Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-26 (2018) 118–123

Constrained Predictive Control of a Robotic Manipulator using quasi-LPV Representations

Pablo S.G. Cisneros * **Aadithyan Sridharan** *
Herbert Werner *

* *Hamburg University of Technology, Hamburg, Germany, (email: pablo.gonzalez@tuhh.de; aadithyan.sridharan@tuhh.de; h.werner@tuhh.de).*

Abstract: In this paper a practical approach to Nonlinear Model Predictive Control (NMPC) of a robotic manipulator subject to nonlinear state constraints is presented, which leads to a successful experimental implementation of the control algorithm. The use of quasi-LPV modelling is at the core of this scheme as complex nonlinear optimization is replaced by efficient Quadratic Programming (QP) exploiting the quasi-linearity of the resulting model and constraints. The quasi-LPV model is obtained via velocity-based linearization which results in an exact representation of the nonlinear dynamics and enables stability guarantees with offset-free control. The experimental results show the efficiency and efficacy of the algorithm, as well as its robustness to unmodelled dynamics.

Keywords: Predictive control, linear parameter-varying, optimal control, robotic manipulator, constrained control.

1. INTRODUCTION

Model Predictive Control (MPC) is one of the few control schemes which can systematically deal with input- and state-constrained systems. However, when the system or constraints are nonlinear and the systems dynamics are fast, the complexity of Nonlinear MPC (NMPC) might preclude real-time implementation. For this reason, there has been an increasing interest in reducing the computational complexity entailed by NMPC. The real time iterations algorithm (Diehl et al. (2005)) is an example of a sequential approximate method which efficiently solves the nonlinear optimization problem using a Sequential Quadratic Programming (SQP) approach. A different approach, which also finds a solution to the nonlinear problem solving a sequence of Quadratic Programs (QP) is the qLMPC (quasi-LPV MPC) algorithm introduced in Cisneros et al. (2016). This method uses quasi-LPV modelling to express the nonlinear system as a parameter varying one. Exploiting the knowledge of the parameter trajectory, given its dependency on the state trajectory, allows for the system to be expressed as a Linear Time-Varying (LTV) model and to solve the optimization problem via a sequence of QPs. Numerical comparisons between these approaches have been carried out in Cisneros and Werner (2017b) and Cisneros and Werner (2017a) showing comparable results.

We make use of the velocity-algorithm NMPC (Cisneros and Werner (2018)), which has the following features: stability guarantees and recursive feasibility even for unreachable set points, built-in offset-free control, simple implementation and computational efficiency. To find a quasi-LPV model of the dynamics, velocity-based linearization is used, which naturally results in a model in velocity form for which the velocity algorithm is directly applicable.

Fig. 1. CRS A465 6-DOF robotic manipulator

In this paper we demonstrate the effectiveness and efficiency of the qLMPC algorithm by means of a case-study, the set point tracking and obstacle avoidance problem on a 2 Degree of Freedom (DOF) robotic manipulator. The obstacle avoidance problem for series robotic manipulators has been revisited numerous times because of its practical relevance. To name a few examples: optimal path planning by means of online optimization for time-varying obstacle avoidance has been reported in dos Santos et al. (2008). This approach bears strong resemblance to MPC, if one chooses mechanical power as a cost function and softens the obstacle constraints, meaning collisions are still possible since constraints are softened. A non-optimizing approach for collision-free path following for redundant robot manipulators is presented in Zlajpah and Nemec

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.11.158

(2002) where a concept similar to repulsive potential fields is used on each obstacle to give the critical links (i.e. the closest to a collision) of the robot a velocity component away from the obstacle. It is assumed that the end-effector is not perturbed on this process, hence the need to have a redundant manipulator.

This paper is organized as follows: Section 2 presents the nonlinear model of the 2-DOF robotic manipulator and its velocity-based linearization. Section 3 introduces the velocity-algorithm for NMPC and the stability result. Section 4 reviews the qLMPC algorithm and presents the quasi-LPV representation of the considered nonlinear constraints. Section 5 presents the experimental results and Section 6 concludes the paper.

