output disturbance. Dashed line: possible feedback from the measurements to the MPC to update the state of the model. In an open-loop strategy, this feedback is not present.

A. Convexity and Optimality

In general, there might exist several local optima to the underlying optimization problem. The global optimum may not be easy to find, even if the problem is feasible. In the case of multiple local optima, special care must be taken to avoid jumping between different solutions during optimization, which may lead to divergence. On the other hand, at the cost of limiting the scope of the problems, choosing convex cost functions and convex and compact constraint sets, we are assured that if a solution exists, it has to be globally optimal [28].

B. Point-to-Point Planning

In the general objective function (3), the desired controlled states do not need to be specified at every sample. This is an important feature for point-to-point trajectory generation. To clarify this feature, assume that

$$\Psi(k) \subset \{k+1, \ldots, k+N\} \tag{7}$$

denotes the set of indices where there are desired values for the controlled states z. By assuming that $r(i) = 0$ when $i \notin \Psi(k)$, we can rewrite (3) as

$$\begin{aligned} V_k(\mathcal{U}) = &\sum_{i \in \Psi(k)} \|z(i) - r(i)\|^2_{Q(i)} + \sum_{i=k+1}^{k+N} \|z(i)\|^2_{\bar{Q}(i)} \\ &+ \sum_{i=k}^{k+N-1} \|u(i)\|^2_{R(i)} \end{aligned} \tag{8}$$

where $\bar{Q}(i) = Q(i)$ if $i \notin \Psi(k)$ and $\bar{Q}(i) = 0$ if $i \in \Psi(k)$. For reference-tracking problems, the first and the last terms are usually important. In the point-to-point trajectory generation, the first term can be ignored provided that explicit constraints for the desired controlled states are defined. Nevertheless, if soft constraints are preferred, then all of the terms might be used. In this case, the benefit is immediate if the optimization problem with explicit constraints on the controlled variables $z(i)$, $i \in \Psi(k)$, is infeasible.

Since the optimization, in general, runs at every sample instant of the controlled system, it is possible to account for the updates in the target or deviations from the planned trajectories of the robot. However, if the robot motion-control system can assure good trajectory tracking, the feedback loop can also be closed around the model, providing an open-loop control strategy (see Fig. 1). In this case, the robot feedback

controller handles possible deviations from the computed trajectories.

Although it is possible to introduce constraints on the states at any sample instant, to account for the target, we limit ourselves to constraints on $z(k+N)$. This strategy is advantageous if no information about the variations in the target is available ahead of time or if we do not want this information to influence the current decision. Moreover, in such cases, we can utilize the available computational power more efficiently to obtain a solution with a finer resolution as the system follows the computed trajectory toward the target state. This is possible by keeping the number of discretization points constant, while successively reducing the sample period of the dynamic model, i.e., the prediction horizon with respect to the number of discrete time steps remains constant, while the prediction horizon in continuous time gradually decreases. In this case, N is determined by the number of discretization points rather than the length of the trajectory and N is upper bounded by the available computational power. Sufficient number of points in the input signal must, however, be considered to avoid an overconstrained optimization problem.

C. Interpolation of Trajectories

Because of the real-time execution of the motion planning, it is required to have a valid trajectory at every moment. This is also the case when the optimization algorithm is unsuccessful as a result of either an infeasible problem specification or missing the computation deadline. Thus, we consider piecewise trajectories. The previous trajectory is valid until the new one is ready to be switched to. We make sure that the states of the model remain continuous at the switching instants. In addition, it should be ensured that a fail-safe trajectory to bring the robot to stand still from its current state can be computed.

Since MPC is formulated for discrete-time systems, we need to interpolate between the computed trajectory points. The interpolation makes it possible to have a different sample period for the discretization of the dynamics in the optimization during the trajectory generation than for the controlled system. Assuming that T is a vector of increasing time instants and U is a matrix of the corresponding inputs

$$T = [t_1, \quad t_2, \quad \ldots, \quad t_n] \tag{9}$$

$$U = [u(1), \quad u(2), \quad \ldots, \quad u(n)] \tag{10}$$

we define the continuous-time signal u_c using a linear interpolation such that

$$u_c(t) = u(k) + \frac{u(k+1) - u(k)}{t_{k+1} - t_k}(t - t_k), \quad t_k \le t < t_{k+1}. \tag{11}$$

D. Discretization

In the MPC framework, the dynamics of a system is described by difference equations. Assuming a sample period h and the following continuous-time linear model:

$$\begin{aligned} \dot{x}_c(t) &= Ax_c(t) + Bu_c(t) \\ y_c(t) &= Cx_c(t) \end{aligned} \tag{12}$$

the discrete-time system obtained using the predictive first-order-hold sampling method [29] is

$$\begin{aligned} x(k+1) &= \Phi x(k) + \frac{1}{h}\Gamma_1 u(k+1) + \left(\Gamma - \frac{1}{h}\Gamma_1\right) u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{13}$$

where

$$[\Phi \quad \Gamma \quad \Gamma_1] = \begin{bmatrix} I & 0 & 0 \end{bmatrix} \exp\left(\begin{bmatrix} A & B & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{bmatrix} h\right). \tag{14}$$

Note that (13) can be rewritten in the standard form by changing to a new state variable ζ according to

$$\zeta(k+1) = \Phi\zeta(k) + \left(\Gamma + \frac{1}{h}(\Phi - I)\Gamma_1\right) u(k) \tag{15}$$

$$y(k) = C\zeta(k) + \frac{1}{h}C\Gamma_1 u(k). \tag{16}$$

This choice of discretization method is motivated by the linear interpolation of the input trajectories as defined in (11).

E. Models

The models for the trajectory planning are developed to approximate the relation between the actuation and the motion. The motion can, in principle, be specified in any set of generalized coordinates. However, choosing a certain coordinate system can significantly simplify the equations. Also, depending on the application, it is sometimes more natural to use Cartesian coordinates in task space rather than joint coordinates. The motion can be specified in terms of position, velocity, or higher order time derivatives of the position. A control input can be a dynamic quantity such as force or torque. Typically, the dynamics of robots are highly nonlinear. Therefore, without using any form of approximation, this fact limits the usage of a linear MPC framework.

Here, we provide two examples of models for which the linear MPC approach is applicable.

1) Chains of Integrators: Assuming a good tracking performance for a robot, a simple kinematic robot model using only the kinematic variables can be constructed by multiple decoupled chains of integrators, e.g., [1]. Since robots have various structures, we choose the generalized coordinates q for the representation of the states. In this way, q may represent either the joint values or the Cartesian coordinates depending on the application. We choose the state vector as

$$x = [q_1, \dot{q}_1, \ddot{q}_1, \ldots, q_d, \dot{q}_d, \ddot{q}_d]^T$$

where d is the number of degrees of freedom (DoFs).

