Predictive Path-following Control: Concept and Implementation for an Industrial Robot

Timm Faulwasser, Janine Matschek, Pablo Zometa, and Rolf Findeisen

***Abstract*— Many robotic applications, such as milling, glueing, or high precision measurements require the exact following of a pre-defined geometric path. We outline nonlinear model predictive control approaches to path-following problems. We show the real-time feasibility of predictive path following applied to an industrial robot. Specifically, we consider constrained output path following with and without pre-specified reference speeds. The proposed predictive path-following approach is experimentally validated considering a KUKA lightweight robot IV.**

***Index Terms*— nonlinear model predictive control, constrained path following, real-time implementation, robot control**

I. Introduction

Typically, one distinguishes the problems of set-point stabilization and trajectory tracking. While the former refers to the task of stabilizing a fixed point in the state space, the latter describes the design of controllers that ensure tracking of a time-dependent reference signals. However, not all control tasks arising in applications fit well into the framework of stabilization and tracking. An example is the task to steer a robot along a pre-specified geometric curve in its workspace, whereby the speed to move along the curve is not fixed a priori. Clearly, this task is not a set-point stabilization problem. While reformulation as a tracking problem is possible by fixing the speed to move along the curve, this often leads to performance losses.

To overcome this limitation path-following control schemes have been proposed, e.g. [16]. Path-following problems refer to the tracking of a geometric reference with high precision, whereby the timing to move along the reference is of secondary interest, and can be considered as an additional degree of freedom. Several approaches to path-following problems have been developed over the recent years. Examples are geometric and Lyapunov-based design methods [13, 16]. While stability guarantees can be given for such approaches, it is in general difficult to consider constraints. As an alternative, which allows for input and state constraints, predictive path-following concepts have been suggested in [6, 7, 11, 17]. The aforementioned geometric and Lyapunov-based path-following methods usually deal with paths defined in output spaces—a problem which is termed as *output path following*. In contrast to that, several works on predictive path-following consider geometric references in the state space, i.e., so called *state space path following*, see [7, 17]. From an applications point of view, however, output path following is more relevant. First approaches to predictive output path following are presented in [6, 11].

So far, only a few successful applications and implementations of path following to real systems have been reported. In [13] the application of geometric path-following methods to a magnetic levitation system is discussed; discrete time predictive path-following control of an x-y table is presented in [10]. Successful real-time implementations of continuous-time predictive path-following controllers have not been reported yet.

In this paper we consider the design of a continuous-time sampled-data nonlinear model predictive path-following control schemes in the presence of input and state constraints. We outline and review two variants of path-following problems: constrained output path-following with and without an assignment of the reference speed along the path. Our main contribution is a *proof-of-concept* demonstration of predictive path-following, which is obtained from a real-time feasible implementation on a KUKA LWR IV robot in a configuration with two actuated joints.

The remainder of the paper is structured as follows: in Section II we briefly review the problem of path following as well as the conceptual ideas of model predictive path-following control. Details of the implementation of the predictive path-following controller are discussed in Section III. The results from laboratory experiments are presented in Section IV.

Notation

The Euclidean norm of a vector $x \in \mathbb{R}^{n_x}$ is denoted as $\|x\|$. A trajectory $y : [0, T) \to \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ is briefly written as $y(\cdot)$. The solution of an ODE $\dot{x} = f(t, x, u)$ at time t originating at time t_0 from x_0 driven by an input $u(\cdot)$ is denoted as $x(t, t_0, x_0|u(\cdot))$.

II. Predictive Path Following

We consider nonlinear systems of the form

$$\dot{x} = f(x, u), \quad x(t_0) = x_0, \tag{1a}$$

$$y = h(x), \tag{1b}$$

where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$, and $y \in \mathbb{R}^{n_u}$ represent respectively the state, the input, and the output of the system. 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 piecewise 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

TF is with the Laboratoire d'Automatique, Ecole Polytechnique Federale de Lausanne, Switzerland. E-mail: timm.faulwasser@epfl.ch. JM, PZ and RF are with the Institute for Automation, Otto-von-Guericke-Universität Magdeburg, Germany. E-mail: {janine.matschek, pablo.zometa, rolf.findeisen}@ovgu.de.

maps $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ 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., $\dim u = n_u = \dim y$.

