Joint Stiction Avoidance with Null-Space Motion in Real-Time Model Predictive Control for Redundant Collaborative Robots

Salt Ducaju, Julian M.; Olofsson, Bjorn; Robertsson, Anders; Johansson, Rolf

Published in:
2021 30th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN 2021

DOI:
10.1109/RO-MAN50785.2021.9515514

2021

Document Version:
Peer reviewed version (aka post-print)

Joint Stiction Avoidance with Null-Space Motion in Real-Time Model Predictive Control for Redundant Collaborative Robots

Julian M. Salt Ducaju, Björn Olofsson, Anders Robertsson, Rolf Johansson

***Abstract*— Model Predictive Control (MPC) is an efficient point-to-point trajectory-generation method for robots that can be used in situations that occur under time constraints. The motion plan can be recalculated online to increase the accuracy of the trajectory when getting close to the goal position. We have implemented this strategy in a Franka Emika Panda robot, a redundant collaborative robot, by extending previous research that was performed on a 6-DOF robot. We have also used null-space motion to ensure a continuous movement of all joints during the entire trajectory execution as an approach to avoid joint stiction and allow accurate kinesthetic teaching. As is conventional for collaborative and industrial robots, the Panda robot is equipped with an internal controller, which allows to send position and velocity references directly to the robot. Therefore, null-space motion can be added directly to the MPC-generated velocity references. The observed trajectory deviation caused by discretization approximations of the Jacobian matrix when implementing null-space motion has been corrected experimentally using sensor feedback for the real-time velocity-reference recalculation and by performing a fast sampling of the null-space vector. Null-space motion has been experimentally seen to contribute to reducing the friction torque dispersion present in static joints.**

I. Introduction

Trajectory generation is a well-studied problem in the robotics field. It consists of defining the path and the course of motion as a function of time. An overview of the many ways for doing this task is provided in [1]. In an industrial setting, it is common to aim for performing a task in the shortest time possible to increase productivity. To this purpose, the robot should perform the given task under time constraints, making it convenient to formulate the problem as an optimal control problem, which provides a performance metric by means of an objective function [2].

Model Predictive Control (MPC) [3], [4] is a well-grounded option for trajectory generation in robotic applications, since its formulation can include a final-state constraint to be satisfied at the end of its prediction horizon while respecting states' and inputs' limits during the motion. MPC uses a model of the robot to predict the future states and outputs based on the solution's choice of the input sequence. In the presence of an internal controller with a short time constant considering the robot dynamics, position or velocity references can be used directly making a complex dynamic model not necessary in the MPC. Therefore, a purely kinematic model can be used.

Moreover, online MPC trajectory recalculation can be performed to increase the resolution of the computed trajectory by setting a fixed final time while keeping the number of discretization points of the MPC prediction horizon constant. Then, the continuous-time prediction horizon of the MPC problem will shrink, successively causing a reduction of the sampling period every time the trajectory is recalculated online [5].

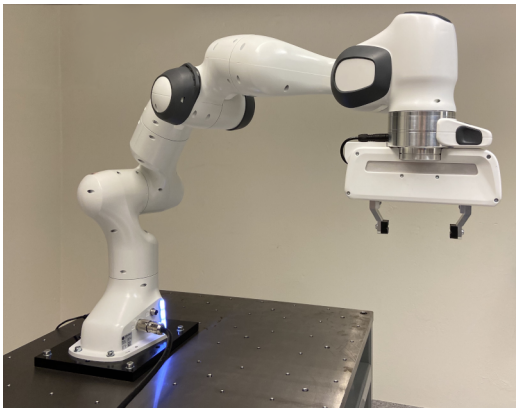

Fig. 1. Franka Emika Panda robot used in the experiments.

In the context of robot trajectory reprogramming, it is convenient that a human operator guides the robot through direct interaction [6], which is known as kinesthetic teaching [7] or lead-through programming (LTP), throughout the entire trajectory or parts of it. For an operator to be able to teach the robot, it is necessary to apply force on the robot end-effector or links. In this situation, the human operator should be comfortable with the physical interaction with the robot. Thus, it is important to be familiar with the force/torque required for leading the robot. Furthermore, the online MPC trajectory recalculation scheme is useful in the presence of human-robot interaction (HRI) since, after the human intervention is over, it can still be possible to reach the robot's goal pose without violating the problem's fixed-time constraint.

Therefore, the necessary force should not vary greatly between different human interventions. In order to ensure that the force that the operator needs to apply is always similar, joint stiction should be avoided. Joint static friction, or stiction, occurs when a joint has zero velocity and it becomes locked and constrained against relative motion [8]. This phenomenon is caused by the interactions between the

The authors are members of the ELLIIT Strategic Research Area at Lund University. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

J. M. Salt Ducaju, julian.salt_ducaju@control.lth.se, B. Olofsson, A. Robertsson, and R. Johansson are with the Department of Automatic Control, LTH, Lund University, Lund, Sweden.

asperities of the surface in contact in a robot joint, such as gears, bearings, and shafts [9].

