Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 11601–11606

Fast Nonlinear MPC for Reference Tracking Subject to Nonlinear Constraints via Quasi-LPV Representations

Pablo Gonzalez Cisneros * **Herbert Werner** **

* *Institute of Control Systems, Hamburg University of Technology Hamburg, Germany (e-mail: pablo.gonzalez@tuhh.de).*
** *Institute of Control Systems, Hamburg University of Technology Hamburg, Germany (e-mail: h.werner@tuhh.de)*

Abstract: This paper presents an approach to efficiently implement Nonlinear Model Predictive Control (NMPC) for reference tracking in the presence of nonlinear input and state constraints by making use of quasi-Linear Parameter Varying (quasi-LPV) representations. Using this framework, standard Quadratic Program (QP) solvers can be used for the online optimization problem, making its solution very efficient and viable even for fast plants. This is an extension of a previous result which considered the regulator problem with input constraints. This approach is tested in a simulation study of a 2-DOF robotic manipulator and its efficiency is compared to that of state-of-the-art NMPC approaches.

© 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Model predictive control; linear parameter varying systems; nonlinear systems; constrained control; efficient algorithms.

1. INTRODUCTION

Constraints in control and system theory are ubiquitous, be it actuator saturation or physical limitations on state variables, all control systems have limits to an extent. In many applications, one can circumvent these limitations simply by properly tuning a controller for the expected operating conditions. However, this approach may prove overly naive if the system is driven outside these operating conditions or more prominently, when the system has state constraints. Model Predictive Control (MPC) is one of the few control strategies that systematically and optimally deals with constraints. Given that the control law is determined by an online optimization of a suitable cost function, inclusion of both input and state constraints is straightforward.

Most of the MPC literature considers the regulator problem, and conditions for asymptotic stability for both linear and nonlinear case are given for the origin. The tracking problem is in many cases discussed practically, e.g. in Rossiter (2003), but there are also many theoretical extensions to the stability conditions for the tracking problem. One of the earlier results, Gilbert and Kolmanovsky (1995), proposes the use of reference governors to deal with constraints in controlled linear systems. These higher level controllers function similarly to MPC but their goal is to modify the reference such that it fulfills constraints; a low-level controller then steers the plant to this modified reference. Since then, this approach has been extended to nonlinear systems (see e.g. Bemporad (1998) and Gilbert and Kolmanovsky (2002)). In Ferramosca et al. (2009) the tracking problem is solved by creating an artificial steady state as decision variable and penalizing both the deviation of the real state with respect to the artificial steady state as well as the deviation of the artificial steady state with respect to the target state. The standard conditions for MPC stability, namely terminal cost and terminal constraint set are extended to the tracking problem and several suggestions for computing them are disclosed. A quasi-Linear Parameter Varying (quasi-LPV) representation for a particular kind of nonlinear systems is used in Chisci et al. (2005) to compute terminal regions based on a partition of the admissible set. A Dual-Mode control is employed in which mode 1 functions as a regulator around the desired equilibrium and mode 2 drives the system towards this terminal region when a setpoint change occurs. MPC for LPV systems has been explored, e.g. in Kim et al. (2006) and Abbas et al. (2015), where worst case optimization is carried out online to account for the uncertainty in the parameter trajectory.

This paper takes a practical approach and, without giving stability guarantees, proposes a very easy to implement and more importantly extremely time efficient algorithm for tracking of reference setpoints for nonlinear systems with input and state constraints. The quasi-LPV framework is used in order to turn the nonlinear optimization problem into a standard Quadratic Program (QP) even when nonlinear state constraints are present. The algorithm can also be used in conjunction with the conditions developed in Ferramosca et al. (2009) or Chisci et al. (2005) to give strict convergence guarantees.

