2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Real-Time Nonlinear Model Predictive Control of a Motion Simulator Based on a 8-DOF Serial Robot

Mikhail Katliar[1,2], Frank M. Drop[1], Harald Teufel[1], Moritz Diehl[2] and Heinrich H. Bülthoff[1]

Abstract— **In this paper we present the implementation of a model predictive controller (MPC) for real-time control of a motion simulator based on a serial robot with 8 degrees of freedom. The goal of the controller is to accurately reproduce six reference signals simultaneously (the accelerations and angular velocities in the body frame of reference) taken from a simulated or real vehicle, by moving the human participant sitting inside the cabin located at the end effector. The controller computes the optimal combined motion of all axes while keeping the axis positions, velocities and accelerations within their limits. The motion of the axes is computed every $12\,\mathrm{ms}$ based on a prediction horizon consisting of 60 steps, spaced $48\,\mathrm{ms}$ apart, thus looking ahead $2.88\,\mathrm{s}$. To evaluate tracking performance, we measured the acceleration and angular velocity in the cabin using an Inertial Measurement Unit (IMU) for synthetic (doublets and triangle-doublets) and realistic (recorded car and helicopter maneuvers) reference signals. We found that fast-changing acceleration inputs excite the natural frequencies of the system, leading to severe mechanical oscillations. These oscillations can be modelled by a second-order LTI system and mitigated by including this model in the controller. The use of proper algorithms and software allows the computations to be done in real-time.**

I. Introduction

Motion simulators are devices for creating an experience of being inside a moving vehicle. They are widely used for training of aircraft pilots and race car drivers, as well as for studying human motion perception and control behavior. A motion simulator typically consists of a motion system (moving platform) and a visualization system. The human vestibular system senses inertial forces and angular velocity. We refer to these quantities, as functions of time, as the "inertial signal". A control algorithm that drives the moving platform to reproduce a certain inertial signal is called *motion cueing algorithm* (MCA). Different types of MCAs are reviewed in [1]. Here, we do not take sensory or perceptual dynamics of the human into account [2], but focus on the physical reproduction of the inertial signal.

The major difficulty concerning motion cueing is that the linear acceleration needs to be reproduced, which is the second-order derivative of the position. If reproduced one-to-one and integrated over time, it will likely result in a position outside the motion system's physical limits. Therefore, the inertial signal in a simulator has to be necessarily distorted, in order to keep the resulting trajectory within the limits of the motion system.

Recently, it was proposed to use model predictive control (MPC) for motion simulation [3], [4], [5]. In MPC, control inputs are calculated at each time step by numerically minimizing a cost function that reflects (among other) the deviation of the predicted output trajectory from the desired reference trajectory over the *prediction horizon* of N steps [6]. An advantage of MPC is the possibility to impose input and state constraints on the optimization problem.

The MPC architectures of [3], [4], [5] consist of two parts: an MPC-based feedforward compensator that defines the trajectory for the end effector (the motion platform) to follow, and a feedback controller that handles actuator dynamics. The feedback controller makes the dynamics of the resulting system linear and decouples the MPC part from the dynamics of the actuators. The motion constraints are formulated in terms of ranges for linear displacement and velocity, orientation angles and corresponding angular rates. However, when the motion system is a serial robot with multiple revolute joints, the full set of feasible trajectories of the end effector can not be described in this way. Furthermore, when the motion system is over-actuated, the motion of the joints is not uniquely determined by the motion of the end effector.

Motion constraints of a serial robot are usually defined in terms of position, velocity and acceleration ranges for each axis. Therefore, it is much more convenient to formulate the MPC problem in axis space rather than in end effector space. This formulation enables the full use of the simulator's motion capabilities. We derived and implemented an MPC for the 8 DOF serial robot CyberMotion Simulator at the Max Planck Institute for Biological Cybernetics. To validate the implemented controller, we measured the inertial signal in the cabin and compared it to the reference signal for synthetic and realistic signals.

II. The MPI CyberMotion Simulator

The CyberMotion Simulator (CMS) at the Max Planck Institute for Biological Cybernetics in Tübingen, Germany, is used for basic research in human motion perception and to investigate the influence of motion in vehicle control tasks [7]. The simulator is based on a 6-degree-of-freedom robot arm (Robocoaster from KUKA Roboter GmbH, Germany). The robot arm is equipped with an actuated cabin at its end-effector, and moves along a $10\,\mathrm{m}$ rail at its base, see Fig. 1.

The kinematic chain for axes 0–6 is described by Denavit-Hartenberg parameters presented in Table I. The position,

[1]Dept. of Human Perception, Cognition and Action, Max Planck Institute for Biological Cybernetics, Max-Planck-Ring 8, 72076 Tübingen, Germany [mikhail.katliar, frank.drop, harald.teufel, heinrich.buelthoff]@tuebingen.mpg.de

[2]Dept. of Microsystems Engineering & Dept. of Mathematics University of Freiburg, Georges-Koehler-Allee 102, D-79110 Freiburg, Germany moritz.diehl@imtek.uni-freiburg.de

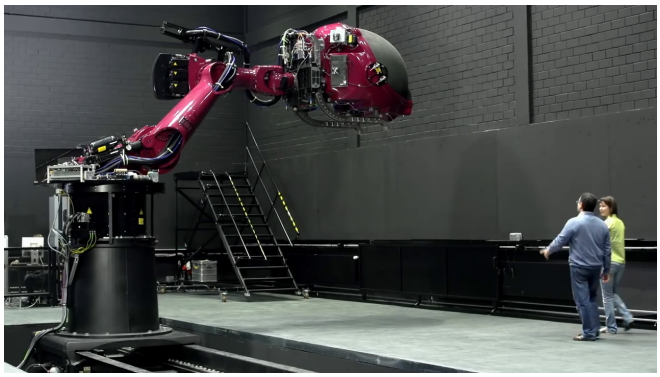


Fig. 1. The MPI CyberMotion Simulator (CMS).

velocity and acceleration limits of the axes are presented in Table II. Axis 0 is a linear rail (prismatic joint), axes 1–6 are revolute joints. Axis 7 is a curvilinear axis which is used to orient the cabin and is normally not moved during the simulation; its kinematics are described in [8]. All axes are electrically actuated and equipped with position sensors; axes 0 and 7 are additionally equipped with velocity sensors. The MPC controller described in Section III controls all 8 axes of the robot. The position limits of axis 7 are set close to each other and the velocity limits are set close to zero to restrict its motion during the experiment.

