Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-28 (2019) 146–151

qLPV Predictive Control - A Benchmark Study on State Space vs Input-Output Approach ⋆

Horacio M. Calderón * **Pablo S.G. Cisneros** * **Herbert Werner** *

* *Hamburg University of Technology, Hamburg, Germany (e-mail: horacio.martinez.calderon@tuhh.de; pablo.gonzalez@tuhh.de; h.werner@tuhh.de).*

Abstract: This paper presents a comparison and evaluation of two approaches to Nonlinear Model Predictive Control (NMPC) via quasi-LPV modeling, by means of a benchmark problem: control of a 4 degree-of-freedom Control Moment Gyroscope (CMG). The use of quasi-LPV modeling allows us to recast the nonlinear optimization problem arising in NMPC, as a repeated quadratic program which can be solved efficiently. The difference between the two presented schemes lies in the modeling paradigm chosen to express the dynamics of the system, namely state space (SS) or input-output (IO) frameworks. In both cases, quasi-LPV models are obtained by performing a velocity-based linearization, which results in an exact representation of the nonlinear dynamics and enables offset free control. Both schemes are successfully implemented on a laboratory CMG, and the experimental results are compared and discussed. Furthermore, advantages and disadvantages of each control scheme are examined.

Keywords: Linear parameter-varying systems, model predictive control, control moment gyroscope, linearization, multivariable control systems.

1. INTRODUCTION

The use of model predictive control (MPC) schemes has historically been restricted to the process industry, where the systems generally have large time constants. The reason behind this is that the relatively complex computations entailed by online solution of optimization problems led to high computation times, hence precluding real-time implementation of MPC algorithms to control plants with fast dynamics. In recent years, this tendency has shifted partly due to an increase in computational power but also because of the development of computationally efficient algorithms e.g. (Diehl et al. (2005)), (Käpernick and Graichen (2014)), (Cisneros et al. (2016)) which enable MPC control laws to be computed in the milli- or even microsecond range. One such algorithm is the quasi-LPV MPC (qLMPC) algorithm. This nonlinear control strategy uses a state space quasi-LPV model of the plant in order to make predictions, and consequently, it selects the input that achieves the desired control objectives. A variant of this algorithm is the velocity qLMPC based on a state space framework (SS-qLMPC), which was presented in (Cisneros et al. (2018)). It differs in that it uses a velocity-based linearization to obtain a quasi-LPV model in velocity-form. Both strategies assume full knowledge of the state vector and therefore often require the use of state observers thus increasing the complexity of the controller.

An alternative approach that uses input-output quasi-LPV (IO-qLPV) models was presented in (Cisneros and Werner (2019)). This algorithm only needs the output vector to implement the predictive control strategy, thus simplifying its design. When compared to its state space counterpart, it also offers the advantage that when no first principles model is available, input-output LPV identification tends to be simpler than LPV subspace identification (Schulz et al. (2017)).

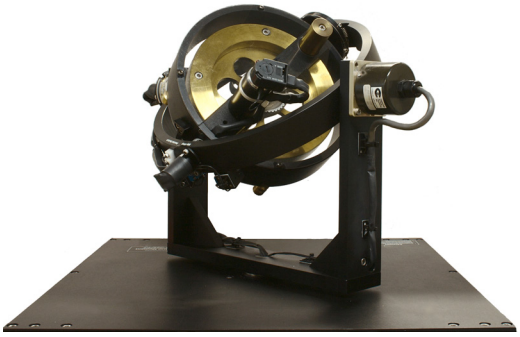

Fig. 1. ECP750 control moment gyroscope.

This paper aims to compare the benefits and shortcomings of each of the aforementioned approaches based on a benchmark problem: control of a 4 degree-of-freedom (DOF) Control Moment Gyroscope (CMG). A CMG is a mechanical device used to generate gyroscopic precession. CMGs are generally used for attitude control of spacecrafts (Bhat and Tiwari (2009)) as well as in roll stabilization of ships. The dynamic equations that describe the motion of a CMG are nonlinear and highly coupled, for this reason,

⋆ HMC acknowledges support from CONACYT-Mexico.