A. Path-following Problems

Output path-following refers to the task of stabilizing a geometric reference in the output space (1b) of system (1) [13, 16]. Here, we assume that this reference is given as a geometric curve

$$\mathcal{P} = \{y \in \mathbb{R}^{n_u} \mid \theta \in [\theta_0, \theta_1] \mapsto y = p(\theta)\}. \tag{2}$$

The scalar variable $\theta \in \mathbb{R}$ is called the path parameter, and $p(\theta)$ is a parametrization of $\mathcal{P}$, which is assumed to be sufficiently often continuously differentiable. The path is a geometric reference. Note that for path-following problems there is no strict requirement *when to be where* on $\mathcal{P}$. In other words, the path parameter θ is time dependent but its time evolution $t \mapsto \theta(t)$ is not specified a priori. Rather the system input $u : [t_0, \infty) \to \mathcal{U}$ and the timing $\theta(t)$ are chosen such that the path is followed as exactly as possible.

Subsequently, we investigate the problem of steering the output (1b) to the path $\mathcal{P}$ and following it along in direction of increasing values.

Problem 1 (Constrained output path following):
Given the system (1) *and the reference path* $\mathcal{P}$ (2)*, design a controller that achieves:*

i) *Path Convergence: The system output* $y = h(x)$ *converges to the set* $\mathcal{P}$ *s.t.:*

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

ii) *Monotonous Forward Motion: The system moves along* $\mathcal{P}$ *in the direction of increasing values of* θ*, s.t.* $\dot\theta(t) \geq 0$ *holds for all* $\theta \in [\theta_0, \theta_1]$ *and* $\lim_{t\to\infty} \theta(t) = \theta_1$.

iii) *Constraint Satisfaction: The constraints on states* $x(t) \in \mathcal{X}$ *and inputs* $u(t) \in \mathcal{U}$ *are satisfied for all times* $t \geq t_0$.

Instead of this formulation, one might require that the path parameter velocity $\dot\theta(t)$ converges to a pre-specified evolution $\dot\theta_{ref}(t)$, cf. [1, 16]:

Problem 2 (Speed-assigned constrained path following):
Given the system (1) *and the reference path* $\mathcal{P}$ (2)*, design a controller that achieves:*

i) *Path Convergence: The system output* $y = h(x)$ *converges to the set* $\mathcal{P}$ *s.t.:*

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

ii) *Velocity Convergence: The path parameter velocity converges to a pre-specified function* $\dot\theta_{ref}(t)$ *s.t.:*

$$\lim_{t\to\infty} \left\|\dot\theta(t) - \dot\theta_{ref}(t)\right\| = 0.$$

iii) *Constraint Satisfaction: The constraints on states* $x(t) \in \mathcal{X}$ *and inputs* $u(t) \in \mathcal{U}$ *are satisfied for all times* $t \geq t_0$.

The conceptual idea of many approaches to path-following problems is to treat the path parameter θ as a virtual state whereby the time evolution $t \mapsto \theta(t)$ can be influenced by an extra input [5, 16]. Usually, the time evolution $t \mapsto \theta(t)$ is described by an additional differential equation which is 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

$$\theta^{(\hat r+1)} = v, \tag{3}$$

where $\hat r \in \mathbb{N}$ is sufficiently large as outlined later.

The *virtual* input of the timing law is assumed to be piecewise continuous and bounded, i.e., $v(\cdot) \in \mathcal{PC}(\mathcal{V}), \mathcal{V} \subset \mathbb{R}$. Using the representation $z := (\theta, \dot\theta, \ldots, \theta^{(\hat r)})^T$ of (3), path-following problems can be analyzed via the augmented system

$$\begin{pmatrix}\dot x \\ \dot z\end{pmatrix} = \begin{pmatrix} f(x,u) \\ l(z,v)\end{pmatrix}, \quad \begin{pmatrix} x(t_0) \\ z(t_0)\end{pmatrix} = \begin{pmatrix} x_0 \\ z_0\end{pmatrix}, \tag{4a}$$

$$\begin{pmatrix} e \\ \theta\end{pmatrix} = \begin{pmatrix} h(x) - p(z_1) \\ z_1\end{pmatrix}. \tag{4b}$$

In this description the system dynamics (1a) are augmented by the dynamics of the path parameter state $z := (\theta, \dot\theta, \ldots, \theta^{(\hat r)})^T$, i.e., by $\dot z = l(z, v)$ which is simply a state space representation of (3). The output (4b) consists of two elements, the path following error $e = h(x) - p(\theta)$, and the path parameter $\theta = z_1$. With respect to the augmented system (4) output path-following (Problem 1) means to ensure that the error output e converges to zero while the path parameter output θ converges to θ_1, which corresponds to the final path point.