TABLE I

DENAVIT-HARTENBERG PARAMETERS OF THE CMS KINEMATIC CHAIN.

Axis	a [m]	d [m]	α [rad]	θ [rad]
0	0	q_0	$-\pi/2$	π
1	0.5	2.03	$-\pi/2$	$-q_1$
2	1.3	0	0	q_2
3	0.055	0	$\pi/2$	$q_3 - \pi/2$
4	0	-1.025	$-\pi/2$	q_4
5	0	0	$\pi/2$	q_5
6	0	-0.29	0	q_6

TABLE II

AXIS LIMITS OF THE CMS USED IN THE EXPERIMENT.

Axis (unit $\cdot$)	$q_{\min}$ $[\cdot]$	$q_{\max}$ $[\cdot]$	$v_{\max}$ $[\cdot/\mathrm{s}]$	$v_{\max,N}$ $[\cdot/\mathrm{s}]$	$u_{\max}$ $[\cdot/\mathrm{s}^2]$
0 (m)	0.200	9.100	1.470	0.001	1.078
1 (rad)	-20	20	1.180	0.001	1.676
2 (rad)	-2.050	-0.980	0.975	0.001	1.197
3 (rad)	-0.680	1.450	1.180	0.001	2.189
4 (rad)	-3.016	3.016	1.300	0.001	0.564
5 (rad)	-0.770	0.770	1.300	0.001	1.625
6 (rad)	-3.016	3.016	2.053	0.001	1.317
7 (m)	1.500	1.600	0.010	0.001	0.100

III. THE MODEL PREDICTIVE MOTION CUEING CONTROLLER

In this section, an MPC controller is derived that calculates optimal combined motion of all axes of the robot to track a given inertial signal inside the cabin while respecting the axis motion limits. First, the model of the system dynamics (Sec. III-A) and the output function (Sec. III-B) are derived. Then, the control task is formulated as a nonlinear receding horizon optimization problem (Sec. III-C). Finally, the implementation of a real-time capable model predictive controller is presented that numerically solves this optimization problem at each time sample (Sec III-D).

A. Model of the system dynamics

The cabin of the CMS is connected to the fixed base via $n_a = 8$ motion joints (or *axes*) forming a serial kinematic chain. The state $\mathbf{x}$ of the system is defined by positions $\mathbf{q} = [q_0, q_1, \ldots, q_{n_a}]^\top \in \mathbb{R}^{n_a}$ and velocities $\mathbf{v} = [v_0, v_1, \ldots, v_{n_a}]^\top \in \mathbb{R}^{n_a}$ of all axes:

$$\mathbf{x} = [\mathbf{q}^\top, \mathbf{v}^\top]^\top \in \mathbb{R}^{n_x},\ n_x = 16\ . \tag{1}$$

The position $\mathbf{q}$ is in length units for axes 0 and 7, and in angular units for axes 1–6.

The MPC motion cueing controller is built on top of a low-level position controller, which runs at a sampling time $t_s = 12\,\mathrm{ms}$. The input of the position controller are the desired axis position increments $\mathbf{\Delta} \in \mathbb{R}^{n_a}$ to be executed within one sampling interval. We assume that the position controller accurately follows these increments and that the axis velocity at *the end* of the control interval equals the *average* velocity on that interval. Hence, the state after one sampling time t_s is

$$\mathbf{x}_{k+1} = \begin{bmatrix}\mathbf{q}_{k+1}\\ \mathbf{v}_{k+1}\end{bmatrix} = \begin{bmatrix}\mathbf{q}_k + \mathbf{\Delta}_k\\ \mathbf{\Delta}_k/t_s\end{bmatrix},\ k = 0, 1, 2, \ldots\ , \tag{2}$$

where index k corresponds to a discrete time step of the position controller. The axis acceleration $\dot{\mathbf{v}}(t)$ is required to calculate the end effector acceleration. Since the actual axis acceleration profile during one sampling interval is not known, we approximate the instantaneous axis acceleration by the *average* axis acceleration $\mathbf{u}$ on one control interval:

$$\dot{\mathbf{v}}(t) \approx \mathbf{u}_k = \frac{\mathbf{v}_{k+1} - \mathbf{v}_k}{t_s} = \frac{\mathbf{\Delta}_k}{t_s^2} - \frac{\mathbf{v}_k}{t_s}\ . \tag{3}$$

The axis increments can now be expressed as

$$\mathbf{\Delta}_k = t_s\mathbf{v}_k + t_s^2\mathbf{u}_k\ , \tag{4}$$

such that, after substituting (4) into (2), we obtain

$$\mathbf{x}_{k+1} = \begin{bmatrix}\mathbf{q}_k + t_s\mathbf{v}_k + t_s^2\mathbf{u}_k\\ \mathbf{v}_k + t_s\mathbf{u}_k\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbb{1} & t_s\mathbb{1}\\ \mathbb{0} & \mathbb{1}\end{bmatrix}}_{A}\mathbf{x}_k + \underbrace{\begin{bmatrix}t_s^2\mathbb{1}\\ t_s\mathbb{1}\end{bmatrix}}_{B}\mathbf{u}_k\ , \tag{5}$$

where $\mathbb{1}$ is an $n_a \times n_a$ identity matrix. Equation (5) defines the discrete time dynamics of the system with sampling time t_s, where average axis accelerations $\mathbf{u} \in \mathbb{R}^{n_u}$, $n_u = 8$ are the "virtual" control inputs. The actual control inputs are the increments $\mathbf{\Delta}$. Note that our assumption about the behavior of the position controller results in equations which are different from the ones that can be derived assuming constant axis acceleration during the control interval. Tests

showed that using the latter model the produced accelerations were about 2 times weaker than expected.