The continuous-time model (12), where each DoF is assumed to be a triple integrator, is specified by the matrices

$$A = \text{blkdiag}([\tilde{A}, \ldots, \tilde{A}]), \quad B = \text{blkdiag}([\tilde{B}, \ldots, \tilde{B}])$$

and $C = I_{3d}$, where I_{3d} denotes an identity matrix in $\mathbb{R}^{3d\times 3d}$, blkdiag$(\cdot)$ forms a block diagonal matrix from the given list of matrices, $A \in \mathbb{R}^{3d\times 3d}$, $B \in \mathbb{R}^{3d\times d}$, and

$$\tilde{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

The matrices for the discretized model concerning the controlled states can be calculated as follows:

$$\Phi = \text{blkdiag}([\tilde{\Phi}, \ldots, \tilde{\Phi}]),\ \Gamma_1 = \text{blkdiag}([\tilde{\Gamma}_1, \ldots, \tilde{\Gamma}_1])$$
$$\Gamma = \text{blkdiag}([\tilde{\Gamma}, \ldots, \tilde{\Gamma}])$$

where $\Phi \in \mathbb{R}^{3d\times 3d}$, $\Gamma_1, \Gamma \in \mathbb{R}^{3d\times d}$, and

$$\tilde{\Phi} = \begin{bmatrix} 1 & h & \frac{h^2}{2} \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}, \quad \frac{1}{h}\tilde{\Gamma}_1 = \begin{bmatrix} \frac{h^3}{24} \\ \frac{h^2}{6} \\ \frac{h}{2} \end{bmatrix}, \quad \tilde{\Gamma} = \begin{bmatrix} \frac{h^3}{6} \\ \frac{h^2}{2} \\ h \end{bmatrix}.$$

2) Linearized Model: As another example, let us assume that we have identified a linear model from the joint reference velocities to the joint velocities around a certain working point in the task space of the robot. As long as the range of the movements is small, this model can be employed to the purpose of trajectory generation. Compared to the model with pure chains of integrators considered in Section III-E1, such a model could provide a more realistic approximation of the robot dynamics. In particular, the benefits are immediate when the mechanism has certain resonant frequencies, resulting from inherent flexibilities in the mechanical parts or coupling between the DoFs. Assume that for each pair of an input and a DoF, we have identified a linear model such that

$$v_i = H_{i,j} v_j^r \tag{17}$$

where v^r and v denote the velocity reference and the velocity along the DoF, respectively, and $H_{i,j}$ is the transfer function from the jth velocity reference to the ith velocity. We can construct a complete model from the reference acceleration a^r to actual positions and velocities such that

$$x = \tilde{H} a^r \tag{18}$$

where $x = \left[q_1, \dot{q}_1, \ldots, q_d, \dot{q}_d\right]^T$, $a^r = \left[a_1^r, a_2^r, \ldots, a_d^r\right]^T$, and

$$\tilde{H} = \begin{pmatrix} \frac{1}{s^2}H_{1,1} & \frac{1}{s^2}H_{1,2} & \cdots & \frac{1}{s^2}H_{1,d} \\ \frac{1}{s}H_{1,1} & \frac{1}{s}H_{1,2} & \cdots & \frac{1}{s}H_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{s^2}H_{d,1} & \frac{1}{s^2}H_{d,2} & \cdots & \frac{1}{s^2}H_{d,d} \\ \frac{1}{s}H_{d,1} & \frac{1}{s}H_{d,2} & \cdots & \frac{1}{s}H_{d,d} \end{pmatrix}. \tag{19}$$

Finally, we convert $\tilde{H}$ to a state-space model and compute its discrete-time counterpart according to Section III-D.

Ideally, several linearization points along the trajectory could be used to obtain a solution with high accuracy for the complete motion in the case of nonlinear robot models. However, in point-to-point movements, only the initial and target states are known *a priori*, and other straightforward choices for linearization points are not available if a single convex optimization problem is opted for. Nevertheless, considering a limited geometric workspace, we can assume that the changes in the robot dynamics are limited. This assumption allows using a single linearization point. If the linearization point is chosen at the target state, higher model accuracy close to the target point is achieved, and thus a more accurate trajectory close to the target, where many applications imply strict requirements on the motion.

F. Cost Function

When choosing a cost function in the optimization, physical interpretations are often beneficial, but, in general, the cost does not need to have a physical meaning. For trajectory generation, we can consider punishing high values of the acceleration or the velocity. We can also choose the cost $V_k(\mathcal{U})$ to model, for instance, the distance traveled by a robot or the mechanical energy. Let ΔX denote the changes in the vector of positions, then the length is approximately proportional to $\sum\|\Delta X\|$. Also, introducing a cost on the form $\sum \dot{X}^2$ can lead to keeping the kinetic energy low. The MPC problem for point-to-point trajectory generation can be formulated by assuming the objective function (8) with $\Psi(k) = \emptyset$ and $\bar{Q}(k+N) = 0$, i.e., ignoring the first term and the cost on the final state.

In addition to the basic formulation outlined in the previous paragraph, other features can be built into the design of the cost function. Considering that an abrupt brake of the robot motion often leads to the excitation of vibration modes, we can shape the control signal such that it smoothly approaches zero toward the end of the motion. For example, an exponential weighting function for the control input can be applied from sample k' onward according to

$$R(i) = \begin{cases} c^{\left(\frac{i-k'+1}{N+k-k'}\right)} R, & \text{if } k' \le i < N+k \\ R, & \text{if } k \le i < k' \end{cases} \tag{20}$$

where $c \ge 1$ denotes the maximum weight at the end of the prediction horizon and R is a constant matrix. Similarly, the deviation of the states from r_f can be penalized more aggressively close to the end of the trajectory to guarantee that the system smoothly approaches the desired final state.

Another possibility is to filter out undesired frequencies by introducing a cost term in the frequency domain. This requires expressing the time signals in the frequency domain using the discrete Fourier transform (DFT) [30]. Let K denote the DFT kernel for a given length and φ be a discrete-time filter specifying the penalties on different frequency components. Then, a cost in the frequency domain for signal a can be formulated as $\|a\|_W^2$. The weight matrix is defined as

$$W = \text{real}(K^* \text{diag}(\varphi^* \varphi) K) \tag{21}$$

where * denotes the complex conjugate transpose and $\text{diag}(\cdot)$ denotes the operator that sets the nondiagonal elements of a given matrix to zero.