This paper is organized as follows: section 2 sets up the general framework. Section 3 elaborates on how the tracking problem is practically implemented. In section 4 we present how linear or nonlinear state constraints can be included in the problem and section 5 illustrates the approach in a simulation example of a 2-DOF robotic

2405-8963 © 2017, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.1650

manipulator. Finally, section 6 gives some concluding remarks.

1.1 Notation

We denote a block diagonal matrix with matrices M repeated N times along the diagonal as $\text{diag}_N(M)$, and a diagonal matrix with elements $i, j, k, ...$ along the diagonal as $\text{diag}(i, j, k, ...)$. The *one vector* $[1\ 1\ \dots\ 1]^\top$ is denoted as $\mathbf{1}$ with dimension omitted when obvious from context. The notation $M \prec 0$ ($M \preceq 0$) means that M is negative (semi-) definite. 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 $A \otimes B$. For notation compactness parameter dependency of a matrix $M(\rho)$ is sometimes denoted as M_ρ; similarly, matrix dependency on a time-shifted parameter $M(\rho_{k+i})$ is sometimes denoted as M_{k+i}.

2. PROBLEM SETUP

We consider the quasi-LPV model

$$\begin{aligned} x_{k+1} &= A(\rho_k)\, x_k + B(\rho_k)\, u_k \\ y_k &= C(\rho_k)\, x_k \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^l$, $\rho_k = f(x_k, u_k) \in \mathbb{R}^{n_\rho}$, and $A : \mathbb{R}^{n_\rho} \to \mathbb{R}^{n\times n}$, $B : \mathbb{R}^{n_\rho} \to \mathbb{R}^{n\times m}$, $C : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times n}$, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_\rho}$ are continuous maps. In this case y_k is not the measured output, as we deal with state feedback, but the tracking output. We assume that $\rho_k \in \mathcal{P}\ \forall k \geq 0$ where $\mathcal{P} \in \mathbb{R}^{n_\rho}$ is a given compact set. This implies that A, B and C are bounded on $\mathcal{P}$. Throughout this paper we will also assume that $(A(\rho), B(\rho))$ is stabilizable and $(A(\rho), C(\rho))$ is detectable $\forall \rho \in \mathcal{P}$.

The focus is on a predictive control scheme based on minimizing the finite horizon cost

$$J_k = \sum_{l=1}^{N} |x_{k+l} - x_{ss}|_Q^2 + |u_{k+l-1} - u_{ss}|_R^2 + \Psi(x_{k+N} - x_{ss}) \tag{2}$$

where $Q = Q^\top \succeq 0$, $R = R^\top \succ 0$ and $\Psi(x) > 0\ \forall\ x \neq 0$, $\Psi(0) = 0$. The values x_{ss} and u_{ss} are constant and represent the steady state value of state and input, respectively. At each sampling instant k, the values of x_k and u_{k-1} are known, and the optimization problem

$$\begin{aligned} &\min_{U_k} J_k(U_k) \\ \text{subject to} \quad &U_k \in \mathcal{U}, \quad \tilde{x}_{k+N} \in \mathcal{X} \quad x_k \in X_{cons} \end{aligned} \tag{3}$$

is solved online for

$$U_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix} \in \mathbb{R}^{Nm}, \tag{4}$$

where $\mathcal{U}$ is a constraint set for the input, $\mathcal{X}$ a constraint set for the terminal state deviation $\tilde{x}(k+N) = x(k+N) - x_{ss}$ and X_{cons} represents the set of admissible state values given by the state constraints. We assume that $f(\mathcal{X} \times \mathcal{U}) \subset \mathcal{P}$. The control law is implemented in a receding horizon fashion, i.e., at time k control input u_k is applied, whereas at time $k+1$ the problem min J_{k+1} is solved for U_{k+1} and the newly calculated control input u_{k+1} is applied etc. We also define

$$X_k = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+N} \end{bmatrix} \in \mathbb{R}^{Nn}, \qquad \mathsf{P}_k = \begin{bmatrix} \rho_k \\ \rho_{k+1} \\ \vdots \\ \rho_{k+N-1} \end{bmatrix} \in \mathbb{R}^{Nn_\rho}, \tag{5}$$

and with a slight abuse of notation use f to denote $\mathsf{P}_k = f(\begin{bmatrix} x_k^\top & X_k^\top \end{bmatrix}^\top, U_k)$. Note that the map f is not directly applied to X_k but to $\begin{bmatrix} x_k^\top & X_k^\top \end{bmatrix}^\top$, as the vector P_k contains the parameters from time k to $k+N-1$ whereas the prediction is done for time $k+1$ to $k+N$.