Remark 1 (Choice of suitable timing laws):
It is fair to ask how to choose the parameter $\hat r$ *in the timing law* (3)*. If system* (1) *has a well-defined vector relative degree with respect to the output* (1b) *then one can choose* $\hat r$ *such that the relative degree of the timing law* (3) *is equal or larger than the largest component of the vector relative degree of* (1)*. This way, one can map the augmented system* (4) *at least locally into suitable coordinates, which allow identifying the directions in the state space which are transversal to the manifold of state trajectories corresponding to output trajectories travelling along* $\mathcal{P}$*. To obtain such a normal form one uses the path error* e *and its time derivatives as new state variables (plus additional states if required). System representations in such coordinates are termed transversal normal forms. For details on these normal forms we refer to [2, 5, 12].*

B. Model Predictive Path-following Control

Subsequently, we briefly outline the main aspects of model predictive path-following control (MPFC) as proposed in [5]. We solve output path-following problems in presence of input and state constraints via a continuous time sampled-data nonlinear model predictive control (NMPC) scheme, which we denote as model predictive path-following control. In essence, we tailor continuous time sampled-data NMPC schemes to tackle Problem 1, see [8]. We first focus on

Problem 1. How to tailor the MPFC scheme to problems with speed assignment (Problem 2) is discussed later.

Our control scheme is based on the augmented system description (4). As common in NMPC the applied 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

$$J\left(x(t_k), \bar{e}(\cdot), \dot{\bar{e}}(\cdot), \bar{\theta}(\cdot), \bar{u}(\cdot), \bar{v}(\cdot)\right) = \int_{t_k}^{t_k+T_p} F\left(\bar{e}(\tau), \dot{\bar{e}}(\tau), \bar{\theta}(\tau), \bar{u}(\tau), \bar{v}(\tau)\right) d\tau. \quad (5)$$

As usual in NMPC $F : \mathbb{R}^{n_u} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathcal{U} \times \mathcal{V} \to \mathbb{R}_0^+$ is called cost function and T_p denotes the prediction horizon. The OCP to be solved repetitively is:

$$\underset{(\bar{u}(\cdot),\bar{v}(\cdot))\in\mathcal{PC}(\mathcal{U}\times\mathcal{V})}{\text{minimize}} \; J\left(x(t_k), \bar{e}(\cdot), \dot{\bar{e}}(\cdot), \bar{u}(\cdot), \bar{\theta}(\cdot), \bar{v}(\cdot)\right) \quad (6a)$$

subject to the constraints

$$\dot{\bar{x}}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}(t_k) = x(t_k) \quad (6b)$$
$$\dot{\bar{z}}(\tau) = l(\bar{z}(\tau), \bar{v}(\tau)), \quad \bar{z}(t_k) = z(t_k) \quad (6c)$$
$$\bar{e}(\tau) = h(\bar{x}(\tau)) - p(\bar{z}_1(\tau)) \quad (6d)$$
$$\dot{\bar{e}}(\tau) = \frac{\partial h}{\partial x} f(\bar{x}(\tau), \bar{u}(\tau)) - \frac{\partial p}{\partial \theta} l(\bar{z}(\tau), \bar{v}(\tau)) \quad (6e)$$
$$\bar{\theta}(\tau) = \bar{z}_1(\tau) \quad (6f)$$
$$\bar{x}(\tau) \in \mathcal{X}, \; \bar{u}(\tau) \in \mathcal{U} \quad (6g)$$
$$\bar{z}(\tau) \in \mathcal{Z}, \; \bar{v}(\tau) \in \mathcal{V} \quad (6h)$$

which have to hold for all $\tau \in [t_k, t_k + T_p]$. As mentioned the MPFC scheme is built upon the augmented dynamics (4), and thus they are considered as dynamic constraints in (6b–f). One should note, however, that we penalize the path error e as well as its time derivative $\dot{e}$ in the cost function F. Hence $\dot{e}$ is added as additional output to the dynamics. This choice is motivated by the fact that penalizing $\dot{e}$ helps to avoid undesirable oscillations around the path. This way we avoid explicitly mapping the augmented system (4) to a transverse normal form, which might exist only locally, cf. Remark 1. We also need to ensure that for $F(\cdot) \to 0$ the error converges such that $\|e\| \to 0$ and the path parameter $\theta \to \theta_1$. Thus we assume that $F : \mathbb{R}^{n_u} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathcal{U} \times \mathcal{V} \to \mathbb{R}_0^+$ is bounded from below by a class $\mathcal{K}$ function $\psi(\|(e, \theta - \theta_1)^T\|)$. Naturally, it is also possible to penalize higher time derivatives of $e(t)$ directly in the cost function. To ensure path convergence it suffices, however, to require the lower boundedness by $\psi(\|(e, \theta - \theta_1)^T\|)$, cf. [5].