G. Constraints

In addition to the constraints related to the final state or the states of the via points, constraints on the physical variables can be included. Bounds on the joint velocity, joint angles, and control signal can be expressed as state or control constraints directly. It is also straightforward to include constraints on linear combinations of the kinematic variables. From a practical perspective, we may consider limiting the coordinate positions,

the velocities, the accelerations, and the jerks by constraining states and inputs. Let us define

$$z_j^{\max} = \left[q_j^{\max}, \quad v_j^{\max}, \quad a_j^{\max}\right], \quad 1 \le j \le d. \tag{22}$$

Therefore, the matrices F and G and the vectors f and g in (5) can be formed such that

$$\begin{aligned} |u(i)| &\le \left[u_1^{\max}, \quad \ldots, \quad u_d^{\max}\right]^T, \quad k \le i \le k+N-1 \\ |z(i)| &\le \left[z_1^{\max}, \quad \ldots, \quad z_d^{\max}\right]^T, \quad k+1 \le i \le k+N \\ z(k+N) &= r_f(k) \end{aligned}$$

where $r_f(k)$ is, for the current time step, the desired final state according to (6). Note that if the desired target state is a constraint on the final state, we can successively reduce the sample period, as described in the last paragraph in Section III-B.

Constraints related to each joint of the robot, such as maximum angles or angular velocities, are naturally expressed in the joint space. On the other hand, constraints on the motion of the end-effector are more conveniently expressed in the Cartesian space. Volumes for the desired work space and obstacles in Cartesian space can be considered. More specifically, a typical work space can be expressed as a set of linear inequalities

$$G\begin{bmatrix} X(k) \\ 1 \end{bmatrix} \le 0 \tag{23}$$

where X is a column vector containing a subset of the Cartesian coordinates and G is a matrix of the proper dimension. By including velocities in X, it is also possible to define velocity-dependent boundaries. Ideally, both joint-space and Cartesian-space constraints could exist and the relation between the variables could be established by forward or inverse kinematics. However, because of the properties of the forward and inverse kinematics, this might lead to difficult nonlinear equations, or nonconvex constraint sets, in one or the other variable set.

The full power of the linear MPC approach for planning a trajectory is utilized when task-space and joint-space limits can be approximated by convex sets. Certain constraints in task space could be approximated by a convex set in joint space and *vice versa*. Here, we give an example for approximating certain geometrical constraints in task space as a convex set in joint space. However, such an approximation implies that we limit the solution to one of the several possible configurations for robots that are redundant with respect to the task.

In the illustration of the method, we consider the kinematics of an ABB IRB140 robot [31]. The end-effector is required to move in a region defined by a thin volume located in front of the robot. We fix joints 4, 5, and 6 to 0°, 40°, and 0°, respectively, and find the range of values for the first three joints of the robot that projects the position of the end-effector to the desired volume in the Cartesian space. In Fig. 2, an ellipsoid is fitted to the allowed points in the joint space. In general, the convex hull [28] or the largest convex subset of the acceptable points can be computed.

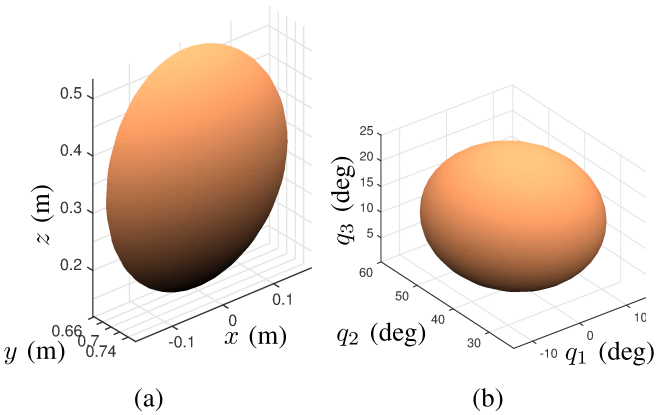

Fig. 2. Example of approximating a geometrical constraint in Cartesian space by a convex constraint in joint space (a) a thin sphere cap in task space and (b) its approximation by an ellipsoid in joint space.

IV. Simulation Results

Simulations were performed to validate the method presented in Section III and investigate its characteristic features for different choices of the cost function and the constraints, combined with the considered modeling approaches. In this section, we present the simulation results concerning the point-to-point trajectory generation, where a linearized local dynamic model of the robot is employed and geometrical constraints are enforced on the robot tool position. For further simulation results in the case of a fixed single target point, employing a model based on decoupled chains of integrators, the reader is referred to [27].

A. Dynamic Model and Geometrical Constraints

Simulations of trajectory generation in joint space were performed, where the approach defined in Section III-E with a linearized model describing the local flexible dynamics of a robot with three joints was employed. We also investigated the effect of geometric constraints in task space, assuming the kinematics of a robot manipulator of type ABB IRB140 [31]. The constraints were enforced in the trajectory generation using the approach outlined in Section III-G, with a convex approximation of the allowed area in the task space (see Fig. 2). Considering that the main objectives of these simulations are to visualize the effects of different constraints and model choices, the trajectories were computed over the complete time horizon directly. This approach does not imply any limitations in the evaluation, since the methods could also be employed with the MPC approach to trajectory generation in a receding-horizon fashion at each sample instant.

The coupling dynamics—the transfer functions $H_{i,j}$ in (19)—were chosen on the form

$$K\frac{\omega_0^2}{s^2 + 2\zeta_0\omega_0 s + \omega_0^2} \tag{24}$$

where K is a coupling factor (which is equal to one for the diagonal terms $H_{i,i}$), ω_0 is the natural frequency, and ζ_0 is the damping. In the simulations, the model parameters were heuristically chosen to resemble the dynamic behavior of an industrial manipulator with flexible modes. The parameters of

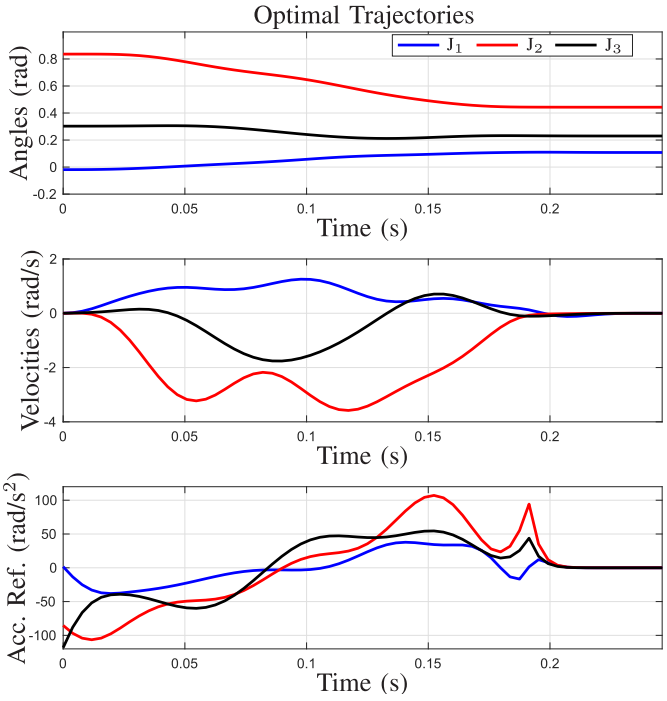

Fig. 3. Optimal trajectories obtained with the linearized model describing the local dynamics of a flexible 3-DoF robot in an area close to the desired final point in the generated trajectories. The joints are denoted by J_i, $i \in \{1, 2, 3\}$. Compared with Fig. 4, the same point-to-point motion-planning task was considered but with an oversimplified model.