1.1 Notation

We denote a (block) diagonal matrix with elements $A, B, ...$ along the diagonal as $\text{diag}(A, B, ...)$. The *one vector* $[1\ 1\ \dots\ 1]^\top$ of length N is denoted as $\mathbf{1}_N$. We use the notation $||x||_Q^2$ to denote the weighted 2-norm, i.e. $|x|_Q^2 = x^\top Q x$. The Kronecker product of two matrices A and B is $A \otimes B$.

2. PLANT MODEL

2.1 Plant description

The robotic manipulator to be used is a CRS Model A465 shown in Figure 1. This is a 6 degree-of-freedom (DOF) robot, out of which we will only use the shoulder and the elbow (q_1 and q_2 in Figure 2). A nonlinear model of the 2-DOF manipulator can be obtained using the Euler-Lagrange formalism and is of the form

$$M(q(t))\ddot{q} + c(q, \dot{q}) + g(q(t)) + \tau_f(t) = \tau(t) \qquad (1)$$

where $q, \dot{q}, \ddot{q} \in \mathbb{R}^2$ are the joint angles, velocities and accelerations, respectively; M is the inertia matrix, c is the Coriolis force vector, g is the gravity vector, τ is the applied torque and τ_f is the friction torque which will be modeled solely as viscous. Note that the nonlinear model (1) is quite general and can be used to model almost any mechanical system. For the case of the 2-DOF robot, the model matrices are given by

$$M = \begin{bmatrix} b_1 + b_2 + 2b_3\cos(q_2) & b_2 + b_3\cos(q_2) \\ b_7 - b_8 + b_3\cos(q_2) & b_7 \end{bmatrix},$$
$$c = \begin{bmatrix} -b_3\dot{q}_2^2\sin(q_2) - 2b_3\dot{q}_1\dot{q}_2\sin(q_2) \\ b_3{\dot{q}_1}^2\sin(q_2) \end{bmatrix},$$
$$g = \begin{bmatrix} -b_4\sin(q_1+q_2) - b_5\sin(q_1) \\ -b_4\sin(q_1+q_2) \end{bmatrix}$$
$$\tau_f = \begin{bmatrix} b_6\dot{q}_1 \\ b_9\dot{q}_2 \end{bmatrix}$$

where parameters $b_1, ..., b_9$ are taken from Hashemi et al. (2012) where parameter identification of the same robot was reported, the parameter values are given in Table 1.

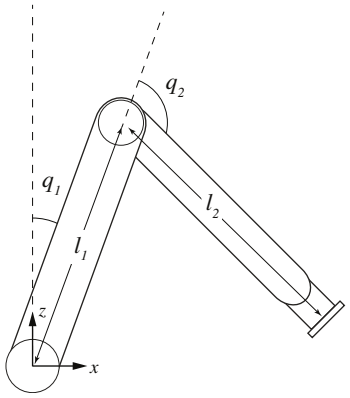

Fig. 2. Schematic of the 2-DOF robot

2.2 Velocity-based quasi-LPV model

There are several ways to obtain a quasi-LPV representation of the model (1), the so-called ad-hoc quasi-LPV parameterization that was used in Hashemi et al. (2012) and Cisneros and Werner (2017a) aims to directly hide nonlinearities of the nonlinear model under parameter variations; this might be difficult in some cases and could potentially lead to an uncontrollable quasi-LPV model for some parameterizations. In this paper we use a parameterization resulting from a velocity-based linearization (Leith and Leithead (1998)), this kind of parameterization, just as the ad-hoc parameterization, result in an *exact* representation of the nonlinear dynamics (1), as opposed to the approximate dynamics valid only around equilibrium points that would result from a Jacobian linearization. We next review the velocity-based linearization; assume a nonlinear model is given by

$$\dot{x} = f(x, u) \qquad (2)$$

a velocity-based linearization is obtained by differentiating the nonlinear model with respect to time, resulting in

$$\ddot{x} = \nabla_x f(x, u)\dot{x} + \nabla_u f(x, u)\dot{u} \qquad (3)$$

which is linear in the velocities $\dot{x}$, $\dot{u}$ and is scheduled by the operating point (x, u). One disadvantage of velocity-based quasi-LPV parameterization is that it often has a higher scheduling order than ad-hoc parameterization, which might be problematic if LMI-based synthesis techniques are used by gridding the parameter space. This is not done in this case and the scheduling order has no significant impact in computational complexity, as it entails only few more online operations to schedule $A(\rho_k)$ and $B(\rho_k)$. On the other hand, this parameterization has the advantage that it is systematic and simple in nature.