Furthermore, the path parameter dynamics (6c) are subject to the constraints (6h) where the constraint $\mathcal{Z}$ is defined as

$$\mathcal{Z} := \left\{[\theta_0, \theta_1] \times [0, \infty] \times \mathbb{R}^{\hat{r}-1}\right\} \subset \mathbb{R}^{\hat{r}+1}. \quad (7)$$

This constraint ensures that $\bar{\theta} = \bar{z}_1 \in [\theta_0, \theta_1]$ and $\dot{\bar{\theta}} \geq 0$. In order to avoid impulsive solutions to the path parameter dynamics $\dot{\bar{z}} = l(\bar{z}, \bar{v})$ the admissible path parameter inputs $\bar{v}$ are restricted to a compact set $\mathcal{V} \subset \mathbb{R}$ containing 0 in its interior.

While at each sampling instance the measured 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. In other words, $z(t_k) = \bar{z}(t_k, t_{k-1}, \bar{z}(t_{k-1})|\bar{v}_{k-1}(\cdot))$. If no initial condition for the first sampling instance $k = 0$ is given, we obtain $z(t_0)$ via

$$\bar{z}(t_0) = (\theta(t_0), 0, \ldots, 0)^T,$$
$$\theta(t_0) = \underset{\theta\in[\theta_0,\theta_1]}{\text{argmin}} \; \|h(x_0) - p(\theta)\|.$$

To initialize the scheme a value $\theta(t_0)$ which (at least locally) minimizes the distance $\|h(x_0) - p(\theta)\|$ should be obtained. The solution to the OCP (6) leads to optimal input trajectories, denoted as $\bar{u}^\star(\cdot, x(t_k))$ and $\bar{v}^\star(\cdot, z(t_k))$. Finally, $\bar{u}^\star(\cdot, x(t_k))$ is applied to system (1) such that for all $t \in [t_k, t_k + \delta)$: $u(t) = \bar{u}^\star(t, x(t_k))$.[1]

Summarizing, the open-loop optimal control problem is augmented by the virtual path parameter state $z = (\theta, \dot{\theta}, ..., \theta^{(\hat{r})})^T$ and by the virtual input v. Essentially, v is used to control the path evolution, i.e., to influence $t \mapsto \theta(t)$. Basically, one obtains the real system inputs $u(\cdot)$ as well as the virtual input $v(\cdot)$ via repeated solutions of the OCP (6). Thus two problems are tackled at once: the planning of suitable trajectories $t \mapsto p \circ \theta(t)$ inside $\mathcal{P}$, as well as the computation of inputs $u(\cdot)$ to track the resulting trajectories. Note that we do not aim at a time-optimal motion along the path as it is often considered in robotics [15]. Instead we aim at stabilization of the zero-path-error manifold. We want to achieve that the system state converges to the path and moves along it as close as possible. In other words, *path convergence is more important than speed*.

Remark 2 (Sufficient convergence conditions):
It is fair to ask for conditions ensuring that the proposed MPFC scheme solves Problem 1 or guarantees path convergence. As discussed in [5, 7] one can rely on a quasi-infinite horizon NMPC approach to achieve this. Basically, one can add an end penalty and a terminal constraint to the OCP (6)*. Another approach to convergence conditions for MPFC is sketched in [6]. It relies on a specific system structure (exact static feedback linearizability) and on a constrained controllability assumption. This paper, however, is focused on the implementation of the MPFC scheme to a real system, and thus the investigation of stability issues is beyond its scope.*

C. MPFC with Speed Assignment

So far we have put the focus on the constrained output path-following problem. Subsequently, we briefly sketch how to modify the MPFC scheme (6) such that also the speed-assigned constrained path-following problem can be considered.