TABLE I

MODEL PARAMETERS IN THE EXAMPLE WITH A ROBOT COMPRISING FLEXIBLE MODES, SEE (24)

	K	$w_0/2\pi$	ζ_0
$H_{1,1}$	1	25	0.4
$H_{2,2}$	1	15	0.1
$H_{3,3}$	1	10	0.09
$H_{1,2}$	-0.3	15	0.2
$H_{1,3}$	-0.3	15	0.2
$H_{2,3}$	-0.48	15	0.2

the model are detailed in Table I. The coupling dynamics between the different actuation directions were introduced symmetrically for physical reasons.

A shaping function with exponential growth toward the end of the motion was introduced in the cost function, see (20). This function acted on the joint angles, the joint velocities, and the control inputs in the optimization. The intention with such a choice is to obtain trajectories that smoothly approach the desired final state.

Fig. 3 shows the trajectories obtained with the linearized model of the robot dynamics in a point-to-point movement. The geometrical constraints were also enforced in the trajectory generation. To highlight the effects of the modeling assumptions and the choice of cost function, no other constraints on the model variables were used. As seen in Fig. 3, the effect of the oscillatory dynamics in the model is reflected in the required input signal and the associated trajectories.

B. Comparison of Models

In order to highlight some of the differences between using the model based on the chains of integrators and the more

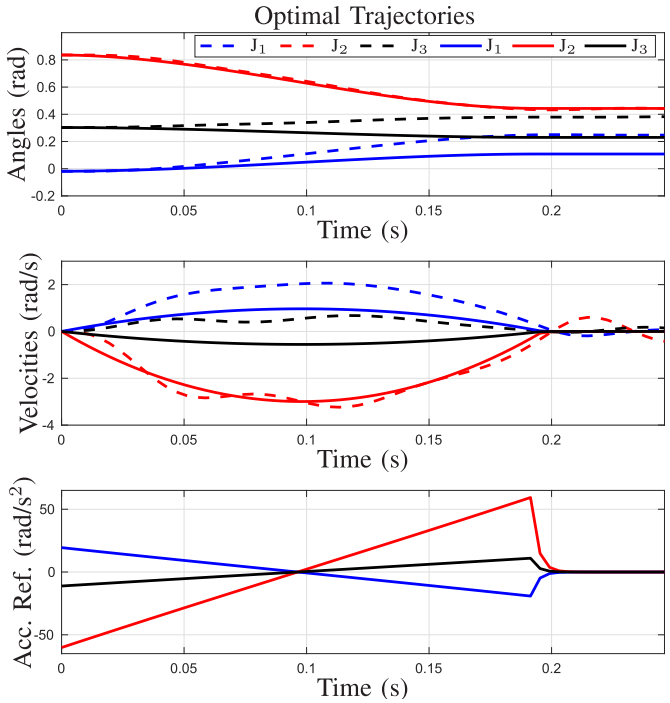

Fig. 4. Comparison of the optimal trajectories obtained with the double-integrator model of the 3-DoF robot and the corresponding trajectories obtained when the same inputs are applied to the linearized model. The trajectories computed with the model based on decoupled double integrators are shown (solid line) and the simulated trajectories when the same inputs are applied to the linearized model are shown (dashed line). The joints are denoted by J_i, $i \in \{1, 2, 3\}$. Note that the target point is not achieved with the oversimplified model in an open-loop strategy.

complex model comprising oscillatory dynamics and coupling between the DoFs, we performed another simulation. In that simulation, we applied the input trajectories (i.e., the acceleration references) computed in a trajectory generation with the simpler model with the chains of integrators to the more complex linearized model. The resulting trajectories and those computed based on a purely kinematic model with decoupled double-integrators are compared in Fig. 4. There is a clear difference between the trajectories in the two cases, which indicates that there is a possible major benefit of using the more complex model when such a model is available.

Comparing the trajectories, it can also be noted that using the inputs computed based on the double-integrator model on a system described by the linearized model does not guarantee that the final position constraints are fulfilled. This is clear from the simulation results in Fig. 4. This effect, though, will be suppressed if feedback controllers for the joint positions and velocities are used, which is standard in conventional industrial robot control systems. However, the price paid for obtaining a good tracking of nonaccurate reference trajectories employing feedback controllers is generally the need for a high control authority and the increase in stiffness of the robot even in applications where this is not desired.

C. Effect of Geometrical Constraints

In order to illustrate the effect of geometrical constraints, the simulation was performed both with and without the constraints and the results are shown in Fig. 5. In the figure, the part where one of the paths violates the geometric

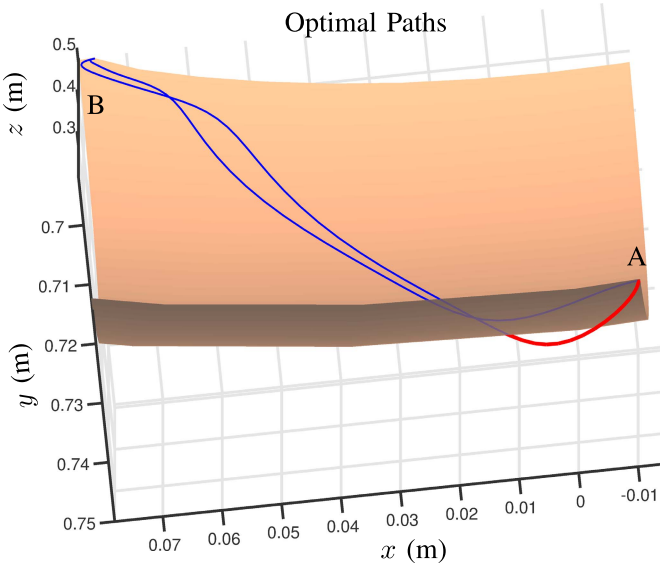

Fig. 5. Comparison of the paths from point A to point B obtained with and without the geometrical constraint corresponding to an allowed area in the task space in a point-to-point movement. The surface shows the boundary of the geometrical constraint, and the part of the path that violates the constraint is shown in red.

constraint is indicated in red and the other path, which resulted from the trajectory generation with the additional geometrical constraint, is within the allowed area. It is thus clear that such geometrical constraints could be handled efficiently in the trajectory generation, if a suitable convex description of the area of interest could be found.

V. Experiments and Results

For evaluating the performance of the trajectory-generation method in a challenging scenario, we considered the task of catching balls with the robot. For additional experimental results for a sequence of target points, the reader is referred to [27]. The method based on a decoupled kinematic robot model using the chains of integrators as described in Section III-E was implemented and experimentally evaluated on an industrial robot system. The trajectory generation was performed in joint space assuming that the DoFs can be independently controlled with good tracking performance. The prediction horizon was set equal to the remaining time to the desired final state using 20 discrete samples. The open-loop strategy to trajectory generation was employed, i.e., no feedback from the robot measurements was used.