Dithering has been proven as a successful method to reduce these uncertainties [10]. However, it may cause vibration of the robot if the torque feedforward signal's amplitude is too high. Another option to avoid stiction, only available for robots that have more than 6 degrees of freedom (DOF), is to use null-space motion [11]. Null-space motion is a linear combination of joint angular velocities in an over-actuated robot that causes no change in the end-effector's pose ($Jb = 0$ with $b \neq 0$, being J the Jacobian and b the null-space vector) [12]. It has previously been used in kinesthetic teaching to modify the robot's configuration without altering the end-effector's pose [13]. However, it can also be added to the trajectory reference to ensure that no joint remains still during the trajectory execution.

A trajectory generated for a 7 (or more) DOF robot may not necessarily involve varying the angular position of all of its joints, since it is a redundant system and it might be able to reach any end-effector's goal pose by only moving 6 of its joints. However, there could be an unexpected robot response if the operator tries to move a stationary joint since the force/torque required will be difficult to predict because of joint stiction [8]. The method that we propose to avoid joint stiction consists of adding null-space motion to an MPC-generated trajectory reference.

The purpose of this paper is to experimentally analyze the effects of adding null-space motion to an MPC-generated point-to-point trajectory reference to evaluate the possible advantages and drawbacks of this method. Moreover, HRI, facilitated by the addition of null-space motion, would allow the operator to locally modify the robot's path, which could be relevant in this context since the trajectory reference is generated considering only an initial and a final point. The implementation has been performed on the Panda robot by Franka Emika [14], a collaborative robot [15], which can be seen in Fig. 1.

Furthermore, in previous research [5], an open-loop strategy was implemented for the online trajectory recalculation where the initial state of the robot used to solve the MPC problem was estimated using the previous MPC solution. However, the addition of null-space motion to the MPC-generated reference may increase the error in the initial state estimation at every trajectory recalculation period, or metaperiod, and this error will accumulate at every online recalculation. For this reason, it is also a goal of this paper to evaluate the influence of joint angular position sensor feedback in the estimation of the initial state of the robot at the beginning of every metaperiod when adding null-space motion to an MPC-generated trajectory reference. The use of sensor feedback is referred to as the closed-loop strategy, as opposed to the previously used open-loop strategy [5].

This paper is outlined as follows: Sec. II presents the method for solving the problem that is being considered. Section III explains the experimental setup and the experiments performed, and presents the results obtained. Finally, conclusions are drawn in Sec. IV.

II. Methods

In this section, we introduce the MPC formulation used for trajectory generation and explain a strategy to add null-space motion to it. The goal of the trajectory-generation formulation used is that the robot reaches a final configuration under a time constraint. Additionally, we outline a hybrid dual-mode controller that would allow to switch between an MPC-based trajectory-following controller with null-space motion, and an admittance controller for human interaction.

A. Trajectory Generation Using MPC

The motion plan generated by MPC consists of a sequence of joint angular velocity references, since the robot's internal controller takes care of applying the necessary torques to each of the joints. Therefore, the optimization problem can be formulated in the joint space of the robot, using the robot's joint configuration $q \in \mathbb{R}^7$ since the robot has 7 joints, instead of formulating the problem in terms of the robot end-effector pose, $\xi \in SE(3)$, which is composed by the end-effector's position and orientation.

The initial and final joint configurations, q_0 and q_F, of the problem are obtained from the initial and desired end-effector poses, ξ_0 and ξ_F, respectively, by means of inverse kinematics [16]:

$$q = \mathcal{K}^{-1}(\xi) \tag{1}$$

Since this problem considers a 7 DOF robot, there will be an infinite number of solutions. Therefore, redundancy can be conveniently exploited to meet additional constraints on the kinematic control problem in order to obtain greater manipulability in terms of manipulator configurations, interaction with the environment, and null-space motion.

Moreover, since the robot is equipped with an internal controller that allows a velocity-reference control mode and we assume good tracking performance without exceeding the torque limits [17], the MPC does not need to use a complex nonlinear robot dynamic model where the torque is the input, and a simpler kinematic linear model is considered where the motion is defined in terms of position, velocity, and other higher-order time derivatives of position [18]. Also, the internal controller reduces the effect of dynamic coupling between joints by means of torque feedforward.

Then, as in previous research [5], the continuous-time model chosen can be constructed by multiple decoupled chains of integrators. Thus, the continuous-time state vector, $x_c \in \mathbb{R}^{21}$, is composed by the angular position, q_i, velocity, $\dot{q}_i$, and acceleration, $\ddot{q}_i$, of each of the robot joints $i = 1, \ldots, 7$:

$$x_c = \begin{bmatrix} q_1 & \dot{q}_1 & \ddot{q}_1 & \ldots & q_7 & \dot{q}_7 & \ddot{q}_7 \end{bmatrix}^{\mathrm{T}} \tag{2}$$

The continuous-time linear model can thus be written as:

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) \tag{3}$$

$$y_c(t) = C_c x_c(t) \tag{4}$$

with

$$A_c = \mathrm{blkdiag}([\tilde{A}_c, \ldots, \tilde{A}_c]), \quad B_c = \mathrm{blkdiag}([\tilde{B}_c, \ldots, \tilde{B}_c])$$

and $C_c = I_{21}$, where I_{21} is the identity matrix in $\mathbb{R}^{21\text{x}21}$, $\text{blkdiag}(\cdot)$ forms a block diagonal matrix from the given list of matrices, $A_c \in \mathbb{R}^{21\text{x}21}$, $B_c \in \mathbb{R}^{21\text{x}7}$, and

$$\tilde{A}_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \tilde{B}_c = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\text{T}}$$

The continuous-time input is the angular jerk of the joints, $u_c = \dddot{q} \in \mathbb{R}^7$.