10.1016/j.ifacol.2019.12.362

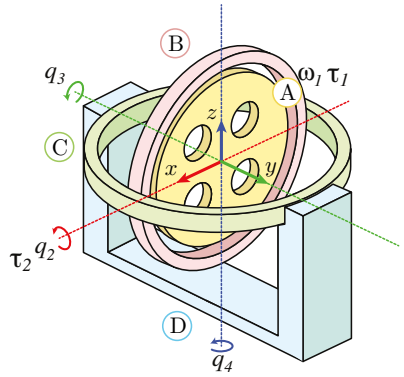

Fig. 2. Kinematic diagram of the CMG.

the design of controllers for this plant is a complex task, making it an excellent benchmark test for nonlinear controllers.

In recent years different control schemes for CMGs have been proposed. A gain scheduled controller that uses LPV techniques has been presented in (Hoffmann and Werner (2015)), and a robust nonlinear controller based on sliding mode control is presented in (Toriumi and Anglico (2018)). In this paper we present the design and implementation of two predictive control laws for a CMG. The proposed design approaches are: the SS-qLMPC, and the IO-qLMPC. These controllers are experimentally validated, and their results are compared and discussed.

This paper is structured as follows: Section 2 starts by giving a description of the CMG dynamics, after this its velocity based linearization is presented. Section 3 describes the SS-qLMPC and the IO-qLMPC algorithms. Section 4 presents the experimental results and a comparison is made. Finally Section 5 gives a conclusion of this work.

1.1 Notation

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

2. PLANT MODEL

The laboratory test bench to be used is a Model 750 CMG from Educational Control Products (Fig. 1). This is a 4 DOF CMG which consists of a motorized flywheel mounted on an actuated gimbal, which is in turn fixed on a set of two unactuated gimbals. In Fig. 2 a schematic of the CMG is shown, where body A is the flywheel with rotational speed ω_1, body B is the actuated gimbal, with rotational angle q_2, and the bodies C (angle q_3) and D (angle q_4) are the unactuated gimbals. Torque τ_1 is used to spin body A and torque τ_2 is used to rotate gimbal B (i.e. to tilt the flywheel). The equations that govern the dynamics of the CMG have the form (Hoffmann and Werner (2015))

$$M(q)\ddot{q} + D\dot{q} + K(q,\dot{q}) = \tau, \tag{1}$$

where $q = [q_1\ q_2\ q_3\ q_4]^\top$ is the vector of generalized coordinates, and $\tau = [\tau_1\ \tau_2\ 0\ 0]^\top$ is the vector of applied torques. The matrix $M(q)$ is the generalized inertia matrix, the vector $K(q,\dot{q})$ represents the centripetal and Coriolis forces, and the matrix D contains the viscous damping coefficients. In order to facilitate the construction of the quasi-LPV models, the second order differential equation (1) is converted into a first order one; to this end, the state vector $x = [q^\top\ \dot{q}^\top]^\top$, and the input $u = [u_1, u_2]^\top$ are defined, yielding the model

$$\dot{x} = \begin{bmatrix} \dot{q} \\ -M(q)^{-1}D\dot{q} - M(q)^{-1}k(q,\dot{q}) + M(q)^{-1}Bu \end{bmatrix}, \tag{2}$$

where $Bu = \tau$ and the output is defined as

$$y = \begin{bmatrix} \omega_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}. \tag{3}$$

2.1 Velocity based linearization.

The derivation of a quasi-LPV model from (1) is not a trivial task. An ad-hoc quasi-LPV parametrization of a simplified model of the CMG was presented in (Abbas et al. (2014)), this model is based on a linearization around a moving operating point, hence it does not fully represent the nonlinear dynamics, but rather a continuous family of Jacobian linearizations. The ad-hoc parametrization aims to hide the nonlinearities of the dynamic model under parameter variations. This approach is not systematic, and can be challenging for complex systems. In this paper we use velocity linearization, (Leith and Leithead (1998)), which leads to an exact representation of the nonlinear dynamics of the model, unlike the approximation given by a Jacobian linearization around an operating point. Indeed, a velocity linearization of a nonlinear model

$$\dot{x} = f(x,u), \tag{4}$$

is obtained by calculating its time derivative, yielding

$$\ddot{x} = \nabla_x f(x,u)\dot{x} + \nabla_u f(x,u)\dot{u}. \tag{5}$$

Notice that the Jacobian matrices $\nabla_x f(x,u), \nabla_u f(x,u)$, depend on x and u, and that the model is linear in $\dot{x}$ and $\dot{u}$, this significantly simplifies the effort needed to find a quasi-LPV parametrization. A consequence of using this linearization is that information of x is lost, however, it can be easily recovered by state augmentation. A velocity based linearization of the model (2) was carried out using the `Symbolic Math Toolbox` from `Matlab`, yielding