Note that for a general LPV system (where ρ_k is independent of x_k and u_k) it is not possible to solve the optimization problem (3), because the future state sequence cannot be predicted: X_k depends not only on the future control inputs U_k (the decision variables), but also on the future scheduling parameters P_k, which for a general LPV system are not assumed to be known *a priori* but only to be measurable online. Such system can be tackled by worst-case optimization as presented in Abbas et al. (2015). In contrast, for a *quasi-LPV system*, where the scheduling parameters ρ_k are determined by x_k and u_k, the state trajectory can be predicted.

A method (henceforth referred to as qLMPC or quasi-Linear Parameter Varying Model Predictive Control) is presented in Cisneros et al. (2016) to solve (3), for the regulator problem, iteratively as a series of Quadratic Programs (QP) or Second Order Cone Programs (depending on the desire to have strict stability guarantee) by making use of the LPV framework and exploiting the functional dependency of the scheduling parameter and the states/inputs. We can rewrite (2) as

$$\begin{aligned} J_k = |H(\mathsf{P}_k)x_k + S(\mathsf{P}_k)U_k - X_{ss}|_{\tilde{Q}}^2 + |U_k - U_{ss}|_{\tilde{R}}^2 \\ +\Psi(x_{k+N} - x_{ss}). \end{aligned} \tag{6}$$

where

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

$$S(\mathsf{P}_k) = \begin{bmatrix} B_k & 0 & \dots \\ A_{k+1}B_k & B_{k+1} & \dots \\ \vdots & \vdots & \dots \\ A_{k+N-1}\dots A_{k+1}B_k & A_{k+N-1}\dots A_{k+2}B_{k+1} & \dots \end{bmatrix},$$

$$\tilde{Q} = \text{diag}_N(Q), \qquad \tilde{R} = \text{diag}_N(R),$$

so that the predicted trajectory X_k is given by

$$X_k = H(\mathsf{P}_k)x_k + S(\mathsf{P}_k)U_k. \tag{7}$$

Note that given the parameter sequence $\mathsf{P}(k)$ the cost function (6) is equivalent to (2), however, this sequence

depends on the predicted state sequence. The solution is then found iteratively as described in Cisneros et al. (2016) and summarized in Algorithm 1 below. The algorithm is initialized (e.g. at $k = 0$) by assuming the parameters to be frozen at their value at k; for the subsequent time steps the parameter trajectory is initialized from the prediction at the previous time step, taking care of the time-shift that occurs. For this reason the parameter trajectory at time $k + 1$ is initiliazed as $\mathsf{P}^0_{k+1} = f(X_k, U_k)$, whereas the parameter trajectory at time step k is given by $\mathsf{P}_k = f([x_k^\top \ X_k^\top]^\top, U_k)$ (cf. (5) and the discussion that follows).