For the choice of sampling period, h, to discretize the continuous-time linear system, a sampling period different from the one of the controlled system was chosen for the discretization of the kinematics in the optimization. Then, a linear interpolation of the calculated input sequence is used to provide references at the sampling rate of the robot [5]. This justifies the use of a predictive first-order-hold (FOH) sampling method [19]:

$$x_{k+1} = \Phi x_k + \frac{1}{h}\Gamma_1 u_{k+1} + \left(\Gamma - \frac{1}{h}\Gamma_1\right) u_k \tag{5}$$

$$y_k = C x_k \tag{6}$$

with

$$\Phi = \text{blkdiag}([\tilde{\Phi}, \ldots, \tilde{\Phi}]), \quad \Gamma_1 = \text{blkdiag}([\tilde{\Gamma}_1, \ldots, \tilde{\Gamma}_1]),$$
$$\Gamma = \text{blkdiag}([\tilde{\Gamma}, \ldots, \tilde{\Gamma}])$$

where $\Phi \in \mathbb{R}^{21\text{x}21}$, $\Gamma_1, \Gamma \in \mathbb{R}^{21\text{x}7}$, and:

$$\tilde{\Phi} = \begin{bmatrix} 1 & h & h^2/2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}$$
$$\tilde{\Gamma} = \begin{bmatrix} h^3/6 & h^2/2 & h \end{bmatrix}^{\text{T}}$$
$$\tilde{\Gamma}_1 = \begin{bmatrix} h^4/24 & h^3/6 & h^2/2 \end{bmatrix}^{\text{T}}$$
$$C = C_c$$

As developed in previous research [5], the discrete-time model obtained from the FOH sampling method (5), (6) can be rewritten in the standard form by using a new discrete state variable, $\zeta \in \mathbb{R}^{21}$:

$$\zeta_{k+1} = A\zeta_k + Bu_k \tag{7}$$

$$y_k = C\zeta_k + Du_k \tag{8}$$

where

$$A = \Phi, \quad B = \Gamma + \frac{1}{h}(\Phi - I_{21})\Gamma_1, \quad D = \frac{\Gamma_1}{h}$$

Since $y_k = x_k$ because of (6) and $C = I_{21}$, we can from (8) obtain the relation:

$$x_k = C\zeta_k + Du_k \tag{9}$$

It should be mentioned that the input u is the discretized counterpart of u_c, and the discrete controlled variable x is the discretized counterpart of x_c. On the contrary, the discrete-time state ζ is not a discretized version of any variable found in the continuous-time state-space system formulation (3), (4).

Moreover, the quadratic cost function chosen for solving this problem at time step k is:

$$V_k(U_k) = \sum_{j=k+1}^{k+H} x_j^{\text{T}} Q x_j + \sum_{j=k}^{k+H-1} u_j^{\text{T}} R u_j \tag{10}$$

where $U_k = [u_k, \ldots, u_{k+H-1}] \in \mathbb{R}^{7\text{x}H}$ is the input signal sequence over the control horizon of H steps that minimizes the cost function over the MPC prediction horizon of H steps at every metaperiod, and $Q \in \mathbb{R}^{21\text{x}21}$ and $R \in \mathbb{R}^{7\text{x}7}$ are positive semi-definite weight matrices that penalize the controlled variables and inputs, respectively.

This optimization problem is subject to the discrete-time model of the system (7), (9). Additionally, a hard constraint on the value of the discrete-time final controlled variables is used to ensure that the robot reaches the desired configuration at the end of the trajectory:

$$x_{k+H} = x_{\text{goal}} \tag{11}$$

In addition, a set of linear constraints must be included to bound the admissible range of the inputs and controlled variables:

$$F[u_k^{\text{T}}, \ldots, u_{k+H-1}^{\text{T}}]^{\text{T}} \leq f \tag{12}$$

$$G[x_{k+1}^{\text{T}}, \ldots, x_{k+H}^{\text{T}}]^{\text{T}} \leq g \tag{13}$$

The choice of the cost function as convex, as well as a linear model and convex constraint sets, makes the whole problem convex, which is beneficial for the computation of the problem since if a solution exists, it is the globally optimal [20].

Finally, this convex problem is solved at every trajectory recalculation period, or metaperiod. The sampling period, h, used in the discretization is equal to:

$$h = \frac{T_F - t_k}{H} \tag{14}$$

where H is the number of discrete steps in the prediction horizon, T_F is the final time where the goal state must be reached, and t_k is the time when the robot starts using the newly recalculated trajectory reference. As mentioned earlier, the continuous-time prediction horizon of the problem will shrink since, as time goes by, t_k will increase while the final time T_F and H are constant, thus increasing the resolution of the computed trajectory as the goal state, x_{goal}, is approached.