A velocity-based linearization of (1) is given by

Table 1. Estimated inertial parameters, Hashemi et al. (2012) and lengths (non-SI units)

Parameter	Value	Parameter	Value
b_1	0.0715	b_7	0.0749
b_2	0.0058	b_8	0.0705
b_3	0.0114	b_9	1.1261
b_4	0.3264	l_1	0.3048
b_5	0.3957	l_2	0.4064
b_6	0.6254		

$$\begin{bmatrix}\ddot{q}\\ \dot{\tau}\end{bmatrix} = \underbrace{\left[\begin{array}{cccc} 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ \hline a_{q_1}(\rho) & a_{q_2}(\rho) & a_{\dot{q}_1}(\rho) & a_{\dot{q}_2}(\rho)\end{array}\right]}_{\tilde{A}_c(\rho)} \underbrace{\begin{bmatrix}\dot{q}\\ \ddot{q}\end{bmatrix}}_{\dot{x}} + \underbrace{\left[\begin{array}{cc} 0 & 0\\ 0 & 0\\ \hline a_{\tau_1}(\rho) & a_{\tau_2}(\rho)\end{array}\right]}_{\tilde{B}_c(\rho)} \dot{\tau}$$

where the vectors $a_*(\rho)$ are given by

$$a_* = -\left(\frac{\partial M^{-1}}{\partial *}c + M^{-1}\frac{\partial c}{\partial *} + \frac{\partial M^{-1}}{\partial *}g + M^{-1}\frac{\partial g}{\partial *} + \frac{\partial M^{-1}}{\partial *}\tau_f + M^{-1}\frac{\partial \tau_f}{\partial *} - \frac{\partial M^{-1}}{\partial *}\tau - M^{-1}\frac{\partial \tau}{\partial *}\right)\Big|_{\rho}$$

and $\rho = [\rho_1\ \rho_2\ \rho_3\ \rho_4\ \rho_5\ \rho_6]^{\mathrm{T}} = [q_1\ q_2\ \dot{q}_1\ \dot{q}_2\ \tau_1\ \tau_2]^{\mathrm{T}}$.

We can define the tracking output to be $y = Cx = [q_1\ q_2]^{\mathrm{T}}$ and augment the state vector with the output to obtain the continuous-time LPV model

$$\begin{bmatrix}\dot{y}\\ \ddot{y}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & C\\ 0 & \tilde{A}_c(\rho)\end{bmatrix}}_{A_c(\rho)}\begin{bmatrix}y\\ \dot{x}\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ \tilde{B}_c(\rho)\end{bmatrix}}_{B_c(\rho)}\dot{\tau}. \tag{4}$$

Finally the model is discretized, for simplicity we choose Euler discretization for both the sate and the input, leading to the discrete-time LPV model

$$x_{a,k+1} = A(\rho_k)x_{a,k} + B(\rho_k)\Delta u_k \tag{5}$$

where $x_a = \begin{bmatrix}y^{\mathrm{T}} & \dot{x}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, $A(\rho_k) = I + T_s A_c(\rho_k)$, $B(\rho_k) = B_c(\rho_k)$ and $\Delta u_k = \Delta\tau_k = \tau_k - \tau_{k-1}$, and the sampling time $T_s = 0.01$s is chosen.

3. PREDICTIVE CONTROLLER

From the discussion in the preceding section it can be concluded that the use of velocity-based linearization results in a system in velocity-form. The theory and stability result of such a velocity algorithm for tracking of piece-wise continuous reference signals has been reported in Cisneros and Werner (2018), here we briefly recall the result.

Consider the following cost function:

$$J_k(y, y_{sp}, \dot{x}, \Delta u) = \sum_{i=0}^{N-1}\left(||y_i - y_{sp}||^2_{Q_1} + ||\dot{x}_i||^2_{Q_2} + ||\Delta u_i||^2_R\right) + ||y_N - y_{sp}||^2_T \tag{6}$$

where y_{sp} is the set point. The optimization problem to be solved online is given by

$$\min_{\Delta u} J_k(y, y_{sp}, \dot{x}, \Delta u) \tag{7a}$$
s.t.
$$\text{model (5)} \tag{7b}$$
$$y \in \mathcal{Y} \tag{7c}$$
$$u_i = u(k-1) + \Sigma_{j=0}^{i}\Delta u(j) \in \mathcal{U} \tag{7d}$$
$$\dot{x}(k+N) = 0 \tag{7e}$$

where $\mathcal{Y}$ and $\mathcal{U}$ are the constraint sets for output and input, respectively.