$$\begin{bmatrix} \ddot{q} \\ \dddot{q} \end{bmatrix} = \nabla_x f(x,u) \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} + \nabla_u f(x,u) \begin{bmatrix} \dot{u_1} \\ \dot{u_2} \end{bmatrix}, \tag{6}$$

where the Jacobian $\nabla_x f(x,u)$ depends on the state variables q_2, q_3, $\dot{q}_1$, $\dot{q}_2$, $\dot{q}_3$, $\dot{q}_4$, and on the inputs u_1 and u_2. The Jacobian $\nabla_u f(x,u)$ only depends on q_2 and q_3. Due to their complexity the values of the Jacobians are not shown here.

2.2 Velocity based quasi-LPV model

In order to build a state space quasi-LPV model out of (6) the vector $x_v = [y^\top\ \dot{x}^\top]^\top$ and the vector of scheduling parameters $\rho = [\dot{q}_1\ q_2\ q_3\ \dot{q}_2\ \dot{q}_3\ \dot{q}_4\ u_1\ u_2]$ are defined, resulting in

$$\begin{bmatrix}\dot{y}\\ \ddot{x}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & C\\ 0 & \nabla_x f(x,u)\end{bmatrix}}_{A_v(\rho)}\begin{bmatrix}y\\ \dot{x}\end{bmatrix} + \underbrace{\begin{bmatrix}0\\ \nabla_u f(x,u)\end{bmatrix}}_{B_v(\rho)}\dot{u}, \quad (7)$$
$$y_v = [I\ 0]\begin{bmatrix}y\\ \dot{x}\end{bmatrix},$$

where $C = [I\ 0]$. Since predictive control laws work in discrete time, the model (7) is discretized using Euler's method. This yields the discrete-time model

$$x_{v,k+1} = A(\rho_k)x_{v,k} + B(\rho_k)\Delta u_k \quad (8)$$

where

$$A(\rho_k) = I + A_v(\rho_k)T_s, \quad B(\rho_k) = B_v(\rho_k),$$
$$\Delta u_k = u_k - u_{k-1},$$

and $T_s = 0.01$s is the sampling period.

2.3 Input-Output quasi-LPV model

To obtain an input output model we now turn our attention to equation (6), particularly to the equation of $\ddot{q}$. We discretize this equation using backward Euler's method, yielding the IO-qLPV model

$$\mathbf{y}_k = -\sum_{i=1}^{3} A_i(\rho_k)z^{-i}\mathbf{y}_k + B_1(\rho_k)\Delta u_{k-1}, \quad (9)$$

where $A_i(\rho(k)) : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times l}$ and $B_1(\rho(k)) : \mathbb{R}^{n_\rho} \to \mathbb{R}^{l\times m}$ are continuous maps on ρ_k, the output is $\mathbf{y}_k = [\omega_{1,k},\ q_{2,k},\ q_{3,k},\ q_{4,k}]^\top$, and z^{-1} is the backward time-shift operator. In the process of discretization for the IO model, $\ddot{q}_1$ is integrated only up to the angular velocity ω_1, i.e. excluding its angular position. The angle q_1 is not of interest since it is dynamically irrelevant (as the flywheel is perpetually in motion). The velocities $\omega_{2,k}$, $\omega_{3,k}$, $\omega_{4,k}$, which are also part of ρ_k, are not included in the output because they can be estimated with $\mathbf{y}_k$.

3. PREDICTIVE CONTROLLERS

The procedures to obtain the predictive control laws used in this paper, as well as the theory and stability results were presented in (Cisneros et al. (2018)) and (Cisneros and Werner (2019)). Both schemes consider a given quadratic cost function, which is minimized online by solving an optimization problem in the form of QPs. The solution gives the optimal input that complies with the input and output constraints.

