Abstract—Many robotic applications, such as milling, gluing, or high precision measurements, require the precise following of a pre-defined geometric path. We investigate the real-time feasible implementation of model predictive path-following control for an industrial robot. We consider constrained output path following with and without reference speed assignment. Finally, we present results of an implementation of the proposed model predictive path-following controller on a KUKA LWR IV robot.

Index Terms—path following, nonlinear model predictive control, constraints, optimal control, KUKA LWR IV

I. INTRODUCTION

Not all control tasks arising in applications fit well into the classical framework of set-point stabilization and trajectory tracking. For instance, consider tasks such as steering an autonomous vehicle along a given reference track, precise machine tooling, or control of autonomous underwater vehicles. All these applications have in common that a system should be steered along a pre-specified geometric curve in a (position) output space, whereby the speed to move along the curve is not fixed a priori. Such control tasks are typically denoted as path-following problems.

These problems arise frequently in the context of dynamic motion planning and trajectory generation for mechatronic systems including bi-pedal walking and standard robots [12, 18, 21, 22]. Typically, a dynamic motion is assigned to a geometric reference path by solving an optimal control problem, or via some heuristics, leading to the desired reference motion along the path. The reference motion itself is then tracked by means of some feedback controller. In other words, path-following problems are frequently decomposed into trajectory generation and trajectory tracking, i.e., path following is reformulated as trajectory tracking.

To avoid the reformulation as a tracking problem, different closed-loop path-following control schemes have been proposed, see e.g. [1, 2, 7, 8, 14, 16]. The common underlying idea of these schemes is that the generation of the reference motion along the path, and the computation of inputs to track this motion, are both done in an integrated fashion at the run-time of the controller. In other words, a closed-loop path-following controller directly modulates the reference speed along the geometric path to reduce the path-following error and, at the same time, computes inputs to track this reference motion. Besides geometric feedback designs for path following—e.g. [1, 2, 16]—different model predictive control approaches tailored to path following have been investigated, see [5, 7, 8, 14].

To this date, only a few laboratory implementations of model predictive control tailored to path-following problems of mechatronic and robotic systems have been reported: Discrete time predictive path-following control of an x-y table is presented in [13]. Real-time implementations of sampled-data predictive path-following controllers have been presented in [5], which is based on differential flatness, and [10], wherein predictive path following of an industrial robot with paths defined in the joint space is discussed.

In the present paper, we discuss the design and implementation of a sampled-data nonlinear model predictive path-following control scheme in the presence of input and state constraints. Our main contribution is a proof-of-concept demonstration of predictive path following. The results are obtained from a real-time feasible implementation on a KUKA LWR IV robot in a configuration with three actuated joints, see Figure 1. The implementation relies on a predictive control approach presented in [8], wherein system-theoretic properties, such as stability and path convergence are analyzed, while implementation aspects and experimental results are not discussed. In contrast to the result presented in [10], we use a robot configuration with three joints instead of two. Furthermore, we consider reference paths defined directly in the operational space of the robot, i.e., the Cartesian space.
where \( x \in \mathbb{R}^{n_x}, u \in \mathbb{R}^{n_u}, \) and \( y \in \mathbb{R}^{n_y} \) represent the state, the input and the output. The states are constrained to a closed set, i.e., for all \( t : x(t) \in \mathcal{X} \subseteq \mathbb{R}^{n_x} \). The inputs \( u : [t_0, \infty) \to \mathcal{U} \) are piece-wise continuous and take values in a compact set \( \mathcal{U} \subset \mathbb{R}^{n_u} \), which is briefly denoted by \( u(\cdot) \in \mathcal{PC}(\mathcal{U}) \). The maps \( f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x} \) and \( h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y} \) are assumed to be sufficiently often continuously differentiable and (1a) is considered to be locally Lipschitz. Also note that the control system is assumed to have a square input-output structure, i.e., \( \dim u = \dim y = n_u \). The solution of (1a) at time \( t \), originating at time \( t_0 \) from \( x_0 \), driven by an input \( u(\cdot) \), is denoted as \( x(t, t_0, x_0|u(\cdot)) \).