Theorem 1. (Cisneros and Werner (2018)). Assume the model $(A(\rho_k), B(\rho_k))$ is stabilizable $\forall\rho_k \in \mathcal{P}$, and let the offset penalty $T = \alpha Q_1$ for any $\alpha > 0$. Given a set point y_{sp}, the control law derived from the solution of optimization problem (7) starting from an admissible state x_a is stable, respects constraints and converges to at least a suboptimal set point.

Remark 1. The terminal constraint (7e) gives the velocity algorithm one of its strongest features. Instead of using terminal constraint on the positions, which might make the optimization problem infeasible if said set point is unreachable, the fact that all equillibria are mapped to the origin of the velocity states $\dot{x} = 0$, $\Delta u = 0$ is used to be able to guarantee stability even for unreachable set points, provided *any* equillibrium is reachable in N steps.

Remark 2. The same constraint (7e) can have adverse effects on performance if the horizon is short. To avoid this, an appropriately long horizon should be chosen so that the velocity at the initial step is not affected by the fact that it has to return to 0 at the end of the horizon [1]. Fortunately, the presented approach is efficient enough so a relatively long horizon can be used.

Problem (7) is a nonlinear optimization problem given the dependence of the scheduling trajectory on the state and input trajectories, solving such a complex problem online would likely preclude real-time implementation, for this reason we make use of the qLMPC algorithm (Cisneros et al. (2016)), which is briefly reviewed next.

4. qLMPC

Quasi-LPV MPC (qLMPC) is an iterative algorithm which solves, at each time instant, a series of QPs by freezing the scheduling trajectory

$$\mathsf{P}_k = \begin{bmatrix}\rho_k\\ \rho_{k+1}\\ \cdots\\ \rho_{k+N-1}\end{bmatrix} \tag{8}$$

to the one given by the previous solution, essentially turning the quasi-LPV model (5) into a Linear Time-Varying (LTV) model. Under these conditions, the constraint (7b) is linear and can be eliminated from the problem by substituting the future state trajectory into the cost function. The state trajectory is given by

$$X_k = \Lambda(\mathsf{P}_k)x_{a,k} + S(\mathsf{P}_k)\Delta U_k \tag{9}$$

where

$$\Lambda(\mathsf{P}_k) = \begin{bmatrix} A(\rho_k)\\ A(\rho_{k+1})A(\rho_k)\\ \vdots\\ A(\rho_{k+N-1})A(\rho_{k+N-2})\dots A(\rho_k)\end{bmatrix},$$

$$S(\mathsf{P}_k) = \begin{bmatrix} B(\rho_k) & 0 & \dots\\ A(\rho_{k+1})B(\rho_k) & B(\rho_{k+1}) & \dots\\ \vdots & \vdots & \dots\\ A(\rho_{k+N-1})\dots B(\rho_k) & A(\rho_{k+N-1})\dots A(\rho_{k+2})B(\rho_{k+1}) & \dots\end{bmatrix}$$

[1] The use of receding horizon means that the velocity will not actually reach 0 after N steps in closed-loop, unless the set points is reached

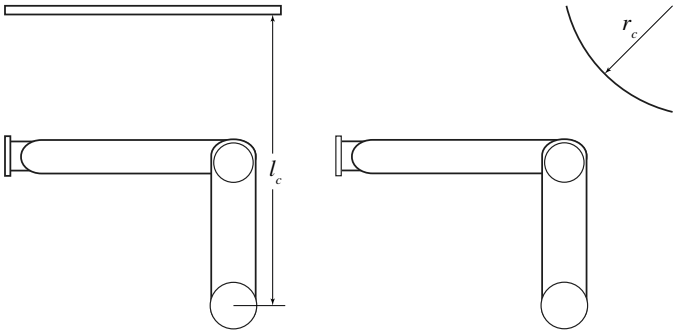

Fig. 3. Obstacles considered for the experiments

4.1 Nonlinear constraints

Similar to the nonlinear dynamics, nonlinear constraints of the form $h(y_k) \leq b$ can be expressed as quasi-LPV constraints (Cisneros and Werner (2017a)) by defining