B. Null-Space Motion Addition to the Reference Trajectory

The manipulator's Jacobian matrix, $J(q) \in \mathbb{R}^{6\text{x}7}$, maps the joint angular velocities, $\dot{q}$, to the end-effector's twist, $\gamma = [\omega^{\text{T}}, v^{\text{T}}]^{\text{T}} \in \mathbb{R}^6$, with v and ω denoting the linear and angular velocity of the end-effector, respectively:

$$\gamma = J(q)\dot{q} \tag{15}$$

Therefore, null-space motion is constructed by using the null-space vector of this Jacobian matrix:

$$\dot{q} = J^{\dagger}(q)\gamma + N(q)\dot{q}_{\text{a}} \tag{16}$$

where the matrix $N(q) = I_7 - J^{\dagger}(q)J(q) \in \mathbb{R}^{7\text{x}7}$ projects the additional arbitrary joint angular velocity, $\dot{q}_{\text{a}}$, into the null space so that it is independent of the end-effector Cartesian motion [16].

The first term of (16) is the relationship between the joint velocity $\dot{q}$ and the end-effector's twist γ by means of the manipulator Jacobian (15), and superscript † denotes the Moore-Penrose pseudoinverse matrix given by $J^{\dagger} = (J^{\text{T}}J)^{-1}J^{\text{T}}$ [21]. This term is shared for both 6 and 7 DOF robots, although in the case of 6 DOFs, the Jacobian is a square matrix. However, the second term of (16) is the null-space motion, which only appears in redundant manipulators. The null-space motion unitary vector is calculated as:

$$\dot{q}_{\text{nsu}} = \frac{N(q)\dot{q}_{\text{a}}}{\|N(q)\dot{q}_{\text{a}}\|} \tag{17}$$

Also, since the Jacobian matrix is particular for each robot configuration, this vector should be sampled in real-time.

The null-space unitary vector given in (17) has to be scaled before being included with the MPC-generated angular velocity references. A sinusoidal signal has been chosen to smoothly transition between positive and negative scaling values to avoid reaching any joint limit. Its frequency depends on the length of the trajectory execution, to make sure that the first and last velocity references sent are equal to 0. Additionally, $\alpha \in \mathbb{R}$ is a constant used to scale its amplitude:

$$\dot{q}_{\text{NS}} = \dot{q}_{\text{nsu}}\alpha \sin\left(\frac{2\pi t}{T_F}\right) \tag{18}$$

Then, null-space motion is calculated at each robot sampling instant and added to the velocity references calculated by the optimization to avoid joint stiction:

$$\dot{q}_{\text{ref}} = \dot{q}_{\text{MPC}} + \dot{q}_{\text{NS}} \tag{19}$$

where $\dot{q}_{\text{ref}}$ is the velocity references sent to the robot, $\dot{q}_{\text{MPC}}$ is the linearly interpolated velocity reference sequence calculated by the MPC, and $\dot{q}_{\text{NS}}$ is the null-space motion component obtained from (18).

Moreover, the controlled-variable constraint (13) should consider the superposition of the null-space motion on the MPC solution to avoid any possible constraint violation. Therefore, when solving the MPC optimization the joint-velocity range should be reduced for every joint in a proportional way to the maximum possible joint-velocity component corresponding to the added null-space motion. With this approach, it is guaranteed that the joint-velocity limits are fulfilled. Also, the joint-acceleration range should be conservative to never exceed the joints' torque limits [17].

Finally, joint angular position sensor feedback can be used to reduce the mismatch between the estimation of the initial state used for the online optimization-problem calculation at every metaperiod and its real value caused by the addition of null-space motion to the MPC-generated trajectory. Therefore, a closed-loop form of the problem is proposed to obtain a more accurate estimation of the initial state to be used in the MPC. However, data samples from the robot's sensors cannot be directly used as the initial state, since there is a planned computational delay that accounts for the time required to solve the optimization problem in the MPC. Therefore, in order to provide a precise initial state estimation it is necessary to use the system's model described by (7) and (9) to estimate the state evolution between the sampling time of the sensor feedback and the time where the new trajectory velocity references are deployed to the system.

C. Human-Robot Interaction (HRI)

Even though the main focus of this research is to analyze the effects of a method that facilitates HRI by reducing joint stiction, we also provide an illustrative example of one possible way that a human operator can interact with the robot. Then, we outline a hybrid dual-mode controller where the robot receives commands from the MPC-generated trajectory that includes null-space motion (19), or from human-robot interaction, but never from both sources simultaneously, as summarized in Algorithm 1.

Since human input is, in this scenario, a path correction to the previously generated MPC trajectory reference, admittance control [13] is a suitable strategy for the human-interaction control mode. Another common human-robot interaction control strategy such as compliance control [22] is less appropriate for this application since its virtual spring component would try to bring the robot closer to the MPC reference rather than allowing the human to freely operate the robot.

If a joint-torque interface is available, a simple way to implement admittance control is to supply the robot with joint torque commands. For this, the rigid-body dynamics of the robot is used [12]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + \tau_{\text{fric}} = \tau_{\text{mot}} \tag{20}$$

where $M(q) \in \mathbb{R}^{7\text{x}7}$ is the generalized inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{7\text{x}7}$ describes the Coriolis and centripetal forces effects, $g(q) \in \mathbb{R}^{7\text{x}1}$ captures the gravity-induced torques, and $\tau_{\text{fric}} \in \mathbb{R}^{7\text{x}1}$ and $\tau_{\text{mot}} \in \mathbb{R}^{7\text{x}1}$ represent the friction and motor torques, respectively.