The MPC controller runs at the same sampling time t_s as the underlying position controller. However, the prediction horizon is divided into intervals of length T which is an integer multiple of t_s: $T = mt_s$, $m \in \mathbb{Z}_+$. The rationale behind this is that the inertial signals have bandwidth of $\approx 10\,\mathrm{Hz}$, therefore longer sampling times up to $\approx 50\,\mathrm{ms}$ can be used for their representation, which also allows longer prediction horizon in terms of time. Assuming that the same control input is applied m times in a row in (5), we obtain a downsampled model

$$\begin{aligned} \mathbf{x}_{k+m} &= A_m \mathbf{x}_k + B_m \mathbf{u}_k \\ A_m &= A^m = \begin{bmatrix} \mathbb{1} & mt_s\mathbb{1} \\ \mathbb{0} & \mathbb{1} \end{bmatrix} \\ B_m &= \sum_{i=0}^{m-1} A^i B = \begin{bmatrix} \frac{(m+1)m}{2} t_s \mathbb{1} \\ mt_s \mathbb{1} \end{bmatrix} . \end{aligned} \tag{6}$$

B. Calculating the output values

In a motion simulator, we want to reproduce inertial forces and angular velocities in a reference frame attached to the pilot's head (the *head frame*, HF) as close as possible to a given reference profile. These quantities need to be expressed as functions of system state and control input. Consider a point mass m attached to the cabin at the origin of HF. Let $-\mathbf{F}_\mathrm{H}$ denote the cabin reaction force acting on the mass, expressed in HF. According to Newton's 2nd law,

$$m\ddot{\mathbf{r}}_\mathrm{H}^\mathrm{W} = R_\mathrm{H}^\mathrm{W}(\mathbf{q})(-\mathbf{F}_\mathrm{H}) + m\mathbf{g}_\mathrm{W} , \tag{7}$$

where $\mathbf{r}_\mathrm{H}^\mathrm{W}$ is the position of the origin of HF expressed in world frame (WF), $R_\mathrm{H}^\mathrm{W}(\mathbf{q})$ is the rotation matrix from HF to WF and $\mathbf{g}_\mathrm{W}$ is the gravity vector in WF. The quantity

$$\mathbf{f}_\mathrm{H}(\mathbf{x}, \mathbf{u}) := \frac{\mathbf{F}_\mathrm{H}}{m} = R_\mathrm{H}^\mathrm{W}(\mathbf{q})^\top \left(\mathbf{g}_\mathrm{W} - \ddot{\mathbf{r}}_\mathrm{H}^\mathrm{W} \right) \tag{8}$$

is the *specific force* measured in $\mathrm{m/s^2}$, which is essentially the "inertial force" in HF divided by mass. It is the combined result of gravity and accelerated motion of the HF.

The homogeneous transformation matrix from HF to WF is

$$T_\mathrm{H}^\mathrm{W}(\mathbf{q}) = \begin{bmatrix} R_\mathrm{H}^\mathrm{W}(\mathbf{q}) & \mathbf{r}_\mathrm{H}^\mathrm{W}(\mathbf{q}) \\ \mathbb{0} & 1 \end{bmatrix} = \prod_{i=0}^{n_a-1} T_i^{i-1}(q_i) \cdot T_\mathrm{H}^\mathrm{C} , \tag{9}$$

where $T_i^{i-1}(q_i)$ is the homogeneous transformation matrix from the reference frame of axis i to the frame of axis $i-1$ (or to WF in case $i = 0$), and T_H^C is the constant transformation matrix from HF to the last axis (cabin) frame. The $T_i^{i-1}(q_i)$ matrices are computed based on Denavit-Hartenberg parameters from Table I for $i = 0 \ldots 6$, and based on the kinematic model of the cabin axis [8] for $i = 7$.

The acceleration $\ddot{\mathbf{r}}_\mathrm{H}^\mathrm{W}$ is obtained by taking the second time-derivative of (9), which results in a function of $\mathbf{q}$, $\mathbf{v}$ and $\dot{\mathbf{v}}$. We approximate $\dot{\mathbf{v}}$ by its average $\mathbf{u}$ as in (3).

The angular velocity of HF with respect to WF expressed in WF can be found from the equation

$$[\boldsymbol{\omega}_\mathrm{W}]_\times = \dot{R}_\mathrm{H}^\mathrm{W}(\mathbf{q}, \mathbf{v}) R_\mathrm{W}^\mathrm{H}(\mathbf{q}) = \dot{R}_\mathrm{H}^\mathrm{W}(\mathbf{q}, \mathbf{v}) R_\mathrm{H}^\mathrm{W}(\mathbf{q})^\top ,$$

where $[\,\cdot\,]_\times$ denotes the cross-product matrix. The angular velocity in HF is

$$\boldsymbol{\omega}_\mathrm{H}(\mathbf{x}) = R_\mathrm{W}^\mathrm{H}(\mathbf{q}) \boldsymbol{\omega}_\mathrm{W} .$$

Finally, the combined inertial signal produced by the simulator is

$$\mathbf{y}(\mathbf{x}, \mathbf{u}) = [\mathbf{f}_\mathrm{H}(\mathbf{x}, \mathbf{u})^\top, \boldsymbol{\omega}_\mathrm{H}(\mathbf{x})^\top]^\top \in \mathbb{R}^{n_y},\ n_y = 6 . \tag{10}$$

By convention, the X axis of HF is pointing forward in the direction of pilot's sight, the Y axis to the left and the Z axis up.