Algorithm 1 qLMPC

Initialization: reference trajectory, plant model, constants $\tilde{Q}$, $\tilde{R}$, N

$k \leftarrow 0$
Define $\mathsf{P}^0 = \mathbf{1} \otimes f(x_k, u_{k-1})$
repeat
  $l \leftarrow 1$
  repeat
    Solve (3) using P^l_k for U^l_k
    Use (7) to calculate X^l_k using P^l_k and U^l_k
    Define $\mathsf{P}^{l+1}_k = f([x_k^\top \ X_k^{l\top}]^\top, U^l_k)$
    $l \leftarrow l + 1$
  until stop criterion
  Apply u_k to the system
  Define $\mathsf{P}^0_{k+1} = f(X^l_k, U^l_k)$
  $k \leftarrow k + 1$
until end

3. THE TRACKING PROBLEM

The cost function in the previous section is given such that deviations from the steady state values of inputs and states are penalized. However, in the general case one does not wish to provide a reference to the state, but rather a subset of the states which we denoted by y_k. As we need information on the steady state values for all the states as well as for the input, we make use of a target selector, as suggested in Rawlings and Mayne (2009), whose job is to find steady state values for states and inputs in the presence of constraints. Note that this is different from a reference governor in that its job is only to compute steady state values, not a trajectory, such that constrains are fulfilled.

The target values are the solutions of the optimization problem

$$\begin{aligned} &\min_{x_s,u_s} |u_s - u_{sp}|^2_{R_s} + |C(\rho_s)x_s - y_{sp}|^2_{Q_s} \qquad (8)\\ &\text{s.t.}\\ &\begin{bmatrix} I - A(\rho_s) & -B(\rho_s) \\ C(\rho_s) & 0 \end{bmatrix} \begin{bmatrix} x_s \\ u_s \end{bmatrix} = \begin{bmatrix} 0 \\ y_{sp} \end{bmatrix}\\ &u_s \in \mathcal{U}, \qquad x_s \in X_{cons} \end{aligned}$$

where u_{sp} and y_{sp} are setpoint values for the input and tracking output, respectively and $\rho_s = f(x_s, u_s)$. If information about the input reference is not available, the setpoint can be simply set to $u_{sp} = 0$ to choose the minimum-norm input. The weighting R_s must be positive definite and Q_s positive semi-definite. The parameter value ρ_s imposes the same limitation as for the MPC problem: the solution depends on the parameter value, which is given in turn by the solution. The same iterative solution method as the one proposed for MPC in Cisneros et al. (2016) can be used for the target selector, where due to the small number of decision variables a faster convergence is expected. This optimization problem can be solved offline if the tracking reference is known a priori to compute the state and input reference. Otherwise, it can be solved online every time a setpoint change takes place to calculate the new state and input reference.

It is important to note that in order for the solution to exist it must hold that $l \leq m$, that is, the number of tracking outputs must be less than or at most equal to the number of inputs. Integral action can be added by state augmentation either by considering input increments, Δu or by adding an integrating disturbance model. Both these methods are standard and well known and are not discussed here, see e.g. Rossiter (2003), Rawlings and Mayne (2009).

One crucial difference between tracking and regulation is that the stability conditions need to be modified. Usual stability proofs (see e.g. Michalska and Mayne (1993), Mayne et al. (2000)) rely on a terminal constraint set and a terminal cost that makes the cost function a Lyapunov function. For the tracking case appropriate modifications need to be made to keep the stability guarantee. The most straightforward, yet very conservative, choice is to impose a terminal equality constraint $x_N = x_s$, i.e. reduce the terminal set to a point. This constraint would likely need a very long prediction horizon to make the problem feasible. As for the terminal cost, a pragmatic choice is simply to take the one calculated for the regulator problem: define $\Psi(x_N) = x_N^\top P x_N$ where P is the solution to

$$\begin{aligned} (A_{\rho_k} + B_{\rho_k}F_{\rho_k})^\top P_{\rho_{k+1}} (A_{\rho_k} + B_{\rho_k}F_{\rho_k}) - P_{\rho_k} \\ \preceq -Q - F_{\rho_k}^\top R F_{\rho_k} \qquad \forall \rho \in \mathcal{P}. \end{aligned}$$

and F is the dual-mode state feedback controller that takes the system to the origin from any point within a certain terminal region with cost $x^\top P x$. Parameter dependency on P and F can be dropped if the system is quadratically stabilizable, i.e. if there exists a constant P such that the above condition is met $\forall \rho \in \mathcal{P}$, as presented in Cisneros and Werner (2017), otherwise a parameter dependent Lyapunov function and state feedback gain can be used.