Note that if speed-assigned path following (Problem 2) is considered, one should modify the output (4b). Since speed-assigned path following implies to track a reference profile

[1] Usually, in real-time feasible implementations one will merely compute an approximation of the optimal solution $\bar{u}^\star(\cdot, x(t_k))$, i.e., one will apply a feasible but suboptimal iterate of a numerical solution scheme.

of the path parameter velocity $\dot{\theta} = z_2$, it is helpful to regard z_2 instead of z_1 as an output of (4). Furthermore, the cost function F appearing in the cost functional (5) should be modified to $\tilde{F}$. To tackle Problem 2 we need to ensure that convergence of $\tilde{F}(\cdot)$ to 0 implies $\|e\| \to 0$ and $\dot{\theta} \to \dot{\theta}_{ref}(t)$. Thus we assume that $\tilde{F} : \mathbb{R}^{n_u} \times \mathbb{R}^{n_u} \times \mathbb{R} \times \mathcal{U} \times \mathcal{V} \to \mathbb{R}_0^+$ is bounded from below by a class $\mathcal{K}$ function $\psi(\|(e, \dot{\theta} - \dot{\theta}_{ref}(t))^T\|)$. Note that time-varying reference speed profiles imply time variance of the cost function $\tilde{F}$. If the velocity reference does not converge to 0—i.e., $\lim_{t\to\infty} \|\dot{\theta}_{ref}\| \neq 0$—the path parameter θ might grow unbounded upon path convergence. Therefore, the constraints on the path parameter state (7) have to be modified, i.e., the constraint on $\theta = z_1 \in [\theta_0, \theta_1]$ should be dropped.

III. Implementation on a KUKA LWR IV

In the following we outline that the presented approach can be used for real-time feasible path following control of an industrial robot. To this end we sketch the implementation of the MPFC scheme on a KUKA LWR IV robot arm. The considered robot has seven actuated revolute joints, cf. Figure 1, left part. Furthermore, it can be controlled by an external computer via an Ethernet connection [14]. For an overview of this robot we refer the reader to [4]. The control interface allows to superpose control torques on each joint when operated in the so called *joint-specific impedance control mode*. In this mode the torques commanded to the LWR are composed of torques computed inside the motion kernel (i.e. gravity terms) and the torques computed by an external controller and transferred via Ethernet. For the purpose of a real-time implementation, the MPFC scheme is implemented on an external computer, i.e, we use a PC workstation running a Linux operating system and a Intel Xeon X5675 6-core CPU with 3.07 GHz clock frequency.

For the purpose of this paper we consider that only joints number 2 and 4 are operated while the other joints are kept fixed, i.e., we use a configuration known as *two-link planar arm*, cf. Figure 1. The Lagrange formulation yields the joint space dynamic model of the robot

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

Here, $q = (q_1, q_2)^T$ is the vector of joint angular positions, where q_1 corresponds to joint 2, and q_2 to joint 4. The time derivatives $\dot{q}$ and $\ddot{q}$, respectively, refer to the angular velocities and angular accelerations. The vector $\tau = (\tau_1, \tau_2)^T$ denotes the actuation torques applied to the considered joints; $B(q)$ is the inertia matrix which is symmetric positive definite; $C(q, \dot{q})$ represents the centrifugal and Coriolis effects. The vectors $\tau_F(\dot{q})$ and $g(q)$ describe torques in the joints due to friction and gravity, respectively. Note that this model describes the robot moving freely in space, i.e., contact forces are not explicitly included in (8). In other words, they are treated as disturbances. The parameters of the model have been taken from an open-source toolbox, which provides Python and C implementations of parametrized models of the KUKA LWR IV in different configurations [3].

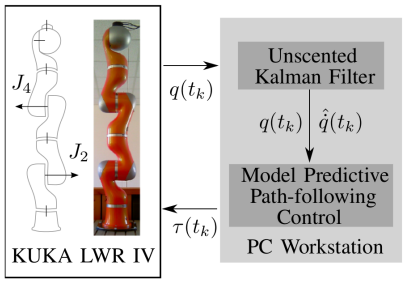


Fig. 1. Sketch of the implemented control setup.