A. Experimental Setup

The robot system consisted of an ABB IRB140 industrial manipulator [31] combined with an ABB IRC5 control cabinet. The control system was further equipped with a research interface, ExtCtrl [32], in order to implement the developed trajectory-generation method as an external controller. The research interface permits low-level access to the joint position and velocity controllers in the control cabinet at a sample period of 4 ms. More specifically, reference values for the joint positions and velocities can be specified and sent to the joint controllers, while the measured joint positions and velocities and the corresponding reference joint torques were sent back to the external controller from the main control cabinet with the same sample period. This sample period defined the real-time requirement and the rate at which the MPC was run, while the sample period of the discretized kinematic model was chosen independently since an interpolation technique as described in Section III-C was used. The implementation of the trajectory generation was made in the programing language Java, and the communication with the external robot controller was handled using the LabComm protocol [33].

The inverse kinematics of the robot were required in order to transform the desired target points in Cartesian space to joint-space coordinates that can be used in the motion planning. The kinematics required were available in Java from a previous implementation [5]. For implementation of the solution of the MPC optimization problem, the CVXGEN code generator [28], [34] was used. The generator produced C code, which was subsequently integrated with the Java program using the Java Native interface. The generated C code was by default optimized for performance in terms of time complexity. It enabled solution of the required quadratic program in the MPC within 0.5 ms, i.e., each optimization cycle, including overhead, required approximately 1–4 ms (for the four DoFs used in the considered task and a prediction horizon of 20 samples) on a standard personal computer with an Intel i7 processor with 4 cores. In order to preserve the numerical robustness of the solver, scaling of the equations, the state constraints, and the inputs was introduced in the optimization. This is important, considering the fact that the matrices can be poorly conditioned in the case of short sample periods when close to the final time.

We employed the computer-vision algorithms and infrastructure developed in [5] and [23] for detection of the balls thrown and prediction of the target point and time. Two cameras detected the ball thrown toward the robot, and the image-analysis algorithm estimated the position and velocity, which provided the basis for the model-based prediction of the target state. A photograph of the experimental setup is shown in Fig. 6. A movie showing the ball-catching experiment is available online as part of the prior conference publication [27].

B. Ball-Catching Experiments

For each DoF, we employed the constant weighting matrices $Q = \text{blkdiag}([0, 1, 1])$ and $R = 0.001$. This choice means that we penalize a quadratic function of the velocity, the acceleration, and the jerk according to (8). The constraints in the optimization were chosen based on the physical properties of the joints [31], with some margin, according to $q_{\max} = 2$ rad, $v_{\max} = \pi$ rad/s, $a_{\max} = 45$ rad/s^2, and $u_{\max} = 1500$ rad/s^3 for all joints. In contrast to the jerk, which is the input to the adopted model, we used the computed joint position and velocity values as the reference inputs to the low-level joint controllers of the robot system.

The time period from the instant that the ball was thrown until it reached the robot was distributed in the interval

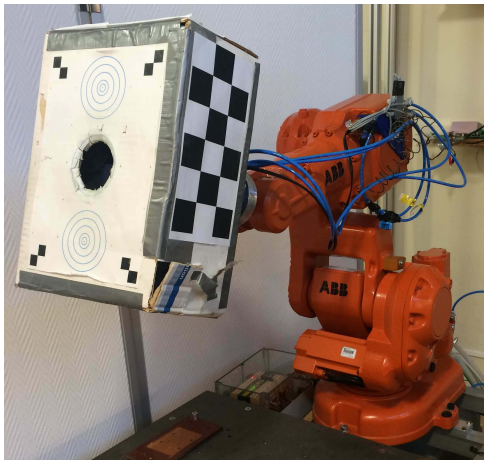

Fig. 6. Experimental setup used for evaluation. The task is to catch a ball in the box attached to the end-effector; two cameras (not visible in the figure) were used for the detection of the ball (see [5] for details regarding the vision system).

[200, 800] ms. As soon as new data from the vision sensors and image-analysis algorithms were available, the estimated contact point and the corresponding arrival time of the ball were updated. These estimates were delivered at an approximate sample period of 4 ms and initiated the computation of new optimal trajectories with a possibly updated target state. Hence, the success rate of the task depended on the satisfaction of the real-time constraints for computing a trajectory for transferring the robot from the home position to the desired final state at the predicted arrival time.

Since the exact orientation of the box was not important for the success of the ball-catching task, not all DoFs of the robot were controlled; specifically, the first three joints were used to position the box and joint 5 to tilt it, while joints 4 and 6 were set to fixed values during the experiments. The trajectories were generated such that the robot should be at the target point with zero velocity and acceleration at a given time. Given a conservative acceleration limit and the velocity constraint, we computed analytically a minimum time for reaching the predicted target point. The fixed final time in the trajectory generation was then always chosen as the maximum of the computed minimum time and the latest estimated arrival time of the ball with a predefined margin. This improved the ball-catching performance, since the initial target-point estimates exhibited large uncertainty. By moving toward the target point, there was a higher chance of catching the ball later on when more accurate estimates were obtained. Once the robot reached the target state and paused there for 100 ms, a new trajectory for returning to the home position in 1 s was computed and executed. In the case that the estimated arrival time was already passed, no optimal trajectory was computed.

Several experiments were performed with balls thrown with a large variation of initial velocities from different positions. The results with regard to trajectory generation from one representative experiment are shown in Fig. 7 for the joints that were active in the robot motion. It is clear that the robot tracks the position references (computed by the trajectory generator) closely. In the bottom-left plot in the figure, the time instants at

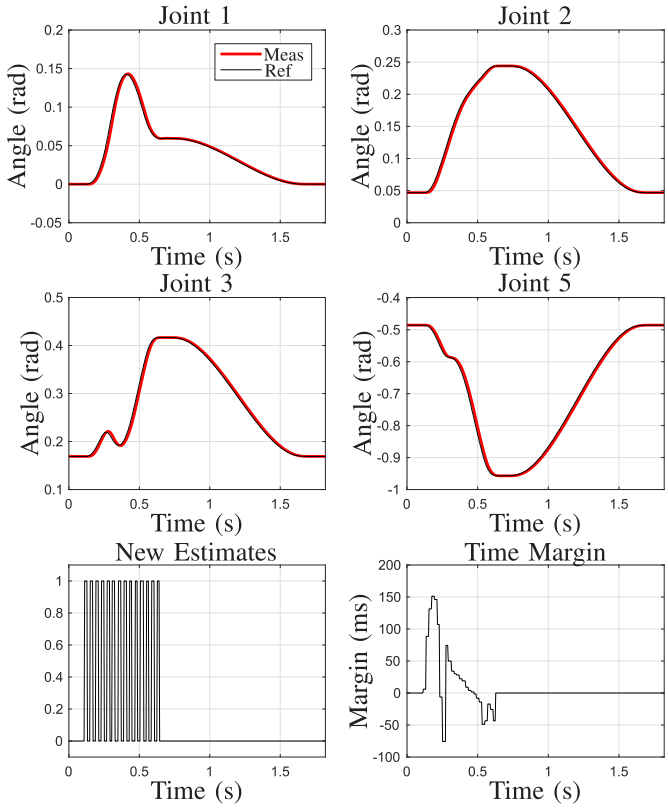

Fig. 7. Results from a ball-catching experiment for one throw. As the ball moved toward the robot, new estimates of the contact position and the arrival time were obtained, and thus the trajectory was recomputed. A new estimate was obtained on the edges of the signal shown in the bottom-left plot. The bottom-right plot shows the margin of the current estimate of the arrival time with respect to the minimum time required to arrive at the target (see the definition in the text).