4. STATE CONSTRAINTS

The cost function as posed in (6) does not include the state sequence as variables. The predicted state was eliminated from the cost function by making the substitution $X_k = H(\mathsf{P}_k)x_k + S(\mathsf{P}_k)U_k$. Using this same substitution, state constraints can be included in the optimization problem, these will however be converted into additional constraints in the input trajectory U_k.

We next define the *constrained output* $y_c(k) = h(x_k) = C_c(\rho_k)x_k$; note that this is in the general case different from the tracking output in (1), although it might coincide for several applications. The predicted constrained output is thus

$$Y_c(k) = \bar{C}_c(\mathsf{P}_k^c)X_k = \bar{C}_c(\mathsf{P}_k^c)(H(\mathsf{P}_k)x_k + S(\mathsf{P}_k)U_k)$$

where $\bar{C}_c(\mathsf{P}_k^c) = \text{diag}(C_c(\rho_{k+1}), C_c(\rho_{k+2}), ..., C_c(\rho_{k+N}))$, and the parameter vector $\mathsf{P}_k^c = f(X_k) = [\rho_{k+1}\ \rho_{k+2}\ \cdots\ \rho_{k_N}]$ is used to schedule the constraint (cf. (5)) . Given a constraint $y_c(k+i) \leq b_c$ where b_c is constant, we can use the definitions above to constrain the predicted state and by extension the input trajectory as

$$C_c(\mathsf{P}_k^c)S(\mathsf{P}_k)U_k \leq \mathbf{1}b_c - C_c(\mathsf{P}_k^c)H(\mathsf{P}_k)x_k.$$

Note that the constraint need not be linear; nonlinear constraints can be accommodated by finding a quasi-LPV representation of $h(x_k)$ and making C_c parameter dependent. By using the predicted parameter trajectory P_k^c this constraint becomes linear and is written in the standard form $Ax \leq b$, which does not increase complexity for the solution of the QP or SOCP.

5. SIMULATION EXAMPLE

The presented algorithm was tested in simulation on a nonlinear 2-DOF robotic manipulator model (Figure 1). The continuous-time nonlinear model and inertial parameters are taken from Hashemi et al. (2012) and are given by

$$\tau(t) = M(q(t))\ddot{q}(t) + c(q(t), \dot{q}(t)) + g(q(t)) \qquad (9)$$

where $q(t) = [q_1(t)\ q_2(t)]^\top$, $\tau(t) = [\tau_1(t)\ \tau_2(t)]^\top$,

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

and Table 1, respectively. Sampling time is set to $T_s = 0.01$s and the prediction horizon is set to $N = 25$.

Table 1. Estimated inertial parameters, Hashemi et al. (2012) and lengths

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 m
b_5	0.3957	l_2	0.4064 m
b_6	0.6254	l_c	0.2 m

We proceed by finding a quasi-LPV representation for the model. Coriolis terms are neglected for the sake of simplicity (results show that there is no noticeable difference by considering them), choosing the parameter vector $\rho = [q_1\ q_2]^\top$ and making the substitution $\sin(q_i) = \text{sinc}(q_i)q_i$ we can write the system in the form

$$\tau(t) = M(\rho)\ddot{q}(t) + c(\rho)\dot{q}(t) + g(\rho)q(t)$$

(note that both $c(\rho)$ and $g(\rho)$ are square matrices) for which a quasi-LPV model is given by

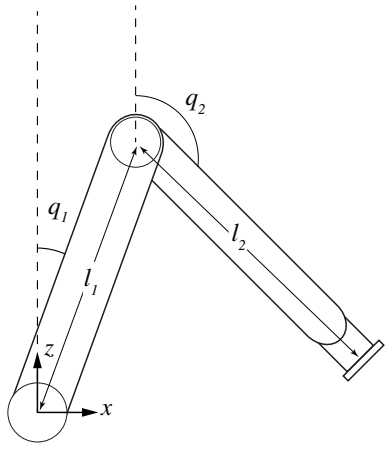

Fig. 1. 2-DOF robotic manipulator.