A. Path-following Problems

Output path-following refers to the task of tracking/following a geometric reference in the output space \( \mathcal{P} \). Here, we assume that this reference is given by

\[
\mathcal{P} = \{ y \in \mathbb{R}^{n_y} | \theta \in \mathbb{R} \mapsto y = p(\theta) \}.
\]  

(2)

The scalar variable \( \theta \in \mathbb{R} \) is called path parameter, and \( p(\theta) \) is a parametrization of \( \mathcal{P} \). Note that in path-following problems there is typically no strict requirement when to be where on \( \mathcal{P} \). In other words, the path parameter \( \theta \) is time dependent but its time evolution \( t \mapsto \theta(t) \) is not specified a priori. Rather the system input \( u(\cdot) \) and the timing \( \theta(\cdot) \) are to be chosen such that the path is followed as exactly as possible. Furthermore, note that in some cases, such as finitely long paths, it can be helpful to restrict the path parameter to \([\theta_0, \theta_1]\), where \( \theta_0 \) is the start point of the path and \( \theta_1 \in [\theta_0, \infty) \) denotes the end point of the path.

Subsequently, we consider the problem of steering the output \( \mathcal{P} \) to the path \( \mathcal{P} \) and following it along in the direction of increasing values of \( \theta \).

Problem 1 (Constrained output path following \[8\]):

Given the system (1) and the reference path \( \mathcal{P} \), design a controller that computes \( u(\cdot) \) and \( \theta(\cdot) \) and achieves:

i) Path convergence: The system output \( y = h(x) \) converges to the set \( \mathcal{P} \) in the sense that

\[
\lim_{t \to \infty} \| h(x(t)) - p(\theta(t)) \| = 0.
\]

ii) Convergence on path: The system moves along \( \mathcal{P} \) in forward direction, i.e.

\[
\dot{\theta}(t) \geq 0 \quad \text{and} \quad \lim_{t \to \infty} \| \theta(t) - \theta_1 \| = 0.
\]

iii) Constraint satisfaction: The constraints on the states \( x(t) \in \mathcal{X} \) and the inputs \( u(t) \in \mathcal{U} \) are satisfied for all times.
Instead of this formulation, one might demand that the path parameter velocity \( \dot{\theta}(t) \) converges to a pre-specified evolution \( \dot{\theta}_{ref}(t) \), cf. [19]. This leads to the following problem.

**Problem 2 (Speed-assigned path following [8]):**
Given the system (1) and the reference path \( \mathcal{P} \), design a controller that computes \( u(\cdot) \) and \( \theta(\cdot) \), achieves part i) & iii) of Problem 1 and guarantees:

- ii) Velocity convergence: The path velocity \( \dot{\theta}(t) \) converges to a predefined profile such that
  \[
  \lim_{t \to \infty} \| \dot{\theta}(t) - \dot{\theta}_{ref}(t) \| = 0.
  \]

**Remark 1 (Speed-assigned paths and tracking):**
Note that path following with velocity assignment is not equivalent to trajectory tracking, as speed assignment does not specify a unique output reference \( p(\theta(t)) \). Rather, the problem with speed assignment admits several reference trajectories \( p(\theta_i(t)), i \in \{1, 2, \ldots\} \), with \( \theta_i(t) = \dot{\theta}_{ref}(t) \) differing with respect to \( \theta \), i.e., \( \theta_i(\cdot) \neq \theta_j(\cdot), i \neq j \). Furthermore, in case disturbances lead to large path deviations, path following with speed assignment allows adjusting the timing \( \theta(t) \), while this can be challenging in standard trajectory-tracking formulations.