$$h(y) = C_c(\rho_k^c)y \leq b(\rho_k^c)$$

where the scheduling vector ρ^c can be different from ρ. Here we consider two different output constraints, the first corresponding to a ceiling or horizontal wall above the robot (Figure 3 left) and the second a circle (Figure 3 right) which could be interpreted as an over-bounding of any arbitrary obstacle. For simplicity we assume that only the tip of the end effector needs to be bounded away from the obstacle (to consider the geometry of the robot, guaranteeing no collision would possibly require over-bounding of the obstacle). We derive now quasi-LPV representations of both via linearization.

Ceiling For this obstacle, the nonlinear constraint is

$$l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \leq l_c. \tag{10}$$

In Cisneros and Werner (2017a) it was shown how to turn a vertical wall into a quasi-LPV constraint using ad-hoc parameterization. In this case it is more difficult to find such a parameterization, because cosine terms do not lend themselves to a simple parameterization as sines do (using the substitution $\sin(x) = \text{sinc}(x)x$ to have a quasi-linear expression), for this reason we proceed by linearization. The constraint can be written in quasi-LPV form as

$$\begin{bmatrix} -l_2 \sin(\rho_1) - l_2 \sin(\rho_1 + \rho_2) & -l_2 \sin(\rho_1 + \rho_2) & 0 & 0 & 0 & 0 \end{bmatrix} x_a \leq$$
$$l_c - l_1 \cos(\rho_1) - l_2 \cos(\rho_1 + \rho_2) + (-l_2 \sin(\rho_1) - l_2 \sin(\rho_1 + \rho_2))\rho_1$$
$$-l_2 \sin(\rho_1 + \rho_2)\rho_2$$

where the same parameter vector ρ is used. Note that all constant terms (i.e. all which do not depend on the state) have been grouped on the right hand side of the inequality.

Circle This constraint is given by

$$-(l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) - c_x)^2 -$$
$$(l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) - c_y)^2 \leq -r_c^2, \tag{11}$$

where r_c is the radius and (c_x, c_y) is the center of the circle measured from the robot's base. After algebraic and trigonometric manipulation we arrive at

$$2c_x(l_1 \sin(q_1) + l_2 \sin(q_1 + q_2)) + 2c_y(l_1 \cos(q_1) + l_2 \cos(q_1 + q_2)) -$$
$$2l_1 l_2 \cos(q_2) \leq -r_c^2 + c_x^2 + c_y^2 + l_1^2 + l_2^2.$$

Proceeding as before with linearization we get

$$C_c(\rho) =$$
$$[2c_x(l_1 \cos(\rho_1) + l_2 \cos(\rho_1 + \rho_2) - 2c_y(l_1 \sin(\rho_1) + l_2 \sin(\rho_1 + \rho_2))$$
$$2l_1 l_2 \sin(\rho_2) + 2c_x l_2 \cos(\rho_1 + \rho_2) - 2c_y l_2 \sin(\rho_1 + \rho_2) \;\; 0 \;\; 0 \;\; 0 \;\; 0]$$

$$b(\rho) = -r_c^2 + c_x^2 + c_y^2 + l_1^2 + l_2^2 - 2l_1 l_2 \cos(\rho_2) +$$
$$2c_x(l_1 \sin(\rho_1) + l_2 \sin(\rho_1 + \rho_2)) + 2c_y(l_1 \cos(\rho_1) + l_2 \cos(\rho_1 + \rho_2)) +$$
$$C_c(\rho) \begin{bmatrix} \rho_1 & \rho_2 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$$

These output constraints can be turned into input constraints by using the prediction (9)

$$\hat{C}_c(\mathsf{P}_k^c)X_k = \hat{C}_c(\mathsf{P}_k^c)(\Lambda(\mathsf{P}_k)x_{a,k} + S(\mathsf{P}_k)\Delta U_k) \leq \mathbf{1} \otimes b$$

where $\hat{C}_c(\mathsf{P}_k^c) = \text{diag}(C_c(\rho_{k+1}), C_c(\rho_{k+2}), ..., C_c(\rho_{k+N}))$. We can then schedule both the dynamics and the constraints using the frozen parameter trajectory P_k, notice however, that the constraint is scheduled with a (subset of the) shifted parameter trajectory, i.e.

$$\mathsf{P}_k^c = \begin{bmatrix} \rho_{k+1} \\ \rho_{k+2} \\ \vdots \\ \rho_{k+N} \end{bmatrix}$$

cf. (8). The algorithm is summarized in Algorithm 1 [2].