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I \\ -M_\rho^{-1}g_\rho & -M_\rho^{-1}c_\rho \end{bmatrix}}_{A(\rho)} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -M_\rho^{-1} \end{bmatrix}}_{B(\rho)} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}.$$

The next step is to discretize the model; if sampling is fast enough, an Euler discretization can be used, however, as is well known, in MPC the sampling time should be kept as long as possible to avoid a long prediction horizon. For this reason a Runge-Kutta 4 discretization is preferred, which can be obtained using the expressions

$$A_d(\rho) = \frac{1}{24}A_\rho^4T_s^4 + \frac{1}{6}A_\rho^3T_s^3 + \frac{1}{2}A_\rho^2T_s^2 + A_\rho Ts + I$$
$$B_d(\rho) = \frac{1}{24}A_\rho^3B_\rho T_s^4 + \frac{1}{6}A_\rho^2B_\rho T_s^3 + \frac{1}{2}A_\rho B_\rho T_s^2 + B_\rho T_s.$$

We define the tracking output as $y(t) = [q_1\ q_2]^\top$ which means $C_y = [\text{I}_2\ 0]$. Instead of imposing known limits to the angles (which can be simply done by giving reasonable reference values) we consider the more interesting problem of avoiding a physical obstacle. Assume the robotic manipulator is installed in close proximity to a wall. We want, without complex trajectory planning, to go from configuration A in Figure 2a) to configuration B, which correspond to $x_A = [0.2\quad 0.2\quad 0\quad 0]$ and $x_B = [0.5\quad \pi\quad 0\quad 0]$. Intuitively, in the absence of state constraints, the optimal trajectory would entail a simultaneous movement in both degrees of freedom toward the target; this is of course not possible in the presence of the described constraint. In order to include this constraint, we must encode it in the constrained output matrix $C_c(\rho)$. For the wall obstacle, the non-linear constraint is

$$l_1\sin(q_1) + l_2\sin(q_2) < l_c,$$

using the same substitution as before, we define

$$C_c(\rho) = [l_1\text{sinc}(\rho_1)\ l_2\text{sinc}(\rho_2)\ 0\ 0];$$

we also consider a constraint in the input such that $u_k \in [-10\quad 10]$ Nm. Weighting matrices are chosen as $Q_s = Q = \text{diag}(1, 1, 0.01, 0.01)$ and $R_s = R = 0.001I_2$, the terminal cost weight P was found offline using the LMI approach from Cisneros et al. (2016).

The evolution of the system is shown for several time instants in Figure 2b). We can see that the optimal trajectory makes the end effector *slide* on the constraint,

Table 2. Algorithm times [1]

	Max time	Mean time	St. dev.	Solver time
qLMPC	4.33 ms	1.04 ms	1.21 ms	0.71 ms
ACADO	4.54 ms	0.86 ms	0.63 ms	0.77 ms
CasADi*	225.66 ms	100.06 ms	49.81 ms	187.42 ms

*Only up until infeasibility appeared

to accomplish this, the first link has to first move in the opposite direction to fulfill the constraint.