3.1 SS-qLMPC

The performance index of the SS-qLMPC control law is

$$J_k = \sum_{i=0}^{N-1}\left(||e_{k+i}||_T^2 + ||\dot{x}_{k+i}||_Q^2 + ||\Delta u_{k+i}||_R^2\right) + ||e_{k+N}||_{T_N}^2, \quad (10)$$

where $e_{k+i} = r_{k+i} - y_{k+i}$ is the tracking error, $\dot{x}_{k+i}$ is the velocity state, Δu_{k+i} is the input increment and N is the prediction horizon. The weighting matrices $Q \in \mathbb{R}^{n\times n}$, $T \in \mathbb{R}^{l\times l}$ and $T_N = \alpha T$ are positive semi-definite, $R \in \mathbb{R}^{m\times m}$ is positive definite and α is a positive scalar. Using (10) the online optimization problem is defined as

$$\min_{\Delta u} J_k \quad (11a)$$
subject to
$$x_{v,k+j+1} = A(\rho_{k+j})x_{v,k+j} + B(\rho_{k+j})\Delta u_{k+j}, \quad (11b)$$
$$u_{k+j} = u_{k-1} + \sum_{i=0}^{j}(\Delta u_{k+i}) \in \mathcal{U}, \quad (11c)$$
for $j \in [0,\ \ldots,\ N-1]$,
$$\dot{x}_{k+N} = 0. \quad (11d)$$

where $\mathcal{U}$ is the input constraint set. Note that the dynamic constraint (11b) is nonlinear due to the dependence on the scheduling trajectory ρ_k, which is in turn a function (x, u). To circumvent this issue we make use of the qLMPC algorithm presented in (Cisneros et al. (2016)). This algorithm turns the problem (11) into a repeated quadratic program (QP); see Section 3.3. It is based on iteratively determining the optimal scheduling trajectory

$$P_k = \begin{bmatrix}\rho_k\\ \rho_{k+1}\\ \vdots\\ \rho_{k+N-1}\end{bmatrix},$$

and using it along with the prediction equation

$$X_{k+1} = \Lambda(P_k)x_{v,k} + S(P_k)\Delta U_k, \quad (12)$$

where

$$X_{k+1} = \begin{bmatrix}x_{k+1}\\ x_{k+2}\\ \vdots\\ x_{k+N}\end{bmatrix}, \quad \Delta U_k = \begin{bmatrix}\Delta u_k\\ \Delta u_{k+1}\\ \vdots\\ \Delta u_{k+N-1}\end{bmatrix}$$

and

$$\Lambda(P_k) = \begin{bmatrix}A(\rho_k)\\ A(\rho_{k+1})A(\rho_k)\\ \vdots\\ A(\rho_{k+N-1})\cdots A(\rho_{k+1})A(\rho_k)\end{bmatrix}$$

$$S(P_k) = \begin{bmatrix}B(\rho_k) & 0 & \cdots & 0\\ A(\rho_{k+1})B(\rho_k) & B(\rho_{k+1}) & \cdots & 0\\ \vdots & \vdots & \ddots & 0\\ A(\rho_{k+N-1})\cdots A(\rho_{k+1})B(\rho_k) & \cdots & \cdots & B(\rho_{k+N-1})\end{bmatrix}.$$

With the scheduling trajectory $\{\rho_{k+i}\}$ fixed, the dynamic constraint is linear, and the optimization problem can be efficiently solved by solving a QP repeatedly (typically 1 or 2 times).

3.2 IO-qLMPC

The IO-qLMPC control law is obtained by solving at each sampling instant the optimization problem

$$\min_{\Delta u} \sum_{j=0}^{N-1}\left(||e_{k+j}||_T^2 + ||\Delta u_{k+j}||_R^2\right) + ||\mathbf{y}_{k+N}||_{T_N}^2 \quad (13a)$$
subject to
$$\mathbf{y}_{k+j} = -\sum_{i=1}^{n_a} A_i(\rho_{k+j})q^{-i}\mathbf{y}_{k+j} + \sum_{i=1}^{n_b} B_i(\rho_{k+j})q^{-i}u_{k+j} \quad (13b)$$
$$u_{k+j} = u_{k-1} + \sum_{i=0}^{j}(\Delta u_{k+i}) \in \mathcal{U} \quad (13c)$$
for $j = [0,\ \ldots,\ N-1]$