As a first step of the controller design we rewrite the model in a state space representation with $x_1 = q, x_2 = \dot{q}$, and $u = \tau$. This leads to

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ B^{-1}(x_1)\,(u - k(x_1, x_2)) \end{pmatrix} \tag{9a}$$

$$y \quad = x_1 \tag{9b}$$

$$y_{ca} \; = h_{ca}(x_1), \tag{9c}$$

whereby the term $k(x_1, x_2) = C(x_1, x_2)x_2 + g(x_1) + \tau_F(x_2)$ is used for sake of brief notation. The output (9b) refers to the joint space, while (9c) describes the Cartesian workspace of the robot. This second output is added since the considered path-following tasks are usually formulated in the Cartesian workspace. Specifically, we consider two different paths in Cartesian coordinates: a circle and a Lissajous curve. Their parametrizations are as follows

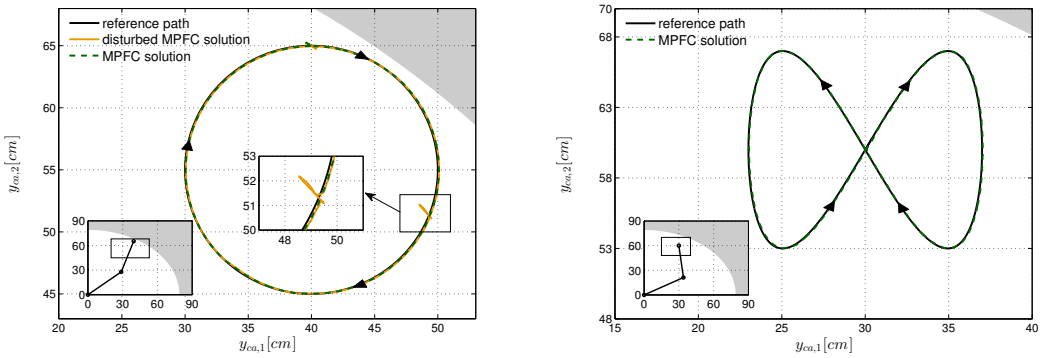
$$r_{circ}(\theta) = \rho \begin{pmatrix} \sin\theta, & \cos\theta \end{pmatrix}^T + r^0_{circ}, \tag{10a}$$

$$r_{lis}(\theta) = a \begin{pmatrix} \sin\theta, & \sin 2\theta \end{pmatrix}^T + r^0_{lis}, \tag{10b}$$

whereby r_{circ} refers to the parametrization of the circle, and r_{lis} is the parametrization of the Lissajous curve. Note that arbitrary paths are possible provided they are sufficiently often continuously differentiable and can be mapped to the joint space. The free parameters are $a = 0.07\,\mathrm{m}$ and $\rho = 0.1\,\mathrm{m}$. In order to simplify the computations the reference paths are mapped from the Cartesian workspace to the joint space. To this end one solves the inverse kinematics problem. Although a closed-form solution can be found, there are in general multiple solutions. This means that a non-unique map $h_{ca}^{-1} : y_{ca} \mapsto y = h_{ca}^{-1}(y_{ca})$, which is the inverse of the Cartesian output map (9c), can be stated. The considered reference paths in the Cartesian workspace are depicted in Figure 2. The region outside of the reachable space is drawn in light grey.

In the joint space the reference paths are described as $p(\theta)_{circ,lis} = h_{ca}^{-1} \circ r_{circ,lis}(\theta)$. This description in the joint space is used to obtain the augmented system (4). The dynamics of the path parameter state $z = (\theta, \dot{\theta})^T$ are an integrator chain of length two. This way we ensure that substitution of (9a-b) into (4) leads to an augmented system with a vector relative degree of $r = (2, 2, 2)$ for Problem 1 and $r = (2, 2, 1)$ for Problem 2.

Note that the Ethernet interface of the robot allows to measure only the joint angles x_1 but not the joint angular



(a) Circle

(b) Lissajous curve

Fig. 2. Problem setting and result in Cartesian space. Grey areas depict regions which are not reachable by the arm.

velocities x_2. Predictive control schemes, however, rely on full state information. Thus we need to reconstruct the states by a suitable state estimator of the dynamics (9a), cf. Figure 1. We use an unscented Kalman filter to estimate x_2 considering all the terms in (8). The state of the path parameter dynamics $z = (\theta, \dot{\theta})^T$ is merely an internal variable of the controller; thus it does not need to be estimated.