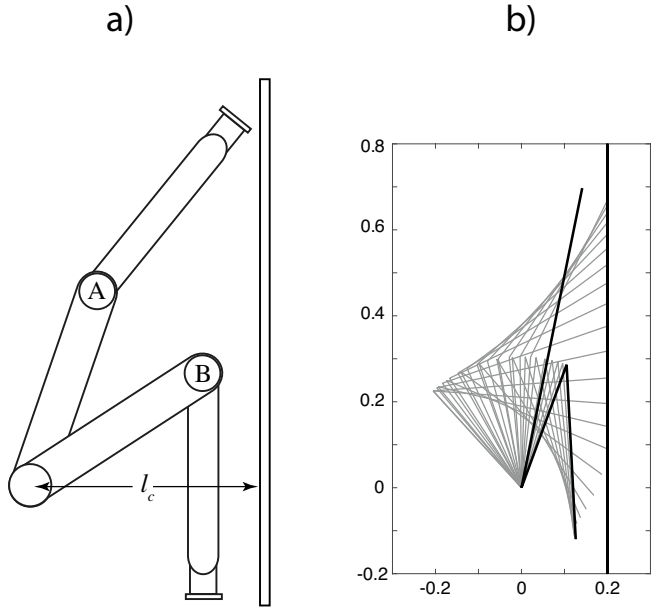

Fig. 2. a) Tracking set-up. b) Robot trajectory. The black lines represent initial and final states.

The presented approach was also compared with a standard NMPC algorithm based on multiple shooting with the nonlinear solver *IPOPT* using *CasADi* (Andersson (2013)) and with ACADO (Ariens et al. (2010)) both in terms of performance and numerical efficiency. The qLMPC algorithm was implemented using the QP solver *QPOASES* (Ferreau et al. (2014)). The time response of these methods is shown in Figure 3 where it can be seen that the method that relies in nonlinear optimization (CasADi with IPOPT) runs into numerical issues leading to an *infeasible problem*, whereas both the proposed method and ACADO, which rely on a QP approximation of the nonlinear optimization, successfully complete the task. A comparison of the numerical efficiency, as measured by the time to find a solution (up to the point in which CasADi runs into numerical issues,) can be seen in Table 2. The proposed algorithm and ACADO have very similar results, with the worst-case execution time favoring the proposed approach but the average time favoring ACADO; intuitively both algorithms are orders of magnitude faster than one that relies on general nonlinear optimization. It is worth noting that while ACADO generates a highly optimized C++ code tailored for the problem, the presented approach is a prototype, meaning that some optimization can still take place to make its execution faster (as can be seen by the relatively large gap between solver time and total execution time).

[1] Tested on a 4.0GHz Intel Core-i7 6700K.

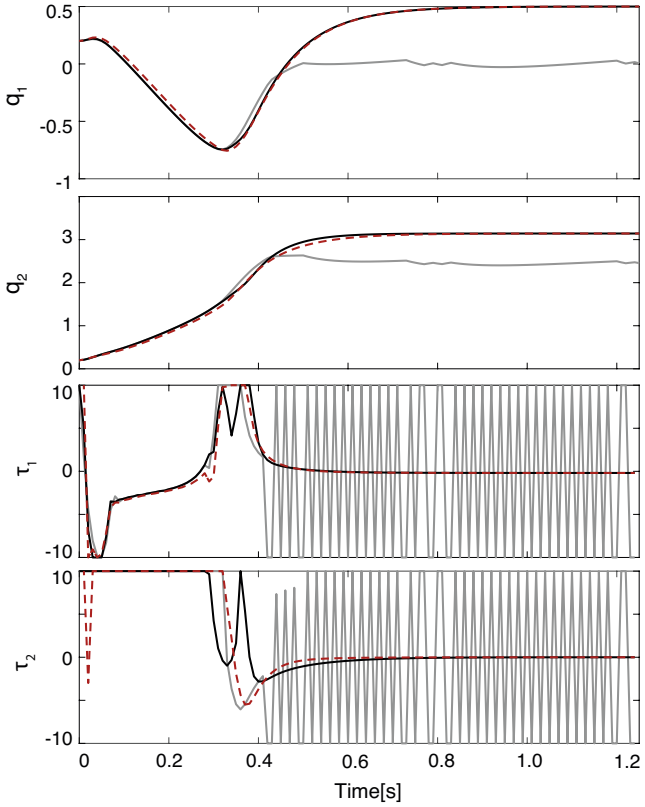

Fig. 3. Comparison of proposed qLMPC (——) multiple shooting based on ACADO (- - - -) and CasADi (——).