As before, the optimization problem (13) can be turned into a repeated QP by using the future scheduling trajec-

tory and a prediction equation, which for the input output case is given by

$$C_a(P_k)Y_{k+1} + H_a(P_k)Y_k^p = C_b(P_k)\Delta U_k + H_b(P_k)\Delta U_{k-1}^p. \tag{14}$$

where

$$Y_{k+1} = \begin{bmatrix} \mathbf{y}_{k+1} \\ \mathbf{y}_{k+2} \\ \vdots \\ \mathbf{y}_{k+N} \end{bmatrix},\ Y_k^p = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k-1} \\ \vdots \\ \mathbf{y}_{k-n_a+1} \end{bmatrix}, \Delta U_{k-1}^p = \begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \vdots \\ \Delta u_{k-n_b+1} \end{bmatrix}$$

and

$$C_a(P_k) = \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ A_1(\rho_{k+1}) & I & 0 & \dots & 0 \\ A_2(\rho_{k+2}) & A_1(\rho_{k+2}) & I & \ddots & 0 \\ \vdots & & \ddots & & \vdots \\ A_{n_a}(\rho_{k+n_a}) & A_{n_a-1}(\rho_{k+n_a}) & \dots & & \\ 0 & \ddots & \dots & \ddots & 0 \\ \vdots & 0 & A_{n_a}(\rho_{k+N-1}) & \dots & I \end{bmatrix},$$

$$H_a(P_k) = \begin{bmatrix} A_1(\rho_k) & A_2(\rho_k) & A_3(\rho_k) & \dots & A_{n_a}(\rho_k) \\ A_2(\rho_{k+1}) & A_3(\rho_{k+1}) & A_4(\rho_{k+1}) & \ddots & 0 \\ A_3(\rho_{k+2}) & \ddots & \ddots & 0 & 0 \\ \vdots & A_{n_a}(\rho_{k+n_a-2}) & \ddots & \ddots & \vdots \\ A_{n_a}(\rho_{k+n_a-1}) & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & 0 & 0 & \dots & 0 \end{bmatrix},$$

$$C_b(P_k) = \begin{bmatrix} B_1(\rho_k) & 0 & 0 & \dots & 0 \\ B_2(\rho_{k+1}) & B_1(\rho_{k+1}) & 0 & \dots & 0 \\ B_3(\rho_{k+2}) & B_2(\rho_{k+2}) & B_1(\rho_{k+2}) & \ddots & 0 \\ \vdots & & \ddots & & \vdots \\ B_{n_b}(\rho_{k+n_b-1}) & \ddots & \dots & & \\ 0 & \ddots & \dots & \ddots & 0 \\ \vdots & 0 & B_{n_b}(\rho_{k+N-1}) & \dots & B_1(\rho_{k+N-1}) \end{bmatrix},$$

$$H_b(P_k) = \begin{bmatrix} B_2(\rho_k) & B_2(\rho_k) & B_3(\rho_k) & \dots & B_{n_b}(\rho_k) \\ B_3(\rho_{k+1}) & B_3(\rho_{k+1}) & B_4(\rho_{k+1}) & \ddots & 0 \\ B_4(\rho_{k+2}) & \ddots & \ddots & 0 & 0 \\ \vdots & B_{n_a}(\rho_{k+n_a-3}) & \ddots & \ddots & \vdots \\ B_{n_b}(\rho_{k+n_a-2}) & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & 0 & 0 & \dots & 0 \end{bmatrix}.$$

3.3 Scheduling trajectory approximation

A key element of both predictive control laws is the use of quasi-LPV modelling to schedule the prediction equations turning them into LTV, thus allowing the transformation of the nonlinear optimization problems into repeated QPs. The prediction equations require the knowledge of the future scheduling parameters P_k, which is not available. However, by exploiting the fact that for quasi-LPV systems the scheduling parameters depend on the states and/or inputs, P_k can be determined iteratively. The procedure for the state space case is implemented as follows:

(1) At time step $k = 0$, the values x_k and u_{k-1} are known, and the optimization problem (11) is solved using the quasi-LPV model (8) and the frozen parameter trajectory $P_{k=0}^{s=0} = \mathbf{1}_N \otimes \varrho(x_k, u_{k-1})$, where s indicates the iteration step in the search for the optimal scheduling sequence P_k^* and $\rho_k = \varrho(x_k, u_{k-1})$.
(2) Solving the problem (11) yields the vector of future control inputs [1] $U_k^{s=1}$. Using this vector, we can calculate the future states $X_k^{s=1}$ with (12), and by extension the parameter trajectory $P_k^{s=1} = \varrho(X_k^{s=1}, U_k^{s=1})$. Now we can solve again problem (11) and obtain the vectors $X_k^{s=2}$, $U_k^{s=2}$ and $P_k^{s=2} = \varrho(X_k^{s=2}, U_k^{s=2})$. Note that the quasi-LPV model is now a linear time varying (LTV) system, since the sequence $P_k^{s=1}$ is not constant. Finally the scheduling P_k^s is then iteratively driven towards its optimal value $P_k^* = \varrho(X_k^*, U_k^*)$, where U_k^* denotes the input sequence of the optimal solution to problem (11).
(3) After the final iteration step s , X_k^s and U_k^s are saved to calculate in the next step $P_{k+1}^0 = \varrho(X_k^s, U_k^s)$. This allows for faster convergence.