The conceptual idea of path following is to treat the path parameter \( \theta \) as a virtual state whereby the time evolution \( t \mapsto \theta(t) \) is influenced by an extra input, cf. [1, 7, 8, 19]. Usually, the time evolution \( t \mapsto \theta(t) \) is described by an additional differential equation termed timing law. Basically, the timing law is an extra degree of freedom in the controller design. Subsequently, we rely on a simple integrator chain as timing law
\[
\theta(k+1) = v,
\]
where \( \hat{\theta} \in \mathbb{N} \) is sufficiently large as outlined in Remark 2. We note that more complex timing laws can be considered. The *virtual input* of the timing law is assumed to be piece-wise continuous and bounded, i.e., \( v(\cdot) \in \mathcal{PC}(\mathcal{V}), \mathcal{V} \subset \mathbb{R} \). Similar to [7, 8], we use the compact notation \( z := (\theta, \dot{\theta}, \ldots, \theta^{(\hat{\theta})})^T \) in (3) to formulate path-following problems via the augmented system
\[
\begin{align*}
\dot{x} &= f(x, u), \\
\dot{z} &= l(z, v), \\
\dot{e} &= h(x) - p(z), \\
\end{align*}
\]

The output (4b) consists of two elements, the path following error \( e = h(x) - p(\theta) \), and the full virtual state \( z \). With respect to the augmented system (4) output path-following (Problem 1) requires that the error \( e \) converges to zero while the path parameter \( \theta \) converges to \( \theta_1 \)—the final path point.

**Remark 2 (Choice of suitable timing laws):**
How should one choose the parameter \( \hat{r} \) in the timing law (3)? This question can be answered using tools from geometric nonlinear control and the concept of *transversal normal forms*, see [2, 7, 8, 15]. The main idea is to choose \( \hat{r} \) sufficiently large such that one can map the augmented system (4) at least locally into suitable coordinates allowing for characterization of the manifold on which any state trajectory corresponds to an output trajectory traveling along \( \mathcal{P} \).

**B. Model Predictive Path-following Control**

We tackle output path-following problems with and without speed assignment in presence of input and state constraints (Problems 1 & 2) via a continuous time sampled-data nonlinear model predictive control (NMPC) scheme, denoted as model predictive path-following control (MPFC), [7, 8].

MPFC is based on the augmented system description (4). As common in model predictive control, the system input is obtained via the repetitive solution of an optimal control problem (OCP). At each sampling instance \( t_k = t_0 + k\delta \), with \( k \in \mathbb{N}_0 \) and sampling period \( \delta > 0 \), the cost functional to be minimized is
\[
J(x(t_k), z(t_k), \bar{u}(\cdot), \bar{v}(\cdot)) = \int_{t_k}^{t_k+T} F(\bar{e}(\tau), \bar{z}(\tau), \bar{u}(\tau), \bar{v}(\tau)) \, d\tau.
\]
As usual in NMPC, \( F: \mathbb{R}^{n_z} \times Z \times U \times V \to \mathbb{R}^+ \) is called cost function and \( T \in (\delta, \infty) \) denotes the prediction horizon. The OCP solved repetitively is:
\[
\text{minimize } J(x(t_k), z(t_k), \bar{u}(\cdot), \bar{v}(\cdot)) \quad (6a)
\]
subject to the constraints
\[
\begin{align*}
\dot{x}(\tau) &= f(\bar{x}(\tau), \bar{u}(\tau)), & \bar{x}(t_k) = x(t_k) \quad (6b) \\
\dot{z}(\tau) &= l(\bar{z}(\tau), \bar{v}(\tau)), & \bar{z}(t_k) = z(t_k) \quad (6c) \\
\dot{e}(\tau) &= h(\bar{x}(\tau)) - p(\bar{z}(\tau)), & \bar{e}(\tau) = \bar{e}(\cdot) \in \mathcal{U} \quad (6d) \\
\bar{x}(\tau) &\in X, \: \bar{u}(\tau) \in U \quad (6e) \\
\bar{z}(\tau) &\in Z, \: \bar{v}(\tau) \in V \quad (6f)
\end{align*}
\]
which have to hold for all \( \tau \in [t_k, t_k + T] \). As common in NMPC we denote predicted variables, i.e., internal variables of the controller, by superscript \( \bar{\cdot} \). The MPFC scheme (6) is built upon the augmented dynamics (4).
and thus they are considered as dynamic constraints in (6b)–(6c).