This example was carried out using 1 iteration (i.e., 2 QP solutions every time step) of Algorithm 1 and with no stability guarantees: no terminal constraint was used and the terminal cost was the one of the regulator problem. It is clear that using a terminal equality constraint would not be advisable as the prediction horizon would need to be very long (around 80-100 steps for this example).

6. CONCLUSION

This paper addressed the problem of tracking in NMPC subject to nonlinear state constraints by introducing parameter dependent constraints, thereby extending the previous work on NMPC via a quasi-LPV representation of the cost function. It was seen how, without stability guarantees, the problem can be very efficiently solved by a series of QPs, meaning that its complexity is comparable to that of Linear MPC. Stability guarantees can be obtained by making use of the results proposed by Ferramosca et al. (2009) or Chisci et al. (2005).

The quasi-LPV framework is flexible enough for this approach to be used for a wide variety of nonlinear systems and its efficiency translates in applicability to fast systems like robotic manipulators. In a simulation example it was shown that rather than using complex trajectory planners for a robotic manipulator, the presented approach is able to deal with obstacles, provided one finds a way of encoding the nonlinear constraint as a quasi-LPV one. The example also served to highlight the numerical efficiency of the algorithm as it proved to be similar to a state-of-

the-art approximation approach and 2 orders of magnitude faster than nonlinear optimization based NMPC.

REFERENCES

Abbas, H.S., Toth, R., Meskin, N., Mahammadpour, J., and Hanema, J. (2015). A model predictive control design for linear parameter-varying systems in input-output form. *IEEE Conference on Decision and Control*, 54, 91–96.

Andersson, J. (2013). *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Ariens, D., Houska, B., Ferreau, H., and Logist, F. (2010). *ACADO for Matlab Users Manual*. Optimization in Engineering Center (OPTEC), 1.0beta edition. `http://www.acadotoolkit.org/`.

Bemporad, A. (1998). Reference governor for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 43(3), 415–419.

Chisci, L., Falugi, P., and Zappa, G. (2005). Predictive tracking control of constrained nonlinear systems. *Proc.-Control Theory and Applications*, 152(3).

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*. `https://www.tuhh.de/t3resources/rts/user_upload/CiVoWe16.pdf`.

Cisneros, P.S.G. and Werner, H. (2017). Parameter dependent stability conditions for quasi-lpv model predictive control. *Submitted to American Control Conference, 2017*. `https://www.tuhh.de/t3resources/rts/user_upload/CiWe17.pdf`.

Ferramosca, A., Limon, D., Alvarado, I., Alamo, T., and Camacho, E. (2009). Mpc for tracking of constrained nonlinear systems. *48th IEEE Conference on Decision and Control*.

Ferreau, H., Kirches, C., Potschka, A., Bock, H., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4), 327–363.

Gilbert, E.G. and Kolmanovsky, I. (1995). Discrete-time reference governors and the control of systems with state and control constraints. *International Journal of robust and nonlinear control*, 5, 487–504.

Gilbert, E.G. and Kolmanovsky, I. (2002). Nonlinear tracking control in the presence of state and control constraints: a generalized reference governor. *Automatica*, 38, 2063–2073.

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.

Kim, T.H., Park, J.H., and Sugie, T. (2006). Output-feedback model predictive control for lpv systems with input saturation based on quasi-min-max algorithm. *IEEE Conference on Decision and Control*, 45, 1454–1459.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and M., S.P.O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36, 789–814.

Michalska, H. and Mayne, D.Q. (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 1623–1632.

Rawlings, J.B. and Mayne, D.Q. (2009). *Model Predictive Control: Theory and Design*. Nob Hill Publishing.

Rossiter, J.A. (2003). *Model-Based Predictive Control: A Practical Approach*. CRC Press.