Note that for this algorithm to work in the input output case slight modifications have to be made. Such as the use of a different prediction equation, and the estimation of scheduling variables that are not in $\mathbf{y}_k$. The complete algorithms for both predictive strategies are shown in the Appendix.

[1] Recall that U_k depends on u_{k-1} and ΔU_k.

4. EXPERIMENTAL RESULTS

This section presents the results of the experimental implementation of the two control schemes discussed throughout the paper. Two different experiments are to be carried out: The first one is the tracking of step changes, which were chosen such that cross coupling between the two channels is to be suppressed. The second experiment is to track a combination of ramps and sinusoidal signals, which are meant to test the situation when the outputs q_3 and q_4, move in opposite directions at the same time.

4.1 Experimental setup

To perform the experiments, the CMG is first brought to the operating point $\dot{q}_1 = 45\frac{\text{rad}}{\text{s}}$, $q_2 = 0°$, $q_3 = 0°$, $q_4 = 0°$. This is achieved by a PI regulator, and after 15s the control authority is given to the MPC controller through a time switch. The controllers were implemented using `Matlab & Simulink` on a 3.5GHz Intel Core-i5 4670 CPU. The angles q_2, q_3, q_4, are measured using encoders and the corresponding velocities and accelerations are estimated using differential filters.

The prediction horizon for both controllers is $N = 30$. The tuning matrices for the SS-qLMPC are $T = \text{diag}(800, 100)$, $R = \text{diag}(1500, 1500)$ and $Q = \text{diag}(0, 5, 2, 2, 0, 0, 0, 0)$; and for the IO-qLMPC are $T = \text{diag}(0,0,1500,1500)$ and $R = \text{diag}(2500, 2500)$. The input constraints are $|\tau_1| \leq 0.66$Nm and $|\tau_2| \leq 2.44$Nm. The tuning for both designs was made such that rapid changes in the input are avoided (recall that R weights input increments) in order to avoid exciting unmodelled high frequency dynamics. In the SS-qLMPC design, the weight Q is used to provide additional damping by weighting velocities.

The controllers were tested in simulation using the nonlinear model (1). This simulation provided us with estimates of the execution times of the algorithms. Figure 3 shows the algorithm's execution time when the reference of Experiment 2 is to be tracked using horizons $N = 30$ and $N = 70$. When the horizon $N = 30$ is selected, both algorithms are executed in each sampling interval in less

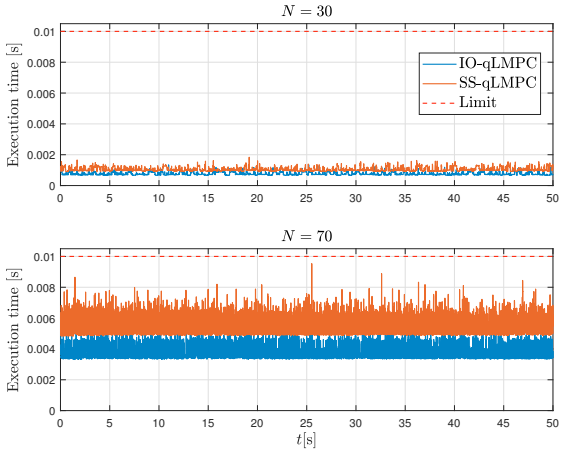

Fig. 3. Simulation: Execution time of the algorithms.

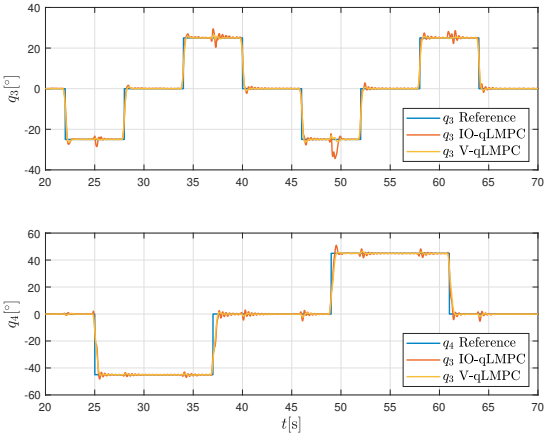

Fig. 4. Experiment 1: Tracking of step changes.

that 2ms; it can be observed that the IO-qLMPC is slightly faster than the SS-qLMPC. Moreover, when the horizon is increased to $N = 70$, it is clear that the IO-qLMPC is faster. A possible reason of this is that the operations entailed by the calculation of the prediction equations (12), (14) scale better for the IO case.