At the core of the MPFC scheme is the repeated solution of (6) in a receding horizon fashion at all sampling instants \( t_k \). The solution to (6) are optimal input trajectories, denoted as \( \bar{u}^*(\cdot, x(t_k)) \) and \( \bar{v}^*(\cdot, z(t_k)) \). Solving (6) at time \( t_k \) with finite horizon \([t_k, t_k + T]\), one plans a reference motion \( t \mapsto p(\theta(t)) \in \mathcal{P} \) and, at the same time, computes the system inputs to track these trajectories. Finally, \( \bar{u}^*(\cdot, x(t_k)) \) is applied to system (1)

\[
\forall t \in [t_k, t_k + \delta] : u(t) = \bar{u}^*(t, x(t_k)).
\]

At the next sampling instant \( t_{k+1} = t_k + \delta \), the OCP (6) is solved again for new initial conditions. While at each sampling instance the measured or observed state \( x(t_k) \) serves as initial condition for (6b) the initial conditions of the timing law (6c) is the last predicted trajectory evaluated at time \( t_k \), i.e., \( z(t_k) = \bar{z}(t_k, t_{k-1}, \bar{z}(t_{k-1})|\bar{v}_{k-1}(\cdot)) \).

Note that in real-time feasible implementations of predictive control schemes one will typically compute an approximation of the optimal solution \( \bar{u}^*(\cdot, x(t_k)) \), i.e., one will apply a feasible but sub-optimal iterate of a numerical solution scheme. It is worth mentioning that the MPFC scheme (6) does not aim at a time-optimal motion along \( \mathcal{P} \) as it is often considered in robotics [18–20]. Rather, we aim at a feedback strategy ensuring that the system output converges to the path and moves along the path in forward direction.

Summarizing, the MPFC scheme is built upon the receding horizon solution of (6), whereby the system model (6b) with the real system input \( u \) and the virtual path parameter dynamics (6c) with the virtual input \( v \) are part of the optimization problem. Since, the initial condition of (6c) at time \( t_k \) is based on the previous solution at time \( t_{k-1} \), the variable \( z \) can be understood as an internal state of the controller. In other words, the MPFC scheme based on (6) is a dynamic feedback strategy, cf. [8, Remark 1].

Remark 3 (Convergence conditions for MPFC): It is fair to ask for conditions ensuring that the proposed MPFC scheme solves Problems 1 & 2 or guarantees path convergence. The present paper focuses on the application and implementation of the MPFC scheme. Thus a detailed investigation is beyond its scope. As discussed in [7, 8], one can establish sufficient conditions by adding an end penalty and a terminal constraint to the OCP (6). This way one can ensure path convergence and recursive feasibility in the presence of state constraints.

C. Problems with and without Speed Assignment

So far we have put the focus on the general MPFC formulation. Next, we show how to account for problems with and without speed assignment.

We consider a quadratic cost function \( F \)

\[
F(e, z, u, v) = \| (e, z_1 - \theta_1, z_2 - \theta_{ref}) \|^2_Q + \| (u, v) \|^2_R, \tag{7}
\]

since this allows efficient computation of approximations of the Hessian of OCP (6). The weighting matrix \( Q \) is positive semi-definite and \( R \) is positive definite while both are diagonal, i.e., \( Q = \text{diag}(w_e, w_e, w_e, w_g, w_g) \) and \( R = \text{diag}(r_u, r_u, r_u, r_v) \). In order to converge to the path, one usually chooses \( w_e \gg w_g, r_v \). If the task at hand is a path-following problem without speed assignment—i.e., Problem 1 and one wishes to stop at \( z_1 = \theta_1 \)—then one penalizes \( z_1 - \theta_1 \). Hence, \( w_g > 0 \) and \( w_v = 0 \) are used in this case. If, however, the task at hand is a path-following problem with speed assignment—i.e., Problem 2 and achieving \( z_2 - \theta_{ref} \approx 0 \) is of interest—one uses \( w_v > 0 \) and \( w_g = 0 \). Besides the parameters of the cost function \( F \) also the constraints \( Z \) in (6) differ for Problems 1 and 2. While the former problem calls for \( Z = [\theta_0, \theta_1] \times [0, \infty) \times \mathbb{R}^3 \), in the latter problem one should use \( Z = [\theta_0, \infty) \times [0, \infty) \times \mathbb{R}^2 \).