Note that the friction term $\tau_F(\dot{q})$ in (8) includes a sign function due to Coulomb friction. To simplify the computations this term is neglected in the model used in the OCP. The Coriolis and centrifugal forces acting on the KUKA LWR IV are small in relation to the other terms in (8). Additionally, we rely on internal functionalities of the robot allowing for gravity compensation. Thus the terms $C(q,\dot{q})\dot{q}$ and $g(q)$ in (8) are also neglected in the OCP. In other words, the term $k(x_1, x_2)$ in (9a) is not present in the prediction model.

For the case of constrained output path following without velocity assignment—i.e., Problem 1—we employ the cost function

$$F(e, \dot{e}, \theta, u, v) = \left\| (e, \dot{e}, \theta - \theta_1)^T \right\|_Q^2 + \left\| (u, v)^T \right\|_R^2 ,$$

whereby the weighting matrices Q and R are positive definite and, respectively, positive semi-definite. Since this task requires to stop at the end of the path we penalize $\theta - \theta_1$. If path following with speed assignment—i.e., Problem 2—is considered, we use

$$\tilde{F}(e, \dot{e}, \dot{\theta}, u, v) = \left\| \left(e, \dot{e}, \dot{\theta} - \dot{\theta}_{ref}(t) \right)^T \right\|_Q^2 + \left\| (u, v)^T \right\|_R^2 ,$$

whereby the weighting matrices Q and R are positive definite and, respectively, positive semi-definite. Here, the cost function penalizes the velocity error $\dot{\theta} - \dot{\theta}_{ref}(t)$. For all considered cases we use a sampling time $\delta = 5\,\mathrm{ms}$ and a prediction horizon $T_p = 50\,\mathrm{ms}$. The input signals are approximated as piecewise constant functions with 10 equi-distant intervals. The input constraints are $|\tau_{1,2}| \leq 10\,\mathrm{Nm}$ and $v \in [-15, 0.1]$ or $v \in [-5, 0.5]$ for circle and Lissajous curve, respectively; the state constraints are $|x_2| \leq 1.7\,\mathrm{rad} \cdot \mathrm{s}^{-1}$, and $\dot{\theta} \geq 0$ or $\dot{\theta} \in [0, 0.25]$. In the case of Problem 1 we additionally consider $\theta \in [-2\pi, 0]$. Also note that we consider neither a terminal penalty nor a terminal region constraint.

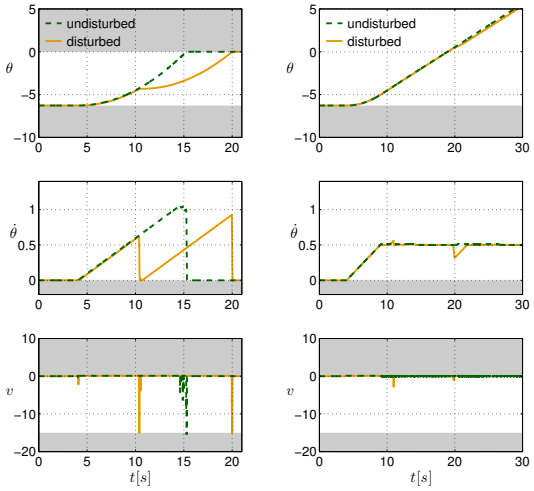
Our prototype implementation combines C and MATLAB code, and runs on a non real-time Linux system. This introduces undesired computational delays that limit the lowest sampling period to $5\,\mathrm{ms}$. It is worth mentioning that the maximum computation time of the optimization is below $1\,\mathrm{ms}$. OCP (6) is solved repeatedly using the automatic code generation features presented in [9].

IV. Experimental Results

The dashed green and orange lines in Figure 2 depict the behavior in the Cartesian workspace of the robot. The left side shows the circular path. It starts at $y_{ca,1} = 40\,\mathrm{cm}$, $y_{ca,2} = 65\,\mathrm{cm}$. It can be seen that the robot's end effector does not initially lie on the path. The right side of Figure 2 shows the Lissajous curve, the initial path point is the double point of the curve.

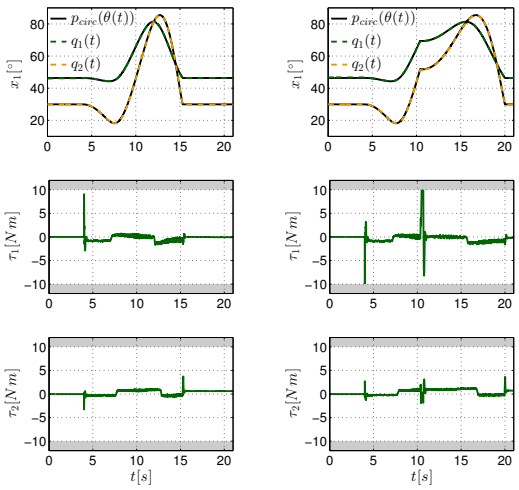
The behavior of the path parameter state z and the virtual input v is shown in Figure 3. Note that the MPFC starts at $t = 4\,\mathrm{s}$. The state and input constraints are pictured in light grey. For the case of path following without speed assignment, depicted in Figure 3(a), the path parameter reaches its end point $\theta = \theta_1 = 0$ and remains there ($\dot{\theta} = 0$). Whereas in the case with velocity assignment, Figure 3(b), the path parameter velocity $\dot{\theta}$ converges to a desired value and the path parameter θ has no upper bound. In the event of disturbances (orange lines)—which have been manually applied to the robot arm by pushing the end effector away from the path—the path velocity slows down in order to minimize the deviation from the path, see Figure 3(a) at $t = 10.5\,\mathrm{s}$ and Figure 3(b) at $t = 20\,\mathrm{s}$.

Figure 4 shows the closed-loop trajectories of the joint positions x_1 and the corresponding input u for the circular path. The input constraints are drawn in light grey, whereas the state constraints are not depicted, since they lie outside of the shown area and do not become active. Figure 4(a) displays the case of an undisturbed circle, whereas Figure 4(b) shows the case of a circle with a disturbance happening at $10.5\,\mathrm{s}$. The MPFC starts in both cases at $t = t_0 = 4\,\mathrm{s}$. During the disturbance as well as when the motion starts,



(a) No speed assignment. (b) Speed assignment.

Fig. 3. Evolution of the path parameter state $z = (\theta, \dot{\theta})^T$



(a) Undisturbed case. (b) Disturbed case.

Fig. 4. Time evolution of x_1 and τ for circular path (Problem 1).

the applied torques are relatively large. The latter case is due to significant differences between the initial path reference $p_{circ}(\theta(t_0))$ and the robot initial position $x_1(t_0)$. As it can be seen in Figure 4(a) and Figure 4(b), the evolutions of the path parameter differ from each other due to the disturbance. In essence, the controller manipulates the path evolution and the joint torques to reduce the path deviation. Loosely speaking, the MPFC scheme *forces* the reference *to wait* for the system output to come back to the reference path. The capability to use the non-fixed reference velocity as a degree of freedom in the control algorithm to increase the systems performance is one of the main advantages of the MPFC.

V. Conclusions

We present results on the design and implementation of continuous time nonlinear model predictive control schemes tailored to constrained path-following problems. The design of the proposed controllers is based on an augmented system description of path-following problems, it allows for consideration of input and state constraints. Furthermore, we demonstrate the benefits of our approach by drawing upon an implementation of the predictive path-following controllers on a lightweight industrial robot. These results underpin that the proposed concept allows for good performance while being real-time feasible.

References

[1] A.P. Aguiar, J.P. Hespanha, and P.V. Kokotovic. “Performance limitations in reference tracking and path following for nonlinear systems”. In: *Automatica* 44.3 (2008), pp. 598–610.

[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. “Modelling, parameter identification and model-based control of a robotic manipulator”. In: *Control Applications (CCA), 2013 IEEE International Conference on*. To appear. IEEE. 2013.

[4] R. Bischoff 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.

[5] T. Faulwasser. *Optimization-based solutions to constrained trajectory-tracking and path-following problems*. Shaker, Aachen, Germany, 2013.

[6] T. Faulwasser and R. Findeisen. “Predictive Path Following without Terminal Constraints”. In: *Proc. of the 20th Int. Symposium on Mathematical Theory of Networks and Systems (MTNS), Melbourne, Australia*. 2012.

[7] T. Faulwasser, B. Kern, and R. Findeisen. “Model predictive path-following for constrained nonlinear systems”. In: *Proc. 48th IEEE Conf. on Decision and Control (CDC)*. 2009, pp. 8642–8647.

[8] F. Fontes. “A General Framework to Design Stabilizing Nonlinear Model Predictive Controllers”. In: *Sys. Contr. Lett.* 42(2) (2001), pp. 127–143.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] S. Yu, L. Xiang, C. Hong, and F. Allgöwer. “Nonlinear Model Predictive Control for Path Following Problems”. In: *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference*. 2012, pp. 145–150.