4.2 Trajectory Tracking

In Figure 4 the results of the first experiment are shown. The response of both controllers is remarkably fast, and the IO-qLMPC exhibits vibrations. A possible cause of this behaviour is that the rapid response of the closed-loop system induces vibrations in the system. In the IO-qLMPC case the cost function does not provide a way to straightforwardly add damping (as in the SS case, where a penalty on velocities can be implemented) consequently, vibrations are difficult to damp. Both of the controllers exhibit cross coupling, but in the IO-qLMPC it is more pronounced. In Figure 5 the control inputs related to this experiment are shown. Notice that both controllers exploit the available torque and saturate when rapid changes are required. Figure 6 illustrates the results of the second experiment. In this case the IO-qLMPC achieves a very small steady state error for the ramps and sine references, when compared to its SS-qLMPC counterpart, although

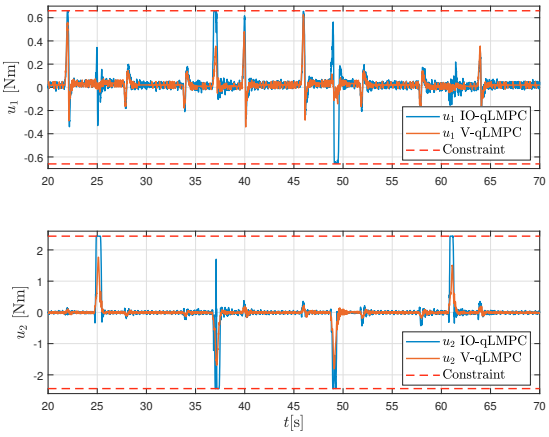

Fig. 5. Experiment 1: Input signals.

the latter still achieves outstanding performance. Notice that in this experiment the amplitude of the references is almost twice as of the first experiment. This demonstrates the tracking capabilities of the controllers over a wide operation range.

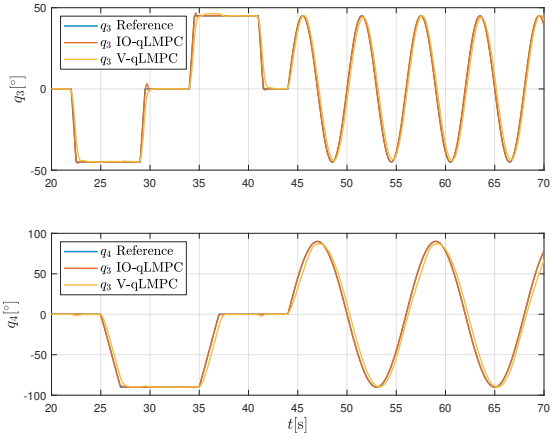

Fig. 6. Experiment 2: Tracking of the ramps and sinusoidal references.

4.3 Comparison

To evaluate the controllers performance the root mean square value of the tracking error signals was calculated. Table 1 shows the results for each experiment. It can be seen that SS-qLMPC outperforms the IO-qLMPC for Experiment 1, but only for a small fraction. On the other hand the performance of the IO-qLMPC is clearly better in Experiment 2. The main advantage of the SS-qLMPC is its ability to penalize velocities, thereby providing a simple way to include damping. In the case of the IO-qLMPC a faster execution time is achieved, which allows to have a longer horizon, and consequently better performance. The main disadvantage of this controller lies in the exclusion of the velocities from the output, since this entails a lack of damping. On the other hand, this allows for a reduction in the controller's complexity.

Table 1. RMS values of the tracking errors

Controller	RMS Value Exp. 1 [°]	RMS Value Exp. 2 [°]
SS-qLMPC	3.38	8.37
IO-qLMPC	3.7	0.9484

5. CONCLUSION

In this paper the design and implementation of two predictive control laws for the trajectory tracking of a CMG was presented. Both controllers utilize the velocity linearization in order to obtain a suitable quasi-LPV model which allows for a wide operation range. Experimental results show the efficacy and efficiency of the qLMPC algorithm, which is essential for the presented control strategies. At the same time these results highlight the main benefits and drawbacks of each controller.

