Auto-Tuning of Model Predictive Control for Bilateral Teleoperation with Bayesian Optimization

Fadi Alyousef Almasalmah, Hassan Omran, Chao Liu, Thibault Poignonec, Bernard Bayle

Auto-Tuning of Model Predictive Control for Bilateral Teleoperation with Bayesian Optimization [⋆]

Fadi Alyousef Almasalmah [*] **Hassan Omran** [*] **Chao Liu** [**] **Thibault Poignonec** [*] **Bernard Bayle** [*]

[*] *ICube lab, University of Strasbourg, France, (e-mail: {alyousef, homran, tpoignonec, bernard.bayle}@unistra.fr)*
[**] *LIRMM, University of Montpellier-CNRS, France, (e-mail: chao.liu@lirmm.fr)*

Abstract: Model Predictive Control (MPC) is becoming a popular control method for teleoperation due to its ability to ensure safety constraints. However, tuning MPC is a non-intuitive process that requires significant expertise and effort. In this work, we propose a method for auto-tuning a model predictive controller in bilateral teleoperation settings. We use the Bayesian Optimization algorithm (BO) to seek the optimal weights of the MPC cost function for precise teleoperation. Our simulations and experiments show the effectiveness of the proposed tuning method.

Keywords: Bilateral Teleoperation, Model Predictive Control, Controller Auto-Tuning, Bayesian Optimization, Physical Human-Robot Interaction.

1. INTRODUCTION

Recently, Model Predictive Control (MPC) has gained more attention in teleoperation applications due to its success in solving practical problems, especially the ability to respect constraints, which is essential for guaranteeing safety. In addition, MPC is an online optimization-based method with the potential to manage the compromise between safety and transparency in a better way than the controllers that separate those two goals (Piccinelli and Muradore, 2020). MPC also offers several ways to improve the closed-loop performance from data (Hewing et al., 2020). This is why a growing number of authors are using MPC in the teleoperation domain, such as Cheng et al. (2022); Piccinelli and Muradore (2020); Almasalmah et al. (2023); Sheng and Spong (2004).

Despite its importance, the choice of the MPC parameters is rarely an intuitive task and it requires theoretical knowledge, experience, and iterative tuning through trial and error. The MPC cost function in teleoperation is often designed to reduce the matching errors between the operator and remote side robots in terms of position, velocity, and force. Previous studies, such as those by Piccinelli and Muradore (2020) and Sheng and Spong (2004), have explored methods for optimizing these errors. In addition, the cost function usually contains a term related to the control inputs. However, it should be noted that accurately matching forces and matching positions are two competing objectives, such that the controller must compromise between the two (Hashtrudi-Zaad, 2000). Assigning the weights in the MPC cost function may not always give the intended results in the actual behavior due to several reasons, such as model errors, short prediction horizon, and the choice of control input weights. Thus, an efficient method for auto-tuning is still needed.

The MPC tuning process can be seen as an optimization problem whose goal is to minimize a certain metric that evaluates the controller performance. This metric can depend implicitly on the controller parameters, such as the MPC cost function. Unfortunately, such a metric does not have a simple analytical form and it can only be evaluated point-wise by performing experiments (or simulations). Hence, black-box optimization methods were used in the teleoperation literature for controller tuning. For example, early work by Kress and Jansen (1992) used the Hooke and Jeeves search method to search for optimal PID gains for a simulated 2-Degrees-Of-Freedom (DOF) robot. Other authors have used Particle Swarm Optimization (PSO) (Shokri-Ghaleh and Alfi, 2014; Alfi et al., 2014), genetic algorithms (Kim and Ahn, 2009), or Artificial Bee Colony Algorithm (Said et al., 2019). Evolutionary learning Neural Networks were used by Talavatifard et al. (2006) to learn the optimal gains for a PID controller based on the current environment. Barbé et al. (2006) used a method called relay auto-tuning to tune the PID controller of the operator robot in a medical needle insertion scenario. Most of the aforementioned methods require a large number of experiments to find the optimal tuning, which limits their applicability to simulations. Optimal tuning obtained in simulations can be applied to the real system, but this might make it suboptimal due to model-plant mismatch.