Then, if the admittance controller is active (Algorithm 1, Line 2), it will send commanding torques to joint motors that are equal to the sensed external joint torques, $\tau_{\text{ext}} \in \mathbb{R}^{7\text{x}1}$:

$$\tau_{\text{mot}} = \tau_{\text{ext}} \tag{21}$$

where part of the commanded joint torque is used for robot motion, but a fraction of the commanded joint torque is used to overcome joint friction, as seen in Eq. (20).

Friction is present in any element that involves relative motion in robot mechanisms. All friction models have in common a significant change of friction magnitude in the zero-velocity vicinity, as shown in Fig. 2, which is the major concern of friction compensation [23]–[26]. For this reason, avoiding joint stiction is helpful for the human operator that interacts with the robot to predict beforehand the necessary force that he/she should apply to the robot to achieve the desired displacement.

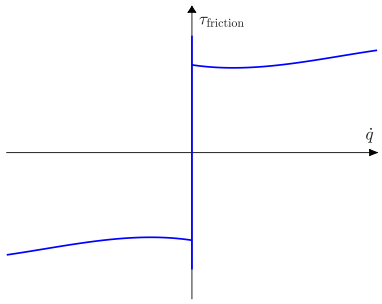

Fig. 2. Joint friction as a function of joint angular velocity.

Finally, a switching mechanism between both control modes (Algorithm 1, Line 1), trajectory following with null-space motion addition and admittance control, based on external torque sensor feedback, can be either automatic, following a collision detection and classification method (a summary of different strategies can be found in [27]), or manually determined by the human operator.

Algorithm 1 Hybrid Dual-Mode Controller

1: **if** human is interacting with robot **then**
2: *HRI mode*: Send τ_{ext} (21) as command to the robot's torque-reference interface.
3: **else**
4: *Trajectory-following mode*: Send $\dot{q}_{\text{ref}}$ (19) as command to the robot's velocity-reference interface.
5: **end if**

III. Experiments and Results

The experiments presented in this section evaluated the performance of the addition of null-space motion onto an MPC-generated trajectory.

A. Implementation and Experimental Setup

The robot used in the experiments is the Franka Emika Panda [14], a 7-DOF robotic arm. In addition to the joint velocity interface, the robot's internal controller also allows the operator to send joint position and torque commands. The Panda robot has a sampling rate of 1 kHz, and therefore, references should be sent to it every 1 ms. A photo of the robot used is shown in Fig. 1.

This collaborative robot, or *cobot* [15], is designed to share its workspace with humans in a safe manner, and it allows the human operator to set different maximum external-torque thresholds for each of the control modes so that if an accidental collision between the robot and the operator happened, the robot would perform a security shutdown.

As for the design choices for trajectory generation, the MPC prediction horizon was chosen to be equal to $H = 25$ as a trade-off between trajectory resolution and real-time computational cost, and the recalculation metaperiod was equal to 0.1 s. Also, the weighting matrix Q penalized the joint velocity and acceleration, but not the joint angular position, since there was no specific desired position between the initial and the final states [5], and since a hard constraint (11) was imposed on the final joint position, $Q = \text{blkdiag}([\tilde{Q}, \ldots, \tilde{Q}])$ where $\tilde{Q} = \text{diag}(\begin{bmatrix}0 & 1 & 1\end{bmatrix})$. Additionally, the input was less penalized than the states, $R = 0.001\ I_7$.

The first experiment presented analyzed the detrimental effects of slowing the sampling rate of the null-space vector of the Jacobian matrix. Then, the second experiment showed the suitability of using sensor feedback when adding null-space motion to MPC-generated trajectory references. Moreover, the third experiment focused on the results obtained for a closed-loop, fast null-space sampling approach where one of the joints would have remained static if null-space motion had not been included. Finally, the last experiment evaluated the dispersion of friction torque as a function of the joint angular velocity.

B. Experiment 1: Analysis of the effects of the null-space sampling rate on the trajectory accuracy

This experiment studied the effects of null-space discretization by performing the same trajectory in different runs, the only difference being that each run was performed at a different sampling rate of the null-space vector of the Jacobian matrix (1, 2, 5, and 10 ms). Since the null-space vector depends on the robot's configuration, a slower null-space sampling increases the difference between the null-space vector that is used for the velocity reference and the actual null-space vector.

The robot's initial configuration, randomly chosen, was, in radians:

$$q_0 = \begin{bmatrix}0 & -0.79 & 0.0 & -2.36 & 0.0 & 1.57 & 0.0\end{bmatrix} \quad (22)$$

Moreover, the trajectory lasted 10 s, enough time to clearly see the detrimental effects of a slower sampling rate of the null-space vector. Also, the trajectory consisted only of null-space motion, and the unitary null-space vector was scaled by a sinusoidal wave of frequency equal to 1 Hz and an amplitude constant, α, equal to 3 in (18). Therefore, at the end of the trajectory, the end-effector should ideally have the same pose as the initial one. Furthermore, an open-loop strategy was used for this experiment in an attempt to isolate the effects of having an insufficient sampling frequency of the null-space vector.

The temporal evolution of the robot's pose has been analyzed using Figs. 3 and 4. Figure 3 shows the temporal evolution of the end-effector's Cartesian coordinates and Fig. 4 the temporal evolution of the end-effector's orientation, by means of the Euler rotation angles (ZYX) from the robot base coordinate reference system to the end-effector's coordinate reference system. It can be observed how slowing the sampling rate caused the robot to drift from the desired constant end-effector's Cartesian pose.