which new sensor data arrived are also indicated. The multiple acceleration and deceleration phases in the trajectory, which are visible in Fig. 7, are because of online replanning with new target states. The figure also shows the time margin, i.e., the difference between the estimated arrival time and the earliest possible time for reaching the ball, given the constraints. Note that the time margin is an estimate based on the latest available vision-based prediction and can, in certain time intervals, become negative. However, this does not necessarily imply that the ball cannot be caught, since the new updates of the estimated arrival time and point are obtained when available. Additionally, whenever the distance between the robot and the ball is within the tolerance for catching, the time margin is not relevant. In any case, the robot makes an effort to reach the estimated target in the computed minimum time, given the constraints, even if the target is not deemed reachable in the current estimated arrival time.

In order to verify that the real-time requirement of the trajectory-generation method was satisfied in this task, the computation times for all four DoFs were measured for 100 cycles of optimization during a sequence of ball throwing. The histogram of the computation times is shown in Fig. 8. It can be observed that all the computation times except one are within the sample period of the robot at 4 ms with the average and standard deviation being 2.74 ± 0.26 ms. As stated in Section III-C, if the real-time deadline is missed, the robot

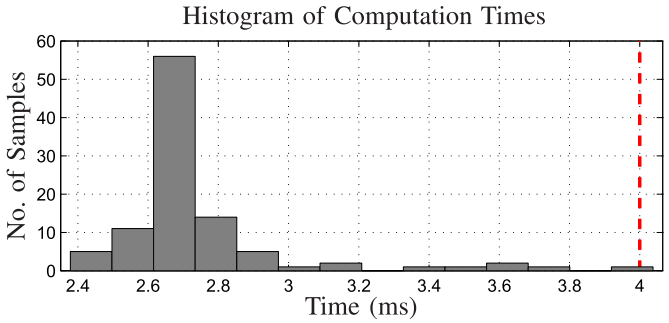

Fig. 8. Histogram of computation times for the trajectory generation during ball-catching experiments based on 100 different executions (four joints were employed). Vertical red dashed line: sample period of the robot system.

continues to move using the latest computed trajectory. In such a case, it is also important to terminate the current optimization as soon as the deadline has been missed.

VI. Discussion

We have proposed an approach based on MPC for fixed-time point-to-point online trajectory generation with real-time computational constraints for robots. The main characteristic of this method is that it allows generation of trajectories that are optimal with respect to a quadratic cost function, while satisfying a final-time constraint as well as linear constraints on the input and state variables. Possible extensions with respect to the cost function and convex constraints were discussed and evaluated in simulations (see Section IV). The MPC solution considered is based on linear systems and convex optimization. This implies that dynamical variables of many manipulators, such as forces and torques, cannot be directly used in the optimization, neither as part of the cost function nor the constraints. However, in many applications, no severe limitation is introduced by this fact, unless extremely high performance, i.e., working close to the physical limits, is demanded. On the other hand, this restriction allows to obtain a convex problem, which does not suffer from local optima and can be solved efficiently. This aspect was illustrated in the robot experiments in Section V, using decoupled chains of integrators as defined in Section III-E and linear constraints on the kinematic variables. In addition, a more complex linear model, comprising oscillatory dynamics, was considered and evaluated in simulations in Section IV. Another interesting feature for trajectory generation that was investigated in simulation was to enforce explicit position constraints in task space by corresponding approximate constraints in joint space.

In contrast to [13]–[15], our method gives the solution of the fixed-time problem and adds more freedom in the formulation of the motion-planning problem. Note that there is an important difference between our method and methods based on a fixed trajectory profile [35], [36], such as the fifth-order polynomial, or time scaling [37] of minimum-time solutions. With neither of these other methods, it is possible to guarantee that the solution not only fulfills the state, the input, and the final-time constraints but also results in the lowest possible value of an objective function that is a function of the states and the control signals. To highlight the characteristics of the solution computed with the developed trajectory generator, the results for a single target point were compared to the solutions obtained by minimizing the integral of the squared jerk (blue dashed curves) and by scaling the minimum-time solution (green dashed-dotted curves) in Fig. 9. These trajectories were computed based on the same model with a chain of integrators. Starting at rest at $q = 0$ rad, the desired state to reach was $q = 1$ rad at time $t = 1$ s with velocity $v = 0.5$ rad/s and acceleration $a = 0$ rad/s^2. The constraints $v_{\max} = 1.2$ rad/s and $u_{\max} = 250$ rad/s^3 were imposed whenever possible.

The first approach considered in the comparison was to minimize the integral of the squared jerk, while fulfilling the initial conditions and the end constraints at the final time t_f. Consequently, the objective in continuous time was

$$\underset{u}{\text{minimize}} \int_0^{t_f} u(t)^2 \mathrm{d}t. \tag{25}$$

In this case, the solutions for the kinematic variables can be calculated analytically and turn out to be polynomials, where the polynomial describing the position is of the fifth order [38]. With the standard minimum-jerk approach, constraints on the kinematic variables—position, velocity, acceleration, and jerk—are not accounted for in the trajectory generation. This is also clear when examining the computed velocity in Fig. 9 (see the blue dashed line in the second top plot). The second alternative approach was to compute the time-optimal solution, given the initial conditions, the final constraints, and the same constraints on the kinematic variables as in the MPC approach, with the objective

$$\underset{u}{\text{minimize}}\ t_f. \tag{26}$$

The optimization problem was solved numerically using the JModelica.org platform [39]. The solution was subsequently scaled in time [37] with a factor $\gamma \leq 1$ in order to reach the final state at the desired time t_f. The solution is visualized in green in Fig. 9, where the scaling factor $\gamma = 0.93$ has been applied. By definition, the scaling factor is equal to or less than one, otherwise the original trajectory-generation problem to be solved would not be feasible. It should also be noted that the minimum and maximum constraints on the kinematic variables are still satisfied, even though the state constraints are not active during the motion as a result of the scaling. After the scaling process, however, the nonzero final constraint on the velocity is not satisfied. In Fig. 9, the velocity for the scaled trajectory is slightly lower than the desired, and hence does not fulfill the terminal constraint $v(t_f) = 0.5$ rad/s.