C. Formulation of the optimal control problem

Let $\tilde{\mathbf{x}}_0$ be the state of the system at current time t_0 and $\hat{\mathbf{y}}_k$ be the reference inertial signal at time $t_0 + kT$. At every time sample, the MPC controller solves the following nonlinear constrained minimization problem:

$$\begin{aligned} \underset{\substack{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1} \\ \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N}}{\text{minimize}} \quad & \frac{1}{N} \sum_{k=0}^{N-1} l_k(\mathbf{x}_k, \mathbf{u}_k) + l_N(\mathbf{x}_N) \\ \text{subject to} \quad & \mathbf{x}_0 = \tilde{\mathbf{x}}_0, \\ & \mathbf{x}_{k+1} = A_m \mathbf{x}_k + B_m \mathbf{u}_k,\ k = 0 \ldots N-1, \\ & \mathbf{x}_{\min,k} \le \mathbf{x}_k \le \mathbf{x}_{\max,k},\ k = 1 \ldots N, \\ & \mathbf{u}_\min \le \mathbf{u}_k \le \mathbf{u}_\max,\quad k = 0 \ldots N-1 \end{aligned} \tag{11}$$

where

$$\begin{aligned} l_k(\mathbf{x}_k, \mathbf{u}_k) = & \left\| \mathbf{y}(\mathbf{x}_k, \mathbf{u}_k) - \hat{\mathbf{y}}_k \right\|_{W_\mathbf{y}}^2 \\ & + \|\mathbf{x}_k - \hat{\mathbf{x}}_k\|_{W_\mathbf{x}}^2 + \|\mathbf{u}_k - \hat{\mathbf{u}}_k\|_{W_\mathbf{u}}^2 , \end{aligned} \tag{12}$$

$$l_N(\mathbf{x}_N) = \|\mathbf{x}_N - \hat{\mathbf{x}}_N\|_{W_{\mathbf{x}_N}}^2 . \tag{13}$$

The position and velocity limits of the axes are included in $\mathbf{x}_{\min,k}$ and $\mathbf{x}_{\max,k}$, whereas $\mathbf{u}_\min$, $\mathbf{u}_\max$ are the axis acceleration limits. The bounds $\mathbf{x}_{\min,N}$, $\mathbf{x}_{\max,N}$ of the terminal state $\mathbf{x}_N$ are chosen to constrain the velocities of the axes to small values. This ensures that the motion can be stopped at the end of the prediction horizon.

Controller behavior is determined by the symmetric positive-definite weighting matrices $W_\mathbf{y}$, $W_\mathbf{x}$, $W_{\mathbf{x}_N}$, $W_\mathbf{u}$. It is possible to switch or balance between output, state and input tracking without restarting the controller, by changing the weighting matrices. The first term in (12) corresponds to the primary application of the controller: output tracking. That is, the tracking of the n_y components of the inertial signal $\hat{\mathbf{y}}$, where $W_\mathbf{y}$ defines the weighting of each individual component. The latter two terms allow the controller to track a given state trajectory $\hat{\mathbf{x}}_k$ and input trajectory $\hat{\mathbf{u}}_k$. A non-zero $W_\mathbf{u}$ penalizes control input and helps to promote local convexity of (11).

State tracking has two useful applications. First, it can be used to move the robot to the desired starting position before experiment trials. Second, it can be used to achieve "washout" behavior during output tracking. That is, setting $W_\mathbf{x}$ to a positive-definite matrix and $\mathbf{x}_k = \mathbf{x}^\mathrm{neut}$ for all $k = 0, 1, \ldots, N$ causes the robot to be slowly "pushed" towards the "neutral" state $\mathbf{x}^\mathrm{neut}$ *during* output tracking. The

neutral state is far from the axis limits, to maximize the available range to track future inertial reference signals. Ideally, washout motion should not be perceived by the participant in the cabin.

D. Implementation of the MPC controller

We use the Real Time Iteration (RTI) scheme [9] to solve (11) online. The RTI scheme is based on a Sequential Quadratic Programming (SQP) scheme. At each sampling time the RTI scheme solves a single quadratic problem (QP) which is an approximation of the original non-linear problem. The rationale is that it is better to return an approximate solution quickly, than to calculate a high-accuracy solution while the system is changing and the solution is getting outdated.

The optimization problem (11) is a linearly constrained problem with a non-linear least squares objective. In the RTI scheme, we use the Gauss-Newton approximation of the Hessian of the objective function, which only requires calculation of the first order derivatives of the nonlinear output function $\mathbf{y}$. We use the `CasADi` framework [10] to generate C code to calculate the derivatives of $\mathbf{y}$ and the time-derivatives of $T_{\mathrm{H}}^{\mathrm{W}}(\mathbf{q})$ in the output function (10).

The optimization variables of the QP in the RTI scheme are the differences $\Delta\mathbf{u}_k$, $\Delta\mathbf{x}_k$ from the solutions obtained on a previous time step:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{x}_k^{\text{prev}} + \Delta\mathbf{x}_k,\ k = 0 \dots N \\ \mathbf{u}_k &= \mathbf{u}_k^{\text{prev}} + \Delta\mathbf{u}_k,\ k = 0 \dots N-1\ .\end{aligned}$$

To help the optimization convergence (in the spirit of Levenberg–Marquardt method), we penalize the control input differences by adding the term $\sum_{k=0}^{N-1} \beta\|\Delta\mathbf{u}_k\|^2$ to the QP objective. We found also that by choosing a proper value for the Levenberg–Marquardt term β we can achieve much smoother control action without significantly affecting tracking performance.

To reduce latencies, the RTI scheme divides calculations into a 'preparation' and a 'feedback' step. The preparation step involves all tasks that can be performed before the state estimate $\tilde{\mathbf{x}}_0$ is updated, including the calculation of the output function Jacobians, the cost function gradients and the Hessians. The feedback step comprises initial value embedding and QP solution, which is performed by the `HPMPC` [11] solver. The axis position increment is calculated according to (4) and sent directly after the feedback phase, after which the preparation phase for the next step begins.