Consequently, the null-space vector should be updated at the fastest update rate available, which in this case was the robot's sampling frequency (1 kHz). However, varying the null-space velocity references in intervals of 1 ms still

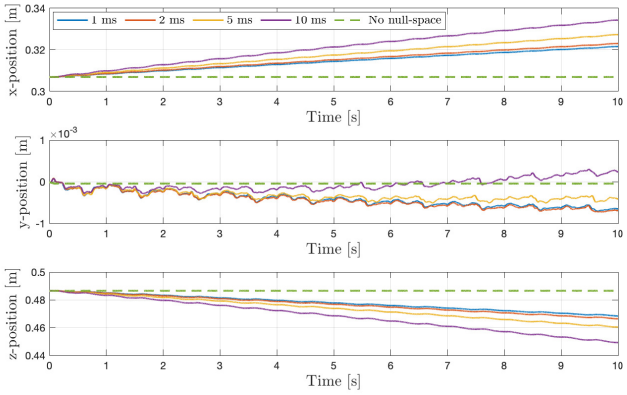

Fig. 3. Experiment 1 — End-effector's position with respect to base frame.

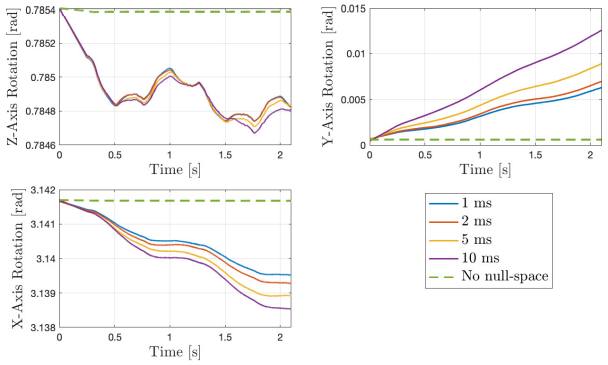

Fig. 4. Experiment 1 — End-effector's orientation with respect to base frame.

introduced a slight deviation from the planned trajectory, as seen in Figs. 3 and 4. For this reason, the next experiment considered a closed-loop approach to compensate the disturbances introduced by the approximate null-space motion.

C. Experiment 2: Comparison of the open-loop and the closed-loop strategies

Sensor feedback from joint position sensors can be used when updating the initial state estimation for online optimization in the MPC to account for the degrading effects of low-rate null-space sampling observed in the previous experiment. To show the benefits of using sensor feedback, it was necessary to compare the results of the implementation of null-space motion in the closed-loop MPC strategy versus the open-loop MPC strategy.

Several reference trajectories with different initial and goal robot configurations were used for this experiment. Additionally, each of them was executed five times. These trajectories combined null-space motion and MPC-generated trajectory references. Also, the null-space vector was sampled every ms and the sinusoidal scaling function's period was equal to the length of the trajectory.

The results of Experiment 2 are presented in Table I, which shows the mean and standard deviation of the end-effector's Cartesian position error at the end of the trajectory. The following expression was used for calculating this error:

$$e = \sqrt{(x_G - x_F)^2 + (y_G - y_F)^2 + (z_G - z_F)^2} \qquad (23)$$

where the subindex G refers to the goal position and the subindex F refers to the final position end-effector coordinate of the corresponding trial.

TABLE I
END-EFFECTOR'S FINAL CARTESIAN POSITION ERRORS [MM]

	CL - NS	*CL*	*OL - NS*	*OL*
Mean	1.43	0.48	5.97	1.18
Std. Dev.	0.78	0.25	0.39	0.11

In Table I, CL and OL refer to the closed-loop and the open-loop implementations, respectively, and NS to the runs that included null-space motion. Several assertions can be made based on Table I. First, when no null-space motion was added, the closed-loop strategy provided a more precise final Cartesian position, since there was a better initial state estimation at each MPC trajectory recalculation. Also, in both open-loop and closed-loop scenarios, including null-space motion was detrimental to the final state precision of the trajectory. Finally, using an open-loop strategy caused a greater total final Cartesian position error and therefore, if possible, a closed-loop scheme should be used to implement null-space motion.

D. Experiment 3: Null-space motion integration with closed-loop MPC in a trajectory that would have left one joint static

Once the two previous sets of experiments had shown the suitability of sampling the null-space vector as fast as possible and using a closed-loop control strategy to compensate for the degrading effects of adding null-space motion to an MPC-generated trajectory reference, the results for the closed-loop controller in one of the trajectories of Experiment 2 were analyzed.

Figure 5 shows how the addition of null-space motion modified the total velocity references (19). It can be seen that Joint 3 was not used in the MPC-generated trajectory, but it was desired to have it continuously moving to avoid its stiction, thus justifying the addition of null-space motion to the trajectory.

Figures 6 and 7 show the temporal evolution of the end-effector's position in Cartesian coordinates and the temporal evolution of the end-effector's orientation parameterized in the Euler rotation angles (ZYX) between the robot's base frame and the end-effector's frame, respectively. Even though the velocity references were different, null-space motion was properly implemented in the MPC trajectory generation since the temporal evolution of the end-effector pose was very similar in both trials, and it only showed slight deviations in the y-position in Fig. 6 and in the x-axis rotation in Fig. 7, which were compensated before the motion was finished. Therefore, joint stiction in Joint 3 was addressed by adding null-space motion, while still being able to perform an accurate trajectory under the task time constraints.