Another characteristic of our method is that it improves the accuracy of the optimal trajectory as the system approaches the desired final state by increasing the time resolution. The trajectory-generation problem with a longer sample period is equivalent to the corresponding problem with a shorter sample period if equality constraints for the input signal are added at certain samples. These constraints imply a higher resulting cost in the optimization function than for the more relaxed problem where more freedom is available in the input signal. Hence, despite a small number of discretization points in the trajectory generation, the trajectory is successively being refined as the remaining time shrinks. This approach decreases

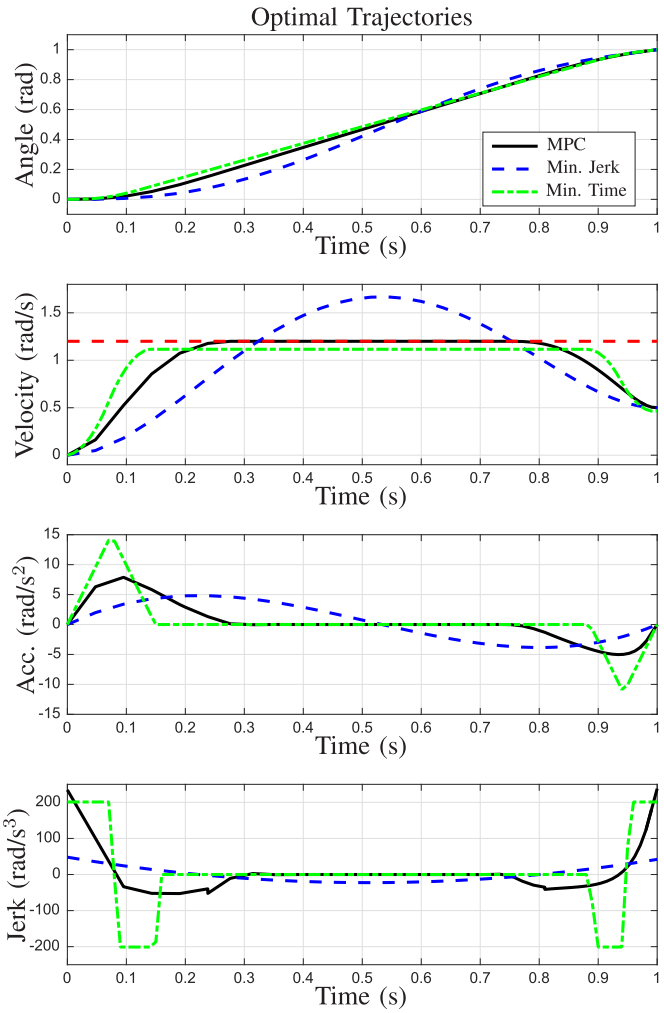

Fig. 9. Comparison of the trajectories computed with the MPC-based optimization approach (black solid line) with the corresponding trajectories obtained by minimizing the jerk (blue dashed line) and by scaling the minimum-time solution (green dashed-dotted line).

the computational burden in the optimization and thus enables computation of optimal trajectories with real-time constraints.

In the future research, the implementation of the approach to trajectory generation in this paper could be extended. For example, the computation time with a more efficient implementation of the algorithms would enable even faster trajectory generation. Since the changes between two consecutive samples can be very small, the time for convergence of the optimization problem can significantly be decreased by warm starting with the previous solution. In the cases where each DoF is assumed to be independently controlled, such as for the model with decoupled chains of integrators discussed in Section III-E, an efficient way to reduce the time complexity is to distribute the computation for each DoF on a separate core of the CPU. Such an approach will allow scaling of the described algorithm to a high number of DoFs, since the major part of the computation time is spent on solving convex optimization problems. Reducing the computation time also allows an increased resolution of the discretization grid, if prompted by the accuracy requirements of a task. In the case of redundancy of the robot with respect to a specific task, such a distributed execution would also allow parallel solutions for different configurations of the robot and subsequent selection of the best possible among the computed solutions, without increasing the computation time.

Further experimental research will be beneficial for investigating the potential benefits of the model with dynamics for capturing flexible modes of the robot (compared to a kinematic model of the robot), minimum-length trajectories for coordination between various DoFs, and costs expressed in the frequency domain. Such an experimental evaluation could be performed in another scenario than the ball-catching considered in this paper, for instance in the scenario of picking objects from a conveyor belt where the pick-up time and position are provided online by an external vision system.

VII. Conclusion

Model predictive control can offer a framework for generating trajectories, which goes beyond tracking problems. A subset of MPC problems—e.g., with quadratic cost functions and linear constraints—has the potential of being efficiently implemented. This fact allowed us to make a real-time implementation of fixed-time point-to-point trajectory planning for an industrial robot with the feature of successive refinement of the planned trajectory. In extensive experiments, the method was evaluated in the demanding and time-critical task of ball catching with online updates of the estimated target state from a vision system as the ball approached the robot. Several additional convex constraints and adaptations of the cost function that can be of interest in the trajectory generation using MPC were investigated and evaluated in simulations.

Acknowledgment

The authors are members of the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[2] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[3] M. M. G. Ardakani, "On trajectory generation for robots," Ph.D. dissertation, Dept. Automat. Control, Lund Univ., Lund, Sweden, Nov. 2016, Thesis No. TFRT–1116–SE.

[4] T. Kröger, *On-Line Trajectory Generation in Robotic Systems* (Springer Tracts in Advanced Robotics), vol. 58, 1st ed. Berlin, Germany: Springer, 2010.

[5] M. Linderoth, "On robotic work-space sensing and control," Ph.D. dissertation, Dept. Automat. Control, Lund Univ., Lund, Sweden, 2013, Thesis No. TFRT–1098–SE.

[6] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[7] J. Maciejowski, *Predictive Control With Constraints*. Boston, MA, USA: Addison-Wesley, 1999.

[8] T. M. Howard, C. J. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Field and Service Robotics*. Cham, Switzerland: Springer, 2010, pp. 69–78.

[9] K. Kanjanawanishkul and A. Zell, "Path following for an omnidirectional mobile robot based on model predictive control," in *Proc. IEEE Int. Conf. Robotics Autom. (ICRA)*, Kobe, Japan, May 2009, pp. 3341–3346.

[10] G. Klančar and I. Škrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robot. Auton. Syst.*, vol. 55, no. 6, pp. 460–469, 2007.

[11] C. Norén, "Path planning for autonomous heavy duty vehicles using nonlinear model predictive control," M.S. thesis, Dept. Elect. Eng., Linköping Univ., Linköping, Sweden, 2013, Thesis No. LiTH-ISY-EX–13/4707–SE.