The IO-qLMPC can penalize velocities, if they are included in the output. However, in this case it was considered impractical. The IO-qLMPC is faster than the SS-qLMPC, and its execution time scales better when the horizon is increased. Both predictive controllers achieve zero steady state error for set point tracking, and high performance is achieved through the exploitation of the explicit definitions of input constraints, which allow the controllers to use the maximum input value when required.

REFERENCES

Abbas, H.S., Ali, A., Hashemi, S.M., and Werner, H. (2014). Lpv state-feedback control of a control moment gyroscope. *Control Engineering Practice*, 24, 129–137.

Bhat, S.P. and Tiwari, P.K. (2009). Controllability of spacecraft attitude using control moment gyroscopes. *IEEE Transactions on Automatic Control*, 54(3), 585–590. doi:10.1109/TAC.2008.2008324.

Cisneros, P.S.G., Sridharan, A., and Werner, H. (2018). Constrained predictive control of a robotic manipulator using quasi-lpv representations. In *LPVS - 2nd IFAC Workshop on Linear Parameter Varying Systems*.

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

Cisneros, P.S.G. and Werner, H. (2019). Stabilizing model predictive control for nonlinear systems in input-output quasi-lpv form. In *2019 American Control Conference*. Accepted.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control Optim.*, 43(5), 1714–1736. doi:10.1137/S0363012902400713. URL `https://doi.org/10.1137/S0363012902400713`.

Hoffmann, C. and Werner, H. (2015). Lft-lpv modeling and control of a control moment gyroscope. In *54th Conference on Decision and Control*.

Käpernick, B. and Graichen, K. (2014). The gradient based nonlinear model predictive control software grampc. In *2014 European Control Conference (ECC)*, 1170–1175. doi:10.1109/ECC.2014.6862353.

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.

Schulz, E., Cox, P.B., Toth, R., and Werner, H. (2017). LPV state-space identification via io methods and efficient model order reduction in comparison with subspace methods. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 3575–3581. IEEE.

Toriumi, F.Y. and Anglico, B.A. (2018). Robust nonlinear control applied to a control moment gyroscope with siso configuration. *IFAC-PapersOnLine*, 51(25), 152 – 157. 9th IFAC Symposium on Robust Control Design ROCOND 2018.

Appendix A. QLMPC ALGORITHMS

The IOqLMPC is summarized in Algorithm 1, and the SS-qLMPC in Algorithm 2.

Algorithm 1. IO-qLMPC algorithm.

Initialize T,R,N, $\mathcal{U}$ and s_{max}
for $k = 0$ *to* T_{sim} **do**
  Read y_k and u_{k-1}
  if *k=0* **then**
    Define $P_k^0 = 1_N \otimes f(y_k, u_{k-1})$
  end
  for $s = 0$ *to* s_{max} *or stop criterion* **do**
    Calculate $H_a(P_k^s)$, $C_a(P_k^s)$, $H_b(P_k^s)$ and $C_b(P_k^s)$
    Set the QP (13a)
    Y_k^s , $\Delta U_k^s \leftarrow$ Solve QP
    $U_k^s = \vartheta(\Delta U_k^s, u_{k-1})$
    Define $P_k^{s+1} = f(Y_k^s, U_k^s)$
  end
  Define $P_{k+1}^0 = f(Y_k^s, U_k^s)$
  Apply $u_k = \Delta u_k + u_{k-1}$ to the system
end

Algorithm 2. SS-qLMPC algorithm.

Initialize T,Q,R,N,α, $\mathcal{U}$, s_{max}
for $k = 0$ *to* T_{sim} **do**
  Read $x_{v,k}$ and u_{k-1}
  if *k=0* **then**
    Define $P_k^0 = 1_N \otimes f(x_{v,k}, u_{k-1})$
  end
  for $s = 0$ *to* s_{max} *or stop criterion* **do**
    Calculate $H(P_k^s)$ and $S(P_k^s)$
    Set the QP (11a)
    X_k^s, $\Delta U_k^l \leftarrow$ Solve QP
    $U_k^s = \vartheta(\Delta U_k^s, u_{k-1})$
    Define $P_k^{s+1} = f(X_k^s, U_k^s)$
  end
  Define $P_{k+1}^0 = f(X_k^l, U_k^l)$
  Apply $u_k = \Delta u_k + u_{k-1}$ to the system
end