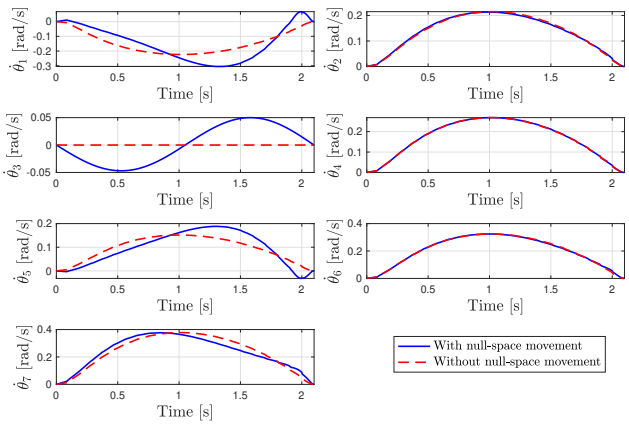

Fig. 5. Experiment 3 — Joint angular commanded velocities' evolution.

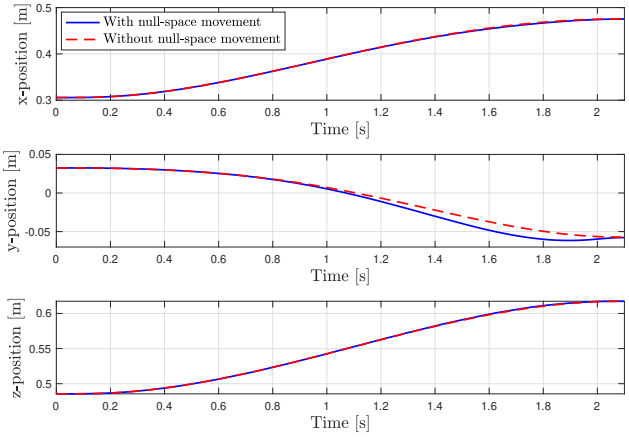

Fig. 6. Experiment 3 — End-effector's position with respect to base frame.

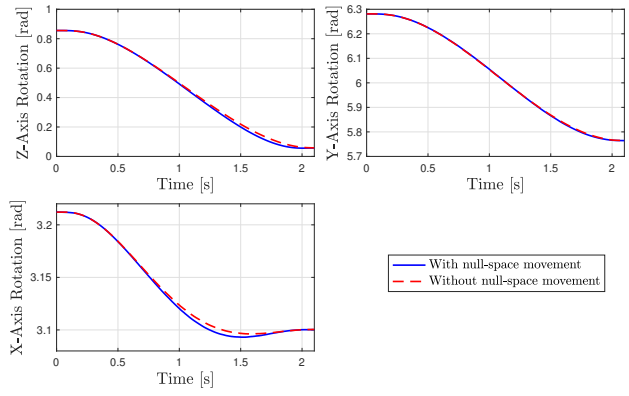

Fig. 7. Experiment 3 — End-effector's orientation with respect to base frame.

E. Experiment 4: Friction torque dispersion

The final experiment evaluated the dispersion of the friction torque in a joint as a function of its angular velocity. For this purpose, the torque-based admittance controller in Sec. II-C was implemented, so that the commanded torque to each of the joints was equal to their sensed external torque signals. Also, the friction torque for all joints was estimated by rewriting Eq. (20) as:

$$\hat{\tau}_{\text{fric}} = \hat{\tau}_{\text{ext}} - (M(\hat{q})\hat{\ddot{q}} + C(\hat{q}, \hat{\dot{q}})\hat{\dot{q}} + g(\hat{q})) \tag{24}$$

where the superscript $\hat{}$ denotes a variable that has been estimated using joint position or torque sensor data.

Then, the experiment consisted of a human operator leading-through the robot for 15 s using this torque-based admittance controller. Figure 8 shows the results in terms of the standard deviation, σ, of the estimated friction torque of a joint in Eq. (24) with respect to its angular velocity.

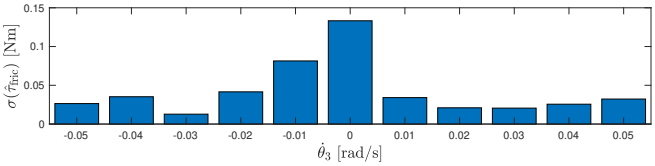

Fig. 8. Experiment 4 — Standard deviation of the friction torque as a function of angular velocity of Joint 3.

The choice of joint and its angular-velocity range, shown in Fig. 8, was related to the experiment presented in Sec. III-D, since the motion of Joint 3 was generated only by null-space motion and had the same angular velocity range, as seen in Fig. 5.

It can be seen that the standard deviation of the friction torque was greater when the angular velocity of the joint was close to zero: $\sigma(\hat{\tau}_{\text{fric}}) = 0.13$ Nm in the vicinity of zero velocity compared to an average value of $\sigma(\hat{\tau}_{\text{fric}}) = 0.03$ Nm in the rest of the angular velocity range analyzed. Therefore, adding null-space motion to a static joint can contribute to reducing the friction torque dispersion.