[12] R. H. Castain and R. P. Paul, "An on-line dynamic trajectory generator," *Int. J. Robot. Res.*, vol. 3, no. 1, pp. 68–72, 1984.

[13] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Trans. Robotics Autom.*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[14] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nice, France, Sep. 2008, pp. 3248–3253.

[15] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Trans. Robotics*, vol. 26, no. 1, pp. 94–111, Feb. 2010.

[16] L. Van den Broeck, M. Diehl, and J. Swevers, "Model predictive control for time-optimal point-to-point motion control," in *Proc. 18th IFAC World Congr.*, Milan, Italy, Aug./Sep. 2011, pp. 2458–2463.

[17] M. H. Ghasemi, N. Kashiri, and M. Dardel, "Time-optimal trajectory planning of robot manipulators in point-to-point motion using an indirect method," *Proc. Inst. Mech. Eng. C, J. Mech. Eng. Sci.*, vol. 226, no. 2, pp. 473–484, 2012.

[18] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Linz, Austria, Jul. 2015, pp. 3352–3357.

[19] I. Duleba, "Minimum cost, fixed time trajectory planning in robot manipulators. A suboptimal solution," *Robotica*, vol. 15, no. 5, pp. 555–562, 1997.

[20] J. Yang, M. Dong, and Y. Tang, "A real-time optimized trajectory planning for a fixed wing UAV," in *Advances in Mechanical and Electronic Engineering*. Berlin, Germany: Springer, 2012, pp. 277–282.

[21] S. Riazi, K. Bengtsson, O. Wigström, E. Vidarsson, and B. Lennartson, "Energy optimization of multi-robot systems," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Gothenburg, Sweden, Aug. 2015, pp. 1345–1350.

[22] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Taipei, Taiwan, Oct. 2010, pp. 2592–2599.

[23] M. Linderoth, A. Robertsson, K. Åström, and R. Johansson, "Object tracking with measurements from single or multiple cameras," in *Proc. IEEE Int. Conf. Robots Autom. (ICRA)*, Anchorage, AK, USA, May 2010, pp. 4525–4530.

[24] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *Proc. IEEE Int. Conf. Robotics Autom. (ICRA)*, Shanghai, China, May 2011, pp. 3719–3726.

[25] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *Proc. 13th IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, Oct. 2013, pp. 292–299.

[26] V. Kumar, Y. Tassa, T. Erez, and E. Todorov, "Real-time behaviour synthesis for dynamic hand-manipulation," in *Proc. IEEE Int. Conf. Robotics Autom. (ICRA)*, May/Jun. 2014, pp. 6808–6815.

[27] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Real-time trajectory generation using model predictive control," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Gothenburg, Sweden, Aug. 2015, pp. 942–948, doi: 10.1109/CoASE.2015.7294220.

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*, 6th ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[29] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Englewood Cliffs, NJ, USA: Prentice Hall, 1997.

[30] D. Sundararajan, *The Discrete Fourier Transform: Theory, Algorithms and Applications*. Singapore: World Scientific, 2001.

[31] ABB Robotics, "ABB IRB140 industrial robot data sheet," 2014, Data Sheet No. PR10031 EN_R15.

[32] A. Blomdell, I. Dressler, K. Nilsson, and A. Robertsson, "Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers," in *Proc. Workshop 'Innov. Robot Control Archit. Demanding (Res.) Appl.,' IEEE Int. Conf. Robot. Autom. (ICRA)*, Anchorage, AK, USA, May 2010, pp. 62–66.

[33] Department of Automatic Control, Lund University. *LabComm Protocol*. Accessed: Nov. 15, 2018. [Online]. Available: http://www.control.lth.se/research/tools-and-software/

[34] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optim. Eng.*, vol. 13, no. 1, pp. 1–27, 2012.

[35] R. Paul, "Manipulator Cartesian path control," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. SMC-9, no. 11, pp. 702–711, Nov. 1979.

[36] R. H. Taylor, "Planning and execution of straight line manipulator trajectories," *IBM J. Res. Develop.*, vol. 23, no. 4, pp. 424–436, Jul. 1979.

[37] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," *ASME J. Dyn. Syst., Meas., Control*, vol. 106, no. 1, pp. 102–106, Mar. 1984.

[38] M. M. G. Ardakani, A. Robertsson, and R. Johansson, "Online minimum-jerk trajectory generation," in *Proc. IMA Conf. Math. Robot.*, Oxford, U.K., Sep. 2015.

[39] J. Åkesson, K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit, "Modeling and optimization with Optimica and JModelica.org—Languages and tools for solving large-scale dynamic optimization problems," *Comput. Chem. Eng.*, vol. 34, no. 11, pp. 1737–1749, 2010.

M. Mahdi Ghazaei Ardakani (M'12) received the M.Sc. degree in secure telecommunication from the Department of Electrical Engineering, Iran University of Science and Technology, Tehran, Iran, in 2006, the master's degree in robotics and intelligent systems from Örebro University, Örebro, Sweden, in 2011, and the Ph.D. degree from the Department of Automatic Control, Lund University, Lund, Sweden, in 2016.

He is currently a Post-Doctoral Researcher at the Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT). His research interests include robotics, system and control theory, and machine learning.

Björn Olofsson received the M.Sc. degree in engineering physics and the Ph.D. degree in automatic control from Lund University, Lund, Sweden, in 2010 and 2015, respectively.

He is currently a Researcher at the Department of Automatic Control, Lund University, and at the Division of Vehicular Systems, Linköping University, Linköping. His research interests include motion control for robots and vehicles, optimal control, system identification, and statistical sensor fusion.

Anders Robertsson (SM'11) received the M.Sc. degree in electrical engineering and the Ph.D. degree in automatic control from LTH, Lund University, Lund, Sweden, in 1992 and 1999, respectively.

He was appointed as a Docent in 2005 and an Excellent Teaching Practitioner in 2007. Since 2012, he has been a Full Professor with the Department of Automatic Control, LTH, Lund University. His research interests include nonlinear estimation and control, robotics research on mobile and industrial manipulators, and feedback control of computing systems.

Rolf Johansson (F'12) received the M.Sc. degree in technical physics, the Bachelor-of-Medicine degree, the doctorate in control theory, and the M.D. degree from Lund University, Lund, Sweden, in 1977, 1980, 1983, and 1986, respectively.

Since 1986, he has been with the Department of Automatic Control, Lund University, where he is currently a Professor of control science. In his scientific work, he has been involved in research in adaptive system theory, mathematical modeling, system identification, robotics, and combustion engine control.

Dr. Johansson is a fellow of the Royal Physiographic Society, Section of Medicine. He was a recipient of the 1995 Ebeling Prize of the Swedish Society of Medicine for distinguished contribution to the study of human balance. Since 1999, he has been an Associate Editor of the *International Journal of Adaptive Control and Signal Processing*. He is a member of the editorial boards of *Mathematical Biosciences* and *Robotics and Biomimetics*.