The reference output trajectory $\hat{\mathbf{y}}$ consisting of N points sampled at interval T was provided to the controller at each $t_s = 12\,\mathrm{ms}$. A Kalman filter provided an estimation of the state, $\tilde{\mathbf{x}}_0$, each time step based on the state space model (2), previous control inputs, and sensor data. The MPC controller was implemented in C++ and running in a single thread on an `Intel® Xeon® CPU E5-1630` @2.67 GHz and `4.8.0-59-generic Linux` kernel.

E. Constraint tightening

Constraint tightening was implemented to cope with potential infeasibility of the optimization problem when operating near the state limits. Problem infeasibility is especially likely to occur during the first few steps, because 1) the controller cannot influence the initial states too much due to limits on the inputs, and 2) noise and model imperfections can cause the initial state estimate $\tilde{\mathbf{x}}_0$ to be unexpectedly close to or even outside the limits, from which recovery is not possible. The controller is given tighter constraints for steps far into the future, to ensure that the planned trajectory easily satisfies the actual constraints, and then given the actual constraints for the first few steps, to give room to recover from unexpected deviations. This is implemented by altering the state limits as follows:

$$\begin{aligned}\mathbf{x}_{\min,k} &= \mathbf{x}_{\min} + \alpha\frac{\min(k, N_{\text{tight}})}{2N_{\text{tight}}}(\mathbf{x}_{\max} - \mathbf{x}_{\min}) \\ \mathbf{x}_{\max,k} &= \mathbf{x}_{\max} - \alpha\frac{\min(k, N_{\text{tight}})}{2N_{\text{tight}}}(\mathbf{x}_{\max} - \mathbf{x}_{\min})\ ,\end{aligned} \tag{14}$$

where $0 \le \alpha < 1$. Constraint tightening does not *guarantee* problem feasibility for every $\tilde{\mathbf{x}}_0$, but works well in practice if N_{tight} and α are chosen appropriately.

IV. Experimental MPC setup

A hardware-in-the-loop experiment was performed to validate the proper implementation of the controller and to assess the output tracking performance for a variety of reference signals $\hat{\mathbf{y}}$. The synthetic reference signals consisted of square pulses and 'doublets' applied to each individual acceleration channel and triangle pulses and 'triangle-doublets' applied to each angular velocity channel, see Fig. 2. The maximum amplitudes A of the synthetic test signals were selected to match amplitudes commonly seen in realistic car and helicopter maneuvers: $A_{f_x} = A_{f_y} = 4\,\mathrm{m/s^2}$, $A_{f_z} = 2\,\mathrm{m/s^2}$, $A_{\omega_x} = A_{\omega_y} = 0.3\,\mathrm{rad\,s^{-1}}$, $A_{\omega_z} = 1\,\mathrm{rad\,s^{-1}}$. Each signal was applied with its maximum and its half amplitude, and with the duration of 1 s and 3 s. We also used two realistic test signals: a 332 s car city drive recording and a 223 s helicopter hover recording.

The signals were sampled at $t_s = 12\,\mathrm{ms}$ intervals. The prediction horizon was, however, sampled at $m = 4$ times longer interval of $T = 48\,\mathrm{ms}$, and thus the reference signals were down-sampled before sending them to the controller. To prevent aliasing effects, the signals were low-pass filtered off-line to 5 Hz using a zero-lag first-order Butterworth filter.

The resulting inertial signal was measured by a `STIM300` inertial measurement unit (IMU) installed in the cabin. The position and orientation of the IMU w.r.t. the cabin were included in (9). As a measure of the reference tracking performance, we use the root mean square (RMS) of the difference between the components of the reference inertial signal $\hat{\mathbf{y}}$ and the actual inertial signal $\mathbf{y}^{\text{imu}}$.

The diagonal weighting matrices $W_{\mathbf{u}} = \mathrm{diag}(\mathbf{w}_{\mathbf{u}})^2$, $W_{\mathbf{x}} = \mathrm{diag}\left([\mathbf{w}_{\mathbf{q}}^\top, \mathbf{w}_{\mathbf{v}}^\top]^\top\right)^2$ were used, with the numerical values summarized in Table III. The values in $W_{\mathbf{y}}$ were

selected to approximately equalize the variance of specific force in $\mathrm{m/s^2}$ and angular velocity in $\mathrm{rad\,s^{-1}}$ as observed in recorded car and helicopter maneuvers. Previous research [12] suggests that these values result in acceptable realism and are a good starting point for future research. The rationale behind the selected values for $W_{\mathbf{u}}$, $W_{\mathbf{x}}$ and $W_{\mathbf{x}_N}$ is to limit the simulators' motion as little as possible whilst maintaining stability. The neutral state $\mathbf{x}^{\text{neut}}$ corresponds to a 'neutral' position in the middle of the workspace, with the participant oriented upright. The reference input $\hat{\mathbf{u}}_k$ is equal to 0 for all k.

TABLE III
SELECTED MPC PARAMETERS

Axis (unit $\cdot$)	$\mathbf{w_u}$ [$\mathrm{s^2}/\cdot$]	$\mathbf{w_q}$ [$/\cdot$]	$\mathbf{w_v}$ [$\mathrm{s}/\cdot$]	$\mathbf{w_{q}}_N$ [$/\cdot$]	$\mathbf{w_{v}}_N$ [$\mathrm{s}/\cdot$]	$\mathbf{x}^{\text{neut}}$ [$\cdot$]
0 (m)	0.200	0.120	0	0.030	10	5
1 (rad)	0.200	0	0	0.030	10	1.571
2 (rad)	0.200	0.200	0	0.030	10	-1.096
3 (rad)	0.200	0.200	0	0.030	10	0.747
4 (rad)	0.200	0.200	0	0.030	10	0
5 (rad)	0.200	0.200	0	0.030	10	0.349
6 (rad)	0.200	0.200	0	0.030	10	0
7 (m)	0.200	100	5000	100	5000	1.550

The motion limits of all axes used in the experiment are presented in Table II. The prediction horizon length was set to $N = 60$, the tightening parameters to $N_{\text{tight}} = 8$, $\alpha = 0.10$ and Levenberg–Marquardt term to $\beta = 1.0$.

V. Experimental results I with plain MPC

Here, we will present the results for a limited set of signals, but note that a dataset is made available online that contains responses to other synthetic and realistic signals.

A. Example responses to test signals

Fig. 2(a) shows an example of the system's behavior in response to a doublet in the f_y channel. The reference $\hat{f}_y$, the expected output f_y, and the measured output f_y^{imu} are plotted. Note that the Y axis of the HF was aligned perpendicular to the linear rail, and thus all axes are involved in tracking this reference. An example response to a triangle-doublet signal in ω_z channel is shown on Fig. 2(b).

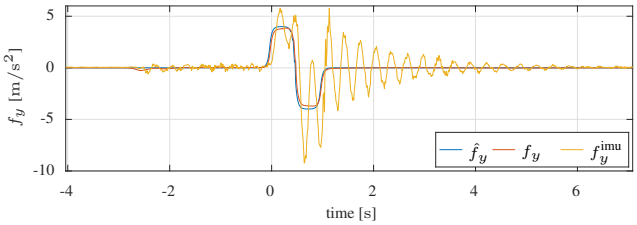
B. Motion of the axes

The position, velocity and acceleration of the axes corresponding to the example on Fig. 2(a) are presented in Fig. 3.

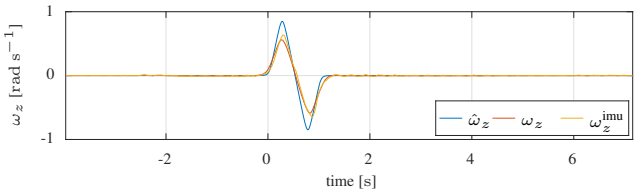
1) Preparation for the motion: The controller is seen to prepare the robot during the $2.88\,\mathrm{s}$ prior to the onset of the lateral doublet for two main reasons.

First, the selected maneuver requires pure lateral movement of the cabin, which is impossible in the selected neutral state of the robot due to the alignment of axis 5, which should be aligned vertically to be moved in synchrony with axis 1. Therefore, the alignment of axis 5 is changed from horizontal to vertical by rotating axis 4 by $\pi/2$ rad, and counter-rotating axis 6 by $-\pi/2$ rad to keep the cabin upright, see Fig. 3(a).

Second, output tracking is improved by moving away from the neutral state prior to the maneuver. That is, a doublet in the lateral acceleration channel results in a non-zero lateral translation in position. The range by which the robot can translate laterally is limited, and thus output tracking is improved by slowly moving in the opposite direction prior to the maneuver. This is seen mainly in axes 1 and 3.



(a) Doublet in f_y.



(b) Triangle-doublet in ω_z.

Fig. 2. Example responses to synthetic signals.

2) Axis limits: For the given example, the reference can not be tracked perfectly, due to the limits imposed on the axes. Fig. 3(c) shows that axes 1 and 5 reach their positive and negative acceleration limits during the doublet, and axis 4 reaches its negative acceleration limit for $-0.15\,\mathrm{s} < t < 1.04\,\mathrm{s}$.

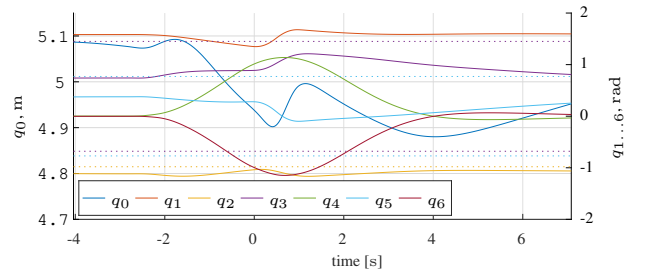
C. Mechanical oscillations

Fig. 2(a) shows large oscillations in measured specific force, suggesting that the mechanical system behaves as an underdamped mass-spring-damper system. In contrast, from Fig. 2(b) we can see that the oscillations in the angular velocity channel are negligible.

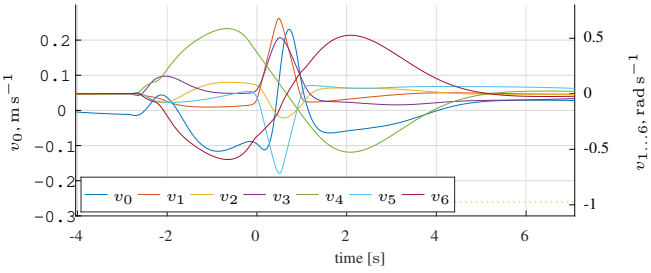
The serial robot is, essentially, a large mass ($\approx 500\,\mathrm{kg}$) attached to a slender arm, and thus an oscillatory response is to be expected. The oscillations seen in this example are large and would be perceived by the participant inside the cabin, but note that such a violent doublet is not commonly seen in realistic reference signals.

VI. Mitigating the vibration

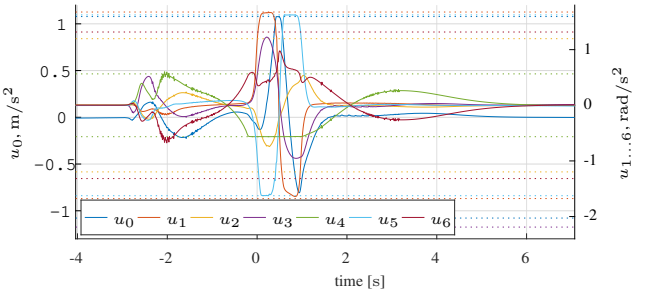
Although the entire system is non-linear, we found that a) the mapping from the expected specific force $\mathbf{f}$ to the measured specific force $\mathbf{f}^{\text{imu}}$ signal can be described by a second-order LTI model with reasonable accuracy, and b) the cross-talk between different components of the specific force is insignificant. We identified the transfer functions from $\mathbf{f}$ to $\mathbf{f}^{\text{imu}}$ using the `tfest` routine from MATLAB's System Identification Toolbox. The transfer functions, natural frequencies ω_n, damping factors ζ and time constants τ are presented in Table IV. These dynamics essentially describe a mass-spring-damper system with a natural frequency in the



(a) Axis position trajectory.



(b) Axis velocity trajectory.



(c) Axis acceleration trajectory.

Fig. 3. State and control input trajectories for the case of doublet-shaped reference tracking in f_y channel. The dotted lines show axis motion limits.

TABLE IV

IDENTIFIED TRANSFER FUNCTIONS FROM THE EXPECTED SPECIFIC FORCE TO THE ACTUAL SPECIFIC FORCE.

Channel	H(s)	ω_n [Hz]	ζ	τ [s]	Fit [%]
f_x	$e^{-0.096s}\frac{0.98s^2+19s+480}{s^2+4.1s+480}$	3.5	0.094	0.49	82
f_y	$e^{-0.072s}\frac{0.51s^2+16s+420}{s^2+1.8s+420}$	3.3	0.045	1.1	83
f_z	$e^{-0.072s}\frac{0.92s^2+14s+440}{s^2+3.6s+430}$	3.3	0.087	0.56	72

range $(3.3\text{–}3.5)$ Hz and a damping ratio of $(0.045\text{–}0.094)$. This gives us also an upper-bound estimate of the delay between sending the reference signal to the controller and its appearance in the physical output of approximately 0.1 s, including all communication delays.

In order to compensate the vibration, we change our model such that the output signal $\mathbf{y}$ is calculated taking the vibration into account. This requires adding to the state vector $n_v = 2$ extra components per each of the directions X, Y, Z, which comprise the *vibration state* $\mathbf{w} \in \mathbb{R}^6$. The augmented state vector is then $\mathbf{x}^{\text{aug}} = [\mathbf{x}^\top, \mathbf{w}^\top]^\top$, and the augmented version of the system dynamics (5) becomes

$$\mathbf{x}^{\text{aug}}_{k+1} = \begin{bmatrix} A & \mathbb{0} \\ \mathbb{0} & A_{\text{v}} \end{bmatrix} \mathbf{x}^{\text{aug}}_k + \begin{bmatrix} B & \mathbb{0} \\ \mathbb{0} & B_{\text{v}} \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \mathbf{f}_{\text{H}}(\mathbf{x}_k, \mathbf{u}_k) \end{bmatrix} \quad (15)$$

and the new output function

$$\mathbf{y}_{\text{v}}(\mathbf{x}^{\text{aug}}, \mathbf{u}) = \begin{bmatrix} \mathbb{0} & C_{\text{v}} \\ \mathbb{0} & \mathbb{0} \end{bmatrix} \mathbf{x}^{\text{aug}} + \begin{bmatrix} D_{\text{v}} & \mathbb{0} \\ \mathbb{0} & \mathbb{1} \end{bmatrix} \begin{bmatrix} \mathbf{f}_{\text{H}}(\mathbf{x}, \mathbf{u}) \\ \boldsymbol{\omega}_{\text{H}}(\mathbf{x}, \mathbf{u}) \end{bmatrix}, \quad (16)$$

where $A_{\text{v}} \in \mathbb{R}^{6\times6}$, $B_{\text{v}} \in \mathbb{R}^{6\times3}$, $C_{\text{v}} \in \mathbb{R}^{3\times6}$, $D_{\text{v}} \in \mathbb{R}^{3\times3}$ are matrices corresponding to discrete-time state-space representation of the identified vibration model.

Note that the model (15) is non-linear because of the non-linear function $\mathbf{f}_{\text{H}}$. This makes the downsampled model (6) also nonlinear, and requires m evaluations of model (15) to calculate the next state and the sensitivities. The model (6) enters the optimization problem (11) as an equality constraint, making the constraint nonlinear too. No additional constraints are placed on the vibration state.

VII. EXPERIMENTAL RESULTS II WITH VIBRATION MITIGATION

In order to validate our approach to mitigate the vibration, we conducted another series of tests with the augmented model from Section VI and the same settings as in Section IV. Fig. 4 shows the response of the system with augmented controller for the same condition as on Fig. 2(a). The reduction of vibration is clearly seen.

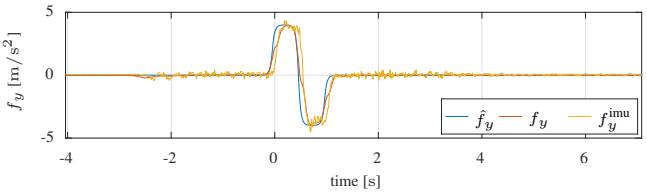


Fig. 4. Response to a doublet in f_y channel with vibration compensation.

To quantify the difference between the simple model (SM) and the augmented model (AM), we calculated the RMS of the tracking error for different test signals for both models. Before calculating the RMS, the measured inertial signal was shifted backwards in time by 72 ms to account for the identified delay. The RMS values are presented in Table V. The rows of the tables correspond to different test signals, and the columns correspond to the components of the inertial signal; the row labeled with "f_x" denotes all synthetic signals applied to the f_x channel, and so on.

The data show that the tracking error reduces or stays at the same level for AM compared to SM for all combination of test signals and output channels. The most significant change is seen in the f_y channel when the synthetic signal is applied to it. In this case, the use of the augmented model reduces the tracking error by a factor of 2.2.

For the two realistic signals there is little improvement in the tracking performance with AM compared to SM. The explanation is that the realistic signals have their energy concentrated below the simulator's natural frequency, and

TABLE V

RMS TRACKING ERROR.

(a) Specific force RMS tracking error in $\mathrm{m/s^2}$.

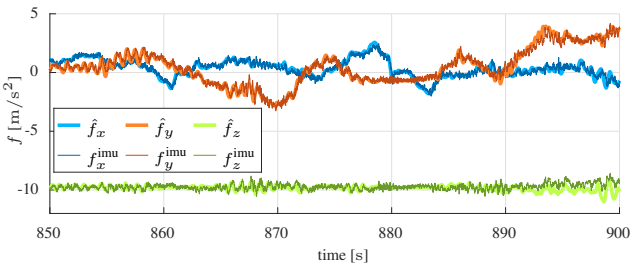
Signal	f_x		f_y		f_z	
	SM	AM	SM	AM	SM	AM
f_x	0.26	0.21	0.10	0.092	0.23	0.16
f_y	0.16	0.12	0.47	0.21	0.19	0.15
f_z	0.11	0.097	0.084	0.076	0.35	0.34
ω_x	0.089	0.081	0.15	0.13	0.098	0.095
ω_y	0.13	0.13	0.077	0.077	0.096	0.096
ω_z	0.13	0.12	0.24	0.18	0.14	0.13
car	0.24	0.22	0.28	0.28	0.28	0.26
heli	0.15	0.13	0.15	0.13	0.14	0.14

(b) Angular velocity RMS tracking error in $^\circ\,\mathrm{s}^{-1}$.

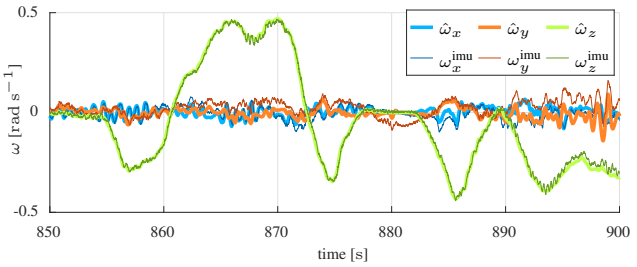
Signal	ω_x		ω_y		ω_z	
	SM	AM	SM	AM	SM	AM
f_x	0.18	0.17	1.5	1.4	0.17	0.14
f_y	1.4	1.4	0.53	0.32	1.1	0.55
f_z	0.13	0.12	0.38	0.28	0.12	0.11
ω_x	1.2	1.2	0.21	0.20	0.33	0.29
ω_y	0.11	0.11	1.1	1.1	0.098	0.096
ω_z	0.60	0.50	0.36	0.32	1.1	1.0
car	1.7	1.7	2.7	2.7	0.62	0.64
heli	0.47	0.47	0.44	0.42	0.34	0.33

therefore the vibrations are not excited. An example of realistic inertial signal (car drive) tracking is presented on Figure 5.

The RTI preparation phase took 5 ms. The feedback phase time varied depending on the number of iterations performed by the `HPMPC` solver, with the maximum time being 5 ms.



(a) Specific force tracking.



(b) Angular velocity tracking.

Fig. 5. Example of a realistic reference trajectory tracking (car drive).

VIII. CONCLUSIONS

This paper presented the real-time implementation of a Motion Cueing Algorithm based on Model Predictive Control for a motion simulator based on an 8-DOF serial robot. The results demonstrate that the reference signal is adequately tracked if it is within the capabilities of the robot, and the measured inertial signal resembles the expected inertial signal. Mechanical oscillations are seen in responses to reference signals that contain fast-changing acceleration, which excites the system's natural frequency. We found that these oscillations can be modelled by a second-order LTI system with reasonable accuracy. By including the oscillation model in the MPC we were able to significantly reduce the oscillations. Further improvement can be made by feeding back the IMU data to the controller in order to obtain more accurate estimate of the vibration state.

The use of proper algorithms and software allowed the computations to be done in real-time even with the augmented model.

ACKNOWLEDGMENT

This research was supported by the EU via FP7-ITN-TEMPO (607 957) and H2020-ITN-AWESCO (642 682), by the German Federal Ministry for Economic Affairs and Energy (BMWi) via eco4wind and DyConPV, and by DFG via Research Unit FOR 2401.

REFERENCES

[1] N. J. I. Garrett and M. C. Best, "Driving simulator motion cueing algorithms – a survey of the state of the art," in *Proc. of the 10th Int. Symp. on Advanced Vehicle Control (AVEC)*, 2010, pp. 183–188.

[2] R. J. Telban and F. Cardullo, "Motion cueing algorithm development: Human-centered linear and nonlinear approaches," *NASA TechReport*, no. May, pp. CR–2005–213 747, 2005.

[3] M. Dagdelen, G. Reymond, A. Kemeny, M. Bordier, and N. Maïzi, "Model-based predictive motion cueing strategy for vehicle driving simulators," *Control Engineering Practice*, vol. 17, no. 9, pp. 995–1003, Sept. 2009.

[4] Z. Fang and A. Kemeny, "Explicit MPC motion cueing algorithm for real-time driving simulator," in *Conference Proceedings - 2012 IEEE 7th International Power Electronics and Motion Control Conference - ECCE Asia, IPEMC 2012*, vol. 2, 2012, pp. 874–878.

[5] A. Beghi, M. Bruschetta, and F. Maran, "A real time implementation of MPC based Motion Cueing strategy for driving simulators," in *Proc. of the IEEE Conf. on Decision and Control*, 2012, pp. 6340–6345.

[6] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Pub., 2009.

[7] F. M. Nieuwenhuizen and H. H. Bülthoff, "The MPI cybermotion simulator: A novel research platform to investigate human control behavior," *J. of Computing Science and Engineering*, vol. 7, no. 2, pp. 122–131, 2013.

[8] C. Masone, P. R. Giordano, and H. H. Bülthoff, "Mechanical design and control of the new 7-DOF CyberMotion Simulator," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 4935–4942.

[9] M. Diehl, H. G. Bock, J. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[10] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," PhD thesis, KU Leuven, October 2013.

[11] G. Frison, H. B. Sørensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *European Control Conf. 2014 (ECC)*, 2014, pp. 128–133.

[12] D. Cleij, J. Venrooij, P. Pretto, M. Katliar, H. Blthoff, D. Steffen, F. Hoffmeyer, and H.-P. Schner, "Comparison between filter- and optimization-based motion cueing algorithms for driving simulation," *Transportation Research F: Traffic Psychology and Behaviour*, 2017.