Algorithm 1 qLMPC

Initialization: plant model, Q, R, N

1: $k \leftarrow 0$
2: Define $\mathsf{P}^0 = \mathbf{1}_N \otimes f(x_k, u_{k-1})$
3: **repeat**
4: $\quad l \leftarrow 0$
5: $\quad$ **repeat**
6: $\quad\quad$ Solve (7) using P_k^l to obtain U_k^l
7: $\quad\quad$ Predict state sequence $X_k^l = \Lambda(\mathsf{P}_k^l) + S(\mathsf{P}_k^l)U_k^l$
8: $\quad\quad$ Define $\mathsf{P}_k^{l+1} = H(X^l, U^l)$
9: $\quad\quad$ $l \leftarrow l + 1$
10: $\quad$ **until** stop criterion
11: $\quad$ Apply u_k to the system
12: $\quad$ Define $\mathsf{P}_{k+1}^0 = H(X_k^l, U_k^l)$
13: $\quad$ $k \leftarrow k + 1$
14: **until** end

4.2 Observer design

The prediction (9) is made assuming full state information. However, given that x_a contains the time derivative of all the states, assuming that every state of the augmented state vector is measured is not realistic. For this application, this means that we would need information of angles, velocities and accelerations; as velocities and accelerations are not available a state estimator is built using a quasi-LPV Kalman Filter. This filter is essentially an Extended Kalman Filter (EKF) but the prediction for both the state and the covariance is done using the velocity LPV model (5), as opposed to EKF where the state prediction is done using the nonlinear model (1) and the covariance prediction is done using the linearization of it. Nevertheless

[2] In line 8, 11, H maps the states and inputs to the parameters for the whole prediction horizon

the structure is the same, the quasi-LPV estimation can be summarized as follows (based on EKF from Simon (2006)):

Predict:

$$\hat{x}^a_{k|k-1} = A(\hat{\rho}_{k-1})\hat{x}^a_{k-1|k-1} + B(\hat{\rho}_{k-1})\Delta U_{k-1}$$
$$P_{k|k-1} = A(\hat{\rho}_{k-1})P_{k-1|k-1}A(\hat{\rho}_{k-1}) + Q_e$$

Update:

$$\tilde{y}_k = y_k - C\hat{x}_{k|k-1}$$
$$K_k = P_{k|k-1}C^{\mathrm{T}}(CP_{k|k-1}C^{\mathrm{T}} + R_e)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_k$$
$$P_{k|k} = (I - K_kC)P_{k|k-1}$$

where Q_e, R_e are the process and measurement noise covariances, often used as tuning parameters for estimation problems. In this case they are set to $Q_e = I_6$, $R_e = 0.001I_2$, given that the encoders that measure joint angles are relatively noise-free. Note that this observer uses a simple 1-step-ahead prediction and the usual correction step (i.e. it is not a moving-horizon estimator).

5. EXPERIMENTAL RESULTS

Both constraint scenarios were tested experimentally. For both the task is to reach the point $y_{sp} = [q_{1,sp}\ q_{2,sp}]^{\mathrm{T}} = [0\ \pi/2]^{\mathrm{T}}$ starting from the initial condition $y_0 = [0\ -\pi/2]$, a prediction horizon of $N = 15$ is used. Intuitively, the optimal solution would be to move the second link while keeping the first link on its initial position. However, in the presence of constraints this is not possible, so the first link needs to be moved to allow the second link to reach its desired position.