IV. Conclusion

We have proposed the addition of null-space motion to an MPC fixed-time point-to-point online trajectory generation method in order to facilitate kinesthetic teaching in a redundant robot. This approach allows a continuous motion of all joints throughout the trajectory execution, even if the MPC-generated trajectory does not include them in its planning, so that joint stiction is suppressed and a human operator can predict the force/torque necessary to move the robot. A reduction of the friction-torque dispersion has been experimentally observed as a consequence of adding null-space motion in a static joint.

The discrete-time control of null-space motion has been observed to be sensitive to discretization approximations of the Jacobian matrix. The experiments performed have justified the extension of a previously studied open-loop scheme [5] to a closed-loop scheme and a fast Jacobian matrix

sampling to correct these slight degrading effects on the trajectory execution performance, thus allowing the addition of null-space motion to the trajectory without causing a significant loss of final-state accuracy.

An additional benefit of the presented closed-loop strategy is that, if human intervention takes place during the trajectory execution, the trajectory can be recalculated online once human intervention is concluded using an accurate estimation of the initial state in the MPC problem.

REFERENCES

[1] T. Kröger, *On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*. Springer, Berlin, Germany, 2010, vol. 58.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, Cambridge, UK, 2006.

[3] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[4] J. M. Maciejowski, *Predictive Control: with Constraints*. Prentice Hall, Harlow, UK, 2002.

[5] M. Ghazaei Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Model predictive control for real-time point-to-point trajectory generation," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 972–983, Apr. 2019.

[6] M. Capurso, M. M. G. Ardakani, R. Johansson, A. Robertsson, and P. Rocco, "Sensorless kinesthetic teaching of robotic manipulators assisted by observer-based force control," in *IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 29 May–2 Jun. 2017, pp. 945–950.

[7] S. Wrede, C. Emmerich, R. Grünberg, A. Nordmann, A. Swadzba, and J. Steil, "A user study on kinesthetic teaching of redundant robots in task and configuration space," *Journal of Human-Robot Interaction*, vol. 2, no. 1, pp. 56–81, 2013.

[8] E. J. Haug, S. C. Wu, and S. M. Yang, "Dynamics of mechanical systems with Coulomb friction, stiction, impact and constraint addition-deletion—I Theory," *Mechanism and Machine Theory*, vol. 21, no. 5, pp. 401–406, 1986.

[9] A. C. Bittencourt and S. Gunnarsson, "Static friction in a robot joint—Modeling and identification of load and temperature effects," *J. Dynamic Syst., Measurement, and Control*, vol. 134, no. 5, 2012.

[10] M. Linderoth, A. Stolt, A. Robertsson, and R. Johansson, "Robotic force estimation using motor torques and modeling of low velocity friction disturbances," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 3–7, 2013, pp. 3550–3556.

[11] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Task-space control of robot manipulators with null-space compliance," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 493–506, 2013.

[12] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, Berlin, Germany, 2016.

[13] A. Wahrburg, J. Boes, B. Matthias, F. Dai, and H. Ding, "Sensorless null-space admittance control for redundant manipulators," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*. Munich, Germany: VDE, 21–22 Jun. 2016, pp. 1–7.

[14] *Panda – Data Sheet*, Franka Emika, 2019.

[15] J. E. Colgate, J. Edward, M. A. Peshkin, and W. Wannasuphoprasit, "Cobots: Robots for collaboration with human operators," in *Proceedings of the ASME Dynamic Systems and Control Division: ASME International Mechanical Engineering Congress and Exposition*, Atlanta, GA, USA, Nov. 17–22, 1996.

[16] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, 1st ed. Springer, Berlin, Germany, 2013.

[17] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 18–22, 2010, pp. 2592–2599.

[18] M. Ghazaei, "On Trajectory Generation for Robots," Ph.D. dissertation, Dept. Automat. Control, Lund Univ., Lund, Sweden, Nov. 2016, TFRT-1116.

[19] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Courier Corporation, Massachusetts, USA, 2013.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ. Press, Cambridge, UK, 2004.

[21] A. Ben-Israel and T. N. Greville, *Generalized Inverses: Theory and Applications*. Springer-Verlag, New York, USA, 2003.

[22] N. Hogan, "Impedance control: An approach to manipulation: Part I," *J. Dynamic Syst., Measurement, and Control*, vol. 107, no. 1, pp. 1–7, 1985.

[23] L. Cai and G. Song, "A smooth robust nonlinear controller for robot manipulators with joint stick-slip friction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1993, pp. 449–454.

[24] D. Karnopp, "Computer simulation of stick-slip friction in mechanical dynamic systems," *J. Dynamic Syst., Measurement, and Control*, vol. 107, no. 1, pp. 100–103, 1985.

[25] L. Freidovich, A. Robertsson, A. Shiriaev, and R. Johansson, "LuGre-model-based friction compensation," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 1, pp. 194–200, 2009.

[26] A. Shiriaev, A. Robertsson, and R. Johansson, "Friction compensation for passive systems based on the LuGre model," in *2nd IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control*, Seville, Spain, Apr. 3–5, 2003, pp. 183–188.

[27] G. Cioffi, S. Klose, and A. Wahrburg, "Data-efficient online classification of human-robot contact situations," in *European Control Conference (ECC)*. Saint Petersburg, Russia: IEEE, May 12–15, 2020, pp. 608–614.