Bayesian Optimization (BO) is a black-box optimization method that has become particularly popular due to its sample efficiency and ability to deal with noise-corrupted

[⋆] This work was supported by the Investissements d'Avenir program (ANR-21-CE33-0004, ANR-11-LABX-0004, Labex CAMI) and by Région Grand Est doctoral program.

objective functions. The algorithm iteratively balances the exploration of promising regions and exploitation of known high-performing areas by fitting and exploiting a surrogate model of the objective function. A review of the method can be found in Shahriari et al. (2016). Recently, BO has been applied to tune controllers in many contexts. For instance, Marco et al. (2016) auto-tuned an LQR controller for balancing an inverted pendulum held by a robotic arm in hardware experiments. Holzmann et al. (2024) used BO to find energy-efficient trajectories for a robotic manipulator by tuning the MPC weights. Dries et al. (2017) auto-tuned a robot controller to interact with unknown objects robustly without causing any damage. Zahedi et al. (2022) designed a user-adaptive variable damping controller for human-robot interaction tasks and used BO to find the optimal user-specific parameters.

In this work, we use the BO algorithm to auto-tune an MPC controller for bilateral teleoperation tasks. Therefore, safety guarantees can be enforced by the MPC controller, while ensuring an accurate teleoperation by finding the optimal MPC weights. The method balances the trade-off between position tracking and force tracking for the operator and remote robots. Moreover, the sample efficiency of the method makes it applicable for tuning on hardware directly with a relatively low number of experiments as demonstrated through our simulations and experiments.

The paper is organized as follows: Section 2 describes the model of the teleoperation system and the MPC controller design. Section 3 presents a brief introduction to the Bayesian Optimization algorithm. In Section 4, an overview of our approach is given. The simulations and hardware experiments are compared in Section 5. Finally, Section 6 concludes the paper.

2. SYSTEM MODELLING AND CONTROLLER DESIGN

2.1 Teleoperation system model

We consider a teleoperation system that consists of the following components: the human operator, the operator robot (or the haptic device), the communication channel, the centralized controller, the remote robot, and the environment. Considering two general multi-DOF robots, each robot's dynamics can be decoupled by feedback linearization, allowing us to view each single axis separately as a 1-DOF robot. Therefore, in the rest of the paper, we consider two 1-DOF robots, and the method can be generalized for more DOFs. The 1-DOF operator robot dynamic equation is given by:

$$m_o a_o + b_o v_o + k_o x_o = f_h + u_o \tag{1}$$

where a_o, v_o, x_o are the acceleration, velocity, and position of the operator robot, respectively, m_o, b_o, k_o are the mass, damping coefficient, and stiffness of the operator robot, respectively, u_o is the control force applied by the robot, and f_h is the force applied by the human hand. Assuming a linear time-invariant environment model, the dynamics of the remote robot that is in contact with the environment are described by:

$$\begin{aligned} m_r a_r + b_r v_r + k_r x_r &= f_e + u_r \\ f_e &= -k_e x_r - b_e v_r \end{aligned} \tag{2}$$

where a_r, v_r, x_r are the acceleration, velocity, and position of the remote robot, respectively, m_r, b_r, k_r are the mass, damping coefficient, and stiffness of the remote robot, respectively, u_r is the control force applied by the remote robot motor, f_e is the force applied by the environment, and b_e, k_e are the damping coefficient and stiffness of the environment. The remote robot dynamics with the environment can be written as:

$$m_r a_r + (b_e + b_r) v_r + (k_e + k_r) x_r = u_r \tag{3}$$

In this work, the environment parameters k_e and b_e are assumed constant and known. The discretized system model can be written from (1) and (3) as:

$$\begin{aligned} x_{t+1} &= A x_t + B u_t \\ y_t &= C x_t + D u_t \end{aligned} \tag{4}$$

where $(\cdot)_t$ refers to the discretized variable $(\cdot)$ at time instant t, $x \in \mathbb{R}^5 = [x_o, v_o, x_r, v_r, f_h]^\top$ is the state vector, where f_h is considered constant in the prediction model, but in practice, it is measured and updated at each time step, $u \in \mathbb{R}^2 = [u_o, u_r]^\top$ is the control input vector, matrices A, B, C, D are known constant matrices that depend on the robots and environment models, and $y \in \mathbb{R}^6 = [x_o, v_o, x_r, v_r, f_h, f_e]^\top$ is the measured output vector. This model constitutes the basis of the following MPC controller design.

2.2 MPC controller design for teleoperation

Equation (4) is used as a prediction model in the MPC controller. Knowing the state x_t, the state trajectory is predicted on a horizon of N steps by deciding which inputs $[u_t, \ldots, u_{t+N-1}]$ will be applied. Here, we assume f_h is constant during the horizon, as in Piccinelli and Muradore (2020). Our goal is to maximize the teleoperation transparency, which means the matching between positions and forces of the operator and the remote robots, while also respecting any safety constraints that could be imposed on system inputs, states, and outputs. To evaluate the transparency, we define the following vector:

$$z_t = E y_t + F u_t \tag{5}$$

where $z \in \mathbb{R}^3 = [x_o - x_r,\ v_o - v_r,\ u_o - f_e]^\top$ is the matching error vector and E, F are known constant matrices. It should be noted that we minimize $(u_o - f_e)$ in the formulation instead of $(f_h - f_e)$ since f_h is an external input to the system that we can only measure. Indeed, it is desired to reflect the environment force f_e to the operator. To achieve this, we follow the work of Sheng and Spong (2004) and minimize $(u_o - f_e)$, assuming u_o will be close to the actual force perceived by the human if the operator robot has low impedance and friction. We can now formulate the MPC optimization problem as follows:

$$\begin{aligned} [u_0^*, \ldots, u_{N-1}^*] = \underset{u_0,\ldots,u_{N-1}}{\arg\min} \sum_{k=0}^{N-1} & \left(\hat{z}_k^\top Q \hat{z}_k + u_k^\top R u_k \right) \\ \text{s.t. } & \hat{x}_0 = x_t, \\ & \hat{x}_{k+1} = A\hat{x}_k + B u_k, \\ & \hat{y}_k = C\hat{x}_k + D u_k, \\ & \hat{z}_k = E\hat{x}_k + F u_k, \\ & u_{min} \le u_k \le u_{max}, \\ & G\hat{x}_k \le g, \end{aligned} \tag{6}$$

where Q is a positive semi-definite matrix, R is a positive-definite matrix, $(\hat{\cdot})$ refers to predicted variables, u_{min} and u_{max} are set based on the motor torque limits or for safety reasons, G and g are, respectively, a constant matrix and a vector that define linear constraints on the states. Note that unlike the traditional bilateral teleoperation architectures, where signals are directly sent and used by the controllers on each side, the MPC-based bilateral teleoperation uses the optimization problem (6) to connect the models (1) and (3) to minimize the matching errors in z_t (5).

At each time step t, the state is updated from the measurement and the problem (6) is solved. As usually done in MPC, only u_0^* is applied as input u_t, and the whole process is repeated at $t+1$. The auto-tuning method focuses on finding the best Q and R to achieve good teleoperation performance by minimizing the closed-loop matching errors of position, velocity, and force between both robots.

3. BAYESIAN OPTIMIZATION FOR AUTO-TUNING

In this section, we provide a brief background about the Bayesian optimization algorithm that we use to find the optimal weights Q, R of the MPC cost function. Interested readers are referred to (Garnett, 2023) for more details about the algorithm. We consider Q and R as diagonal matrices as in Fröhlich et al. (2022) to reduce the search space dimension, and we write $Q = diag(\theta_1, \theta_2, \theta_3)$, $R = diag(\theta_4, \theta_5)$ where $diag(.)$ is a diagonal matrix with the arguments on its main diagonal. Finally, we search for the optimal $\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5] \in \mathbb{R}^5$.

3.1 Background

We define a performance metric $\mathcal{J}(\theta)$ that measures the performance of a certain MPC controller parameterized by θ. The tuning process is to find:

$$\theta^* = \arg\min_{\theta\in\Theta} \mathcal{J}(\theta) \tag{7}$$

$\mathcal{J}(\theta)$ does not have a known analytical form, and the evaluation at a certain point θ could be done experimentally by evaluating the performance, which is time-consuming. Bayesian Optimization helps find θ^* efficiently with a low number of experiments. The algorithm constructs a surrogate function $\hat{\mathcal{J}}(\theta)$ that approximates $\mathcal{J}(\theta)$ in critical regions by sampling the latter. The optimization problem turns into iteratively fitting $\hat{\mathcal{J}}(\theta)$ to $\mathcal{J}(\theta)$ and minimizing $\hat{\mathcal{J}}(\theta)$ using more efficient methods such as derivative-based methods. To choose which points to evaluate $\mathcal{J}(\theta)$ at, the algorithm uses an acquisition function $\alpha(\theta)$ that balances the exploration-exploitation trad-off. Specifically, $\alpha(\theta)$ encourages exploring new regions in the space where the surrogate function has high uncertainty and is likely not close to the actual function. Conversely, $\alpha(\theta)$ promotes exploitation by selecting points that minimize the surrogate function, which are likely to approximate the minimizer of the true function $\mathcal{J}(\theta)$.

Gaussian Processes (GPs) are the most popular choice for the surrogate model, which allows for a closed-form inference of the posterior mean and variance based on noisy function values $\mathcal{D}_n = \{(\theta^i, \mathcal{J}^i = \mathcal{J}(\theta^i) + \epsilon^i)\}_{i=1}^n$ where $(\cdot)^i$ refers to the i-th evaluation, ϵ^i is the evaluation noise with $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. The GP is completely specified by its mean function $\mu(\cdot)$ and covariance