Ceiling For this obstacle, the weighting matrices are chosen as $Q_1 = \mathrm{diag}(1, 2)$, $Q_2 = \mathrm{diag}(0.5, 0.5, 0, 0)$, $R = I$,

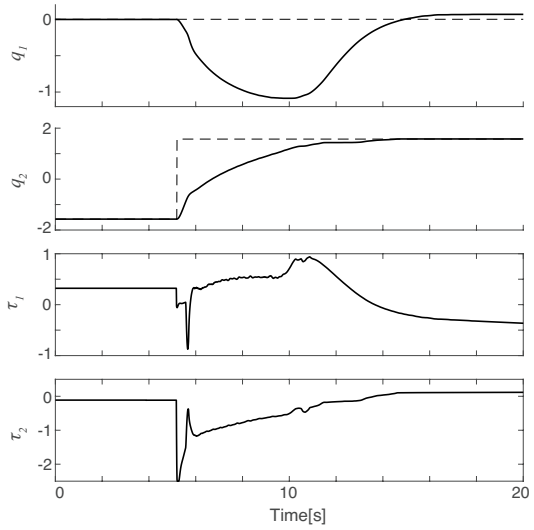

Fig. 4. Time domain plots of experiment with ceiling obstacle

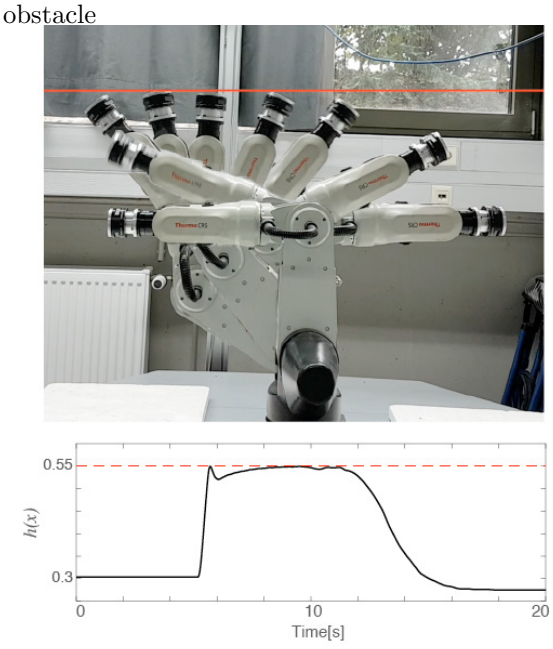

Fig. 5. Robot motion during the experiment (top) and evaluation of the ceiling constraint (bottom)

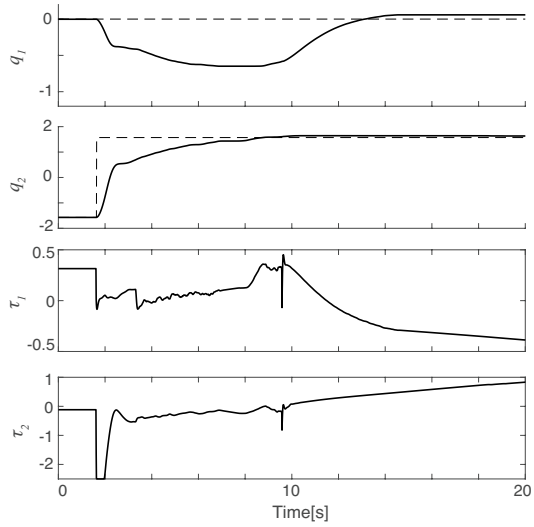

Fig. 6. Time domain plots of experiment with circle obstacle

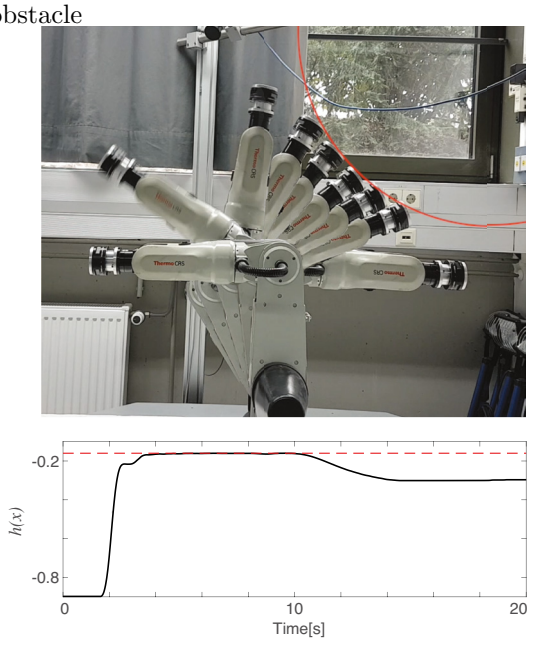

Fig. 7. Robot motion during the experiment (top) and evaluation of the circle constraint (bottom)