III. IMPLEMENTATION

A. Robot Model and MPFC Design

For simplicity, we consider a robot configuration in which only joints 1, 2 and 4 of the KUKA LWR IV are operated while the other joints are kept fixed. The dynamic model of the robot is given by

\[
B(q)\ddot{q} + C(q, \dot{q})\dot{q} + \tau_F(q) + g(q) = \tau. \tag{8}
\]

Here, \( q = (q_1, q_2, q_4)^T \) is the vector of joint angles. The time derivatives \( \dot{q} \) and \( \ddot{q} \), respectively, refer to angular velocities and angular accelerations. The vector \( \tau = (\tau_1, \tau_2, \tau_4)^T \) denotes the actuation torques applied to the corresponding joints; \( B(q) = B(q)\tau > 0 \) is the inertia matrix; \( C(q, \dot{q}) \) represents the centrifugal and Coriolis effects. The vectors \( \tau_F(q) \) and \( g(q) \) describe torques in the joints due to friction and gravity, respectively. Note that this model describes the robot moving freely, i.e., contact forces are not explicitly included. They are treated as disturbances. The parameters of the model can be found in [3].

---

1 If no initial condition for the first sampling instance \( k = 0 \) is given, one can use \( z(t_0) = (\theta(t_0), 0, \ldots, 0)^T \) whereby \( \theta(t_0) \) locally minimizes the distance \( \| h(x_0) - p(\theta) \| \).
First, we rewrite the model in an implicit state-space representation with \( x_1 = q, x_2 = \dot{q}, u = \tau \). This leads to

\[
E \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ u - C(x_1, x_2)x_2 - g(x_1) - \tau F(x_2) \end{pmatrix},
\]

where \( E = \text{diag}(I, B(x_1)) \) is blockdiagonal. Note that the output \( y = h_{ca}(x_1) \) describes the position of the tip of the robotic arm in a Cartesian coordinate system centered at the base of the robot (the operational space of the robot), cf. Fig. 1. We use this output instead of the flat output \( \tilde{y} = x_1 = q \), as path-following tasks are usually formulated in the operational space. Finally, to obtain the augmented system description (4), we choose an integrator chain of length two for (3).

We express the parametrization \( p(\theta) \) of the paths as a set of polynomial splines. To this end, we consider an equi-distant partition of \([\theta_0, \theta_1]\) given by \( \tilde{\theta}_i = \Delta \theta + \tilde{\theta}_{i-1}, \tilde{\theta}_0 = \theta_0 \). We describe \( P \) by

\[
p(\theta) = \sum_{i=1}^{N_P} H \left( \theta - \tilde{\theta}_i \right) H \left( \tilde{\theta}_{i-1} - \theta \right) \sum_{j=0}^{N_o} a_{i,j} \theta^j.
\]

Here, \( H : \mathbb{R} \rightarrow \{0, 1\} \) denotes the Heaviside step function and \( a_{i,j} \) are polynomial coefficients. The constants \( N_P \) and \( N_o \) denote the number of path segments and, respectively, the order of the polynomial on each segment. The coefficients \( a_{i,j} \) are computed such that \( p(\theta) \) is continuously differentiable.

B. Interfacing and Real-Time Feasible Optimization

The considered KUKA LWR IV robot arm is operated via the Fast Research Interface by an external computer via an Ethernet connection. This interface allows sampling rates up to 1 kHz, which is also the sampling rate of the internal control layer. Furthermore, the interface allows to superpose torques on each joint when operated in the so-called joint-specific impedance control mode. In this mode, the torques commanded to the KUKA LWR IV are composed of torques computed inside the motion kernel (i.e., gravity terms) and the torques computed by an external controller (MPFC) and transferred via Ethernet to the robot. The MPFC scheme is implemented on an external PC workstation running a Linux operating system and a Intel Xeon X5675 CPU with 3.07 GHz clock frequency. The proposed MPFC scheme is written entirely in C/C++.

For sake of faster computations, we simplified the model. First, note that the friction term \( \tau_F(\dot{q}) \) in (5) includes a sign function due to Coulomb friction, which is approximated by an arctan in the OCP. Second, we rely on internal functionalities of the robot allowing for gravity compensation. Thus, the term \( g(q) \) in (5) is neglected in the OCP. The OCP (6) is solved repeatedly at the run-time of the controller using the automatic code generation features presented in (1). Specifically, we use a direct single-shooting implementation available in version 1.2.1beta of the ACADO Toolkit. In each iteration we perform one SQP iteration, i.e., we employ a so-called real-time iteration scheme (6). We use an implicit Gauss-Legendre integrator of order 2 with 10 steps.

The prediction horizon is \( T = 100 \text{ms} \) and the sampling period of the MPFC scheme is \( \delta = 1 \text{ms} \). It is worth noting that, in contrast to discrete-time formulations of NMPC, in the sampled-data framework of Section II the choice of the input parametrization is independent of the chosen sampling period. Here, the input signals are approximated as piece-wise constant functions with 10 equi-distant intervals of 10 ms. Hence, at each sampling time \( t_k \), we solve one SQP iteration with a Hessian of size \( 40 \times 40 \).

The research interface of the robot allows to obtain the joint angles \( x_1 \) from magneto-resistive encoders but not the joint angular velocities \( x_2 \). We employ finite differences and low pass filtering of \( x_1 \) to obtain angular velocities \( x_2 \). Note that the state \( z = (\theta, \dot{\theta})^T \) is merely an internal variable of the controller; thus it does not need to be estimated.

The maximum time to solve the OCP (6) is 0.48 ms (mean 0.24 ms, median 0.18 ms). The overall latency, which consists of the time to solve the OCP and the time needed for state estimation, communication, etc., is below 0.92 ms (mean 0.42 ms, median 0.42 ms). Hence, in experiments a sampling rate of 1 kHz can be achieved, which corresponds to the fastest sampling rate available via the research interface.

IV. Experimental Results

Two different experiments of drawing paths to a white board are presented: a three-leaved clover path and a path representing the word \textit{Hello}. The behavior of the robot during the experiments is documented in the videos available at [9]. Note that the experiments shown in these videos correspond to the results depicted in Figures 2 and 3. The tuning parameters of the MPFC scheme are documented in Table I in the Appendix.

Clover Path: The experimental results for the clover path with speed assignment are shown in Figure 2. We plot trajectories that correspond to 3 turns on the clover. The clover path is a closed curve, the speed assignment along the path leads to periodic behavior of...
the inputs and the angular velocities, see Figure 2a. This behavior can be also observed for the virtual path parameter states in Figure 2b as the speed along the path is depending periodically on the curvature of the path. Furthermore, the MPFC schemes accelerates along straight parts of the path while it slows down at sharp corners of the path due to increased curvature. Although we consider path following with speed assignment, it can be seen in Figure 2a that the path parameter velocity \( z_2 = \dot{\theta} \) does not track its reference value \( \dot{\theta}_{\text{ref}} = 250 \text{ s}^{-1} \). The reason for this behavior is the constraint on the angular velocities of the robot, as the bound of the fourth joint is reached, see Figure 2a.

The path-following errors depicted in Figure 2a are computed via the available kinematic model and the measured joint angles, i.e. \( e_y(t) = h_{ca,i}(q(t)) - p_i(\theta(t)), i = 1, 2, 3 \). As can be seen in Figure 2a, the initial error of the experiment was rather large especially in the directions \( y_2 \) and \( y_3 \). The jump of the error at about \( t = 1 \text{ s} \) is due to switching from position control to gravity compensation of the internal robot control. This gravity compensation cannot compensate exactly the contact forces between pen and board, and therefore the robot arm moves away from the starting point reached before by position control. As soon as the MPFC scheme is switched on at \( t = 2.5 \text{ s} \), the errors \( e_{y_1}, e_{y_2}, e_{y_3} \) decrease rapidly and stay below 1 mm for the rest of the experiment, see Figure 2a. The behavior of the robot in the operational space is shown in Figure 2c. As one can see the MPFC scheme compensates for the initial path deviation rapidly and follows the path accurately.

Hello Path: The experimental results for the Hello path are shown in Figure 3. During this experiment we introduced additional disturbances (external forces) by grabbing and holding the robot arm for a short time span. We applied several disturbances during five different intervals \([4.5, 6.5], [8.5, 9.5], [11.5, 13], [15, 16], [18, 18.5]\). The forces applied to the arm change the behavior of the robot completely by stopping it on the path or even moving it away from the path. This leads to large path deviations, see Figure 3a. The disturbance ends by releasing the arm and the MPFC scheme steers the robot tip back to the path. Note that during the disturbance intervals the controller is not switched off. The controller reacts to these disturbances (and to the increased path error) by slowing down and even stopping the reference motion along the path. This can be seen in the plots of the states and input of the virtual system in Figure 3b. When the disturbance ends, i.e., when the external force is no longer applied, the robot tip returns and follows the path as the virtual system speeds
up again. In the zoomed-in part of Figure 3c, which corresponds to the disturbance during \( t \in [8.5, 9.5] \), one can see how the robot tip is forced to move away from the reference during this interval and how it returns as soon as the disturbance ends. The controller stops at the end of the path as for the Hello path \( z_1(t) - \theta_1 \) is penalized.

Note that the experiments have been performed using essentially the same parameters. The only difference is in the choice of \( w_\theta \) and \( w_{\dot{\theta}} \) in \( Q(7) \) and different box constraints, cf. Table I. These values are passed to the optimization code during run-time, i.e. re-compilation of the auto-generated C/C++ code is not necessary.

V. Summary and Conclusions

This paper presented results on the design and implementation of continuous time nonlinear model predictive control schemes tailored to constrained path-following problems for robotic manipulators. We considered constrained output path following with and without speed assignment. We demonstrated that one can tackle both problems by changing only a few parameters of the model predictive path-following control scheme. The real-time feasibility of the proposed scheme was illustrated via a laboratory implementation on a KUKA LWR IV robot.

The proposed model predictive path-following controller is based on an augmented system description of path-following problems allowing for direct consideration of paths defined in Cartesian space as well as input and state constraints. The presented results underpin that the proposed concept is real-time feasible and shows very promising control performance.

Appendix

References

[1] A.P. Aguiar, J.P. Hespanha, and P.V. Kokotovic. “Path-following for nonminimum phase systems removes performance limitations”. In: IEEE Trans. Automat. Contr. 50.2 (2005), pp. 234–239.

[2] A. Banaszuk and J. Hauser. “Feedback linearization of transverse dynamics for periodic orbits”. In: Sys. Contr. Lett. 26.2 (1995), pp. 95–105.

[3] V. Bargsten, P. Zometa, and R. Findeisen. “Modeling, parameter identification and model-based control of a lightweight robotic manipulator”. In: Proc. of 2013 IEEE International Conference on Control Applications (CCA). Hyderabad, India, 2013, pp. 134–139.

[4] R. Bischoff, J. Kurth, G. Schreiber, R. Koepp, A. Albuschäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, et al. “The KUKA-DLR lightweight robot arm—a new reference platform for robotics research and manufacturing”. In: Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK). VDE. 2010, pp. 1–8.
TABLE I
IMPLEMENTATION PARAMETERS.

|                      | Hello path | clover path |
|----------------------|------------|-------------|
| Path segments $N_P$  | 1800       | 2700        |
| $Q = \text{diag}(q)$ | $w_q = 10^7$ |             |
| $q = (q_u, q_v, q_w, q_g)$ | $w_q = 3 \cdot 10^{-4}$ | $w_q = 0$ |
| $R = \text{diag}(r_u, r_v, r_w, r_g)$ | $r_u = 0.5, r_v = 10^{-7}$ |         |
| Ref. values: $\theta_1, \dot{\theta}_r, e$ | $\theta_1 = 1750$ | $\dot{\theta}_r = 250 \text{s}^{-1}$ |
| $\mathcal{X} = [-\pi, \pi]$ | $\bar{q} = \infty \text{ rad}$ | $\bar{q} = \infty \text{ rad}$ |
| $\mathcal{Q} = (\bar{q}, \bar{q}, \bar{q}, \bar{q})^T$ | $\bar{q} = 0.5 \text{rad/s}$ | $\bar{q} = 0.6 \text{rad/s}$ |
| $\mathcal{U} = [-\bar{q}, \bar{q}]$ | $\bar{q} = 60 \text{Nm}$ |             |
| $\mathcal{Z} = [\bar{z}, \bar{z}]$ | $\theta_0 = 0$ | $\theta_0 = 0$ |
| $\mathcal{Z} = (\theta_0, \infty)^T$ | $\theta_1 = 1750$ | $\theta_1 = \infty$ |
| $\mathcal{V}$ | $[-10^4, 8 \cdot 10^3]$ |             |

[5] M. Böck and A. Kugi. “Real-time Nonlinear Model Predictive Path-Following Control of a Laboratory Tower Crane”. In: IEEE Trans. Contr. Syst. Techn. 22.4 (2014), pp. 1461–1473.

[6] M. Diehl, I. Uslu, R. Findeisen, S. Schwarzkopf, F. Allgöwer, H.G. Bock, T. Bürner, E.D. Gilles, J.P. Kienle A. and Schlöder, and E. Stein. “Real-time optimization for large scale processes: Nonlinear model predictive control of a high purity distillation column”. In: Online Optimization of Large Scale Systems. Springer, 2001, pp. 363–383.

[7] T. Faulwasser. Optimization-based Solutions to Constrained Trajectory-tracking and Path-following Problems. Shaker, Aachen, Germany, 2013.

[8] T. Faulwasser and R. Findeisen. “Nonlinear model predictive control for constrained output path following”. In: IEEE Trans. Automat. Contr. 61.4 (2016), pp. 1026–1039.

[9] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen. Implementation of Model Predictive Path-following Control for an Industrial Robot. Video of experimental results. Accessed July 29, 2016. URL: youtu.be/KMj3cW7-Gfr.

[10] T. Faulwasser, J. Matschek, P. Zometa, and R. Findeisen. “Predictive Path-following Control: Concept and Implementation for an Industrial Robot”. In: Proc. of 2013 IEEE Conference on Control Applications (CCA). Hyderabad, India, 2013, pp. 128–133.

[11] B. Houska, H.J. Ferreau, and M. Diehl. “An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range”. In: Automatica 47.10 (2011), pp. 2279–2285.

[12] V. Kumar, M. Zefran, and J.P. Ostrowski. “Motion planning and control of robots”. In: Handbook of industrial robots. Wiley, 1999, pp. 295–315.

[13] D. Lam, C. Manzie, and M. Good. “Application of Model Predictive Contouring Control to an X-Y Table”. In: Proc. of 18th IFAC World Congress, Milano, Italy. 2011, pp. 10325–10330.

[14] D. Lam, C. Manzie, and M. Good. “Model predictive contouring control”. In: Proc. 49th IEEE Conf. Decision and Control (CDC), Atlanta, GA, USA. 2010, pp. 6137–6142.

[15] C. Nielsen and M. Maggiore. “On local transverse feedback linearization”. In: SIAM Journal on Control and Optimization 47 (2008), pp. 2227–2250.

[16] C. Nielsen, C. Fulford, and M. Maggiore. “Path following using transverse feedback linearization: Application to a maglev positioning system”. In: Automatica 46.3 (2010), pp. 585–590.

[17] G. Schreiber, A. Stemmer, and R. Bischoff. “The fast research interface for the KUKA lightweight robot”. In: IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010). 2010.

[18] K. Shin and N. McKay. “Minimum-time control of robotic manipulators with geometric path constraints”. In: IEEE Trans. Automat. Contr. 30.6 (1985), pp. 531–541.

[19] R. Skjetne, T. Fossen, and P.V. Kokotovic. “Robust output maneuvering for a class of nonlinear systems”. In: Automatica 40.3 (2004), pp. 373–383.

[20] J.-J. Slotine and H.S. Yang. “Improving the efficiency of time-optimal path-following algorithms”. In: IEEE Trans. Robot. Automat. 5.1 (1989), pp. 118–124.

[21] E.R. Westervelt, J.W. Grizzle, C. Chevallereau, J.H. Choi, and B. Morris. Feedback control of dynamic bipedal robot locomotion. Vol. 28. CRC press, 2007.

[22] P. Wieber and C. Chevallereau. “Online adaptation of reference trajectories for the control of walking systems”. In: Robotics and Autonomous Systems 54.7 (2006), pp. 559–566.