$P = 2Q$. The value of R is chosen as such to avoid rapid changes in the input which might induce undesired vibrations (recall that unlike the usual case, here R punishes Δu, whereas u is not directly punished). The time domain plots of the experiment are shown in Figure 4 whereas the evolution of the position of the robot at several time instants along with the evaluation of constraint (10) at each time instant is shown in Figure 5.

Circle In this case, the weighting matrices are slightly retuned to $Q_1 = \text{diag}(1, 4)$, $Q_2 = \text{diag}(0.5, 0.5, 0, 0)$, $R = I$, $P = 2Q$, this was done to obtain a smoother response, since Coulomb friction caused the robot the stop at some points using the previous tuning. The time domain plots of the experiment are shown in Figure 6 whereas the evolution of the position of the robot at several time instants along with the evaluation of constraint (11) at each time instant is shown in Figure 7.

In both cases, it can be observed that the control input at the end of the experiment is not stationary. This is a consequence of the non-zero steady-state error caused by a very high Coulomb friction force (stiction); as the velocity-based scheme has integral action the input is increased in order to overcome this effect and ultimately reach zero steady-state error.

Video footage of the experiment can be found at https://youtu.be/IdqJsx-ZJxQ, where experiments with both obstacles are shown.

6. CONCLUSION

The use of qLMPC opens the door of NMPC to fast systems as is demonstrated in this paper: by scheduling the model matrices online, one can consider highly dynamic nonlinear systems while keeping online computation complexity comparable to that of linear MPC. The main features are: guaranteed convergence at least to a suboptimal set point, even for unreachable desired set points; computational efficiency, ease of implementation, since it is similar to standard MPC for LTI systems. The obstacle scenarios examined here are relatively simple since the only critical point considered is the tip of the end effector, but generalizations follow straight-forwardly provided one finds a way to express the constraint as $h(y_k) < b$, which is a very general form.

In the experimental results it was seen that the Coulomb friction, neglected in the model, has a strong impact on the closed-loop response. The inherent robustness and the built-in integral action of the velocity algorithm were able to overcome this unmodelled effect and perform the task successfully.

REFERENCES

Cisneros, P.S.G., Voss, S., and Werner, H. (2016). Efficient nonlinear model predictive control via quasi-lpv representation. *55th IEEE Conference in Decision and Control*.

Cisneros, P.S.G. and Werner, H. (2017a). Fast nonlinear mpc for reference tracking subject to nonlinear constraints via quasi-lpv representations. *20th IFAC World Congress*.

Cisneros, P.S.G. and Werner, H. (2017b). Parameter dependent stability conditions for quasi-lpv model predictive control. *American Control Conference*.

Cisneros, P.S.G. and Werner, H. (2018). A velocity algorithm for nonlinear model predictive control. *Submitted to 57th IEEE Conference in Decision and Control*. Available online: `https://www.tuhh.de/t3resources/ics/user_upload/CiWe18.pdf`.

Diehl, M., Bock, H., and Schlöder, J. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736.

dos Santos, R.R., Steffen, V., and Saramago, S.d.F.P. (2008). Robot path planning in a constrained workspace by using optimal control techniques. *Multibody System Dynamics*, 19(1), 159–177. doi:10.1007/s11044-007-9059-1. URL `https://doi.org/10.1007/s11044-007-9059-1`.

Hashemi, S.M., Abbas, H.S., and Werner, H. (2012). Low-complexity linear parameter-varying modeling and control of a robotic manipulator. *Control Engineering Practice*, 20, 248–257.

Leith, D.J. and Leithead, W.E. (1998). Gain-scheduled and nonlinear systems: dynamic analysis by velocity-based linearization families. *International Journal of Control*, 70(2), 289–317.

Simon, D. (2006). *Optimal State Estimation*. John Wiley & Sons.

Zlajpah, L. and Nemec, B. (2002). Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, 1898–1903 vol.2. doi:10.1109/IRDS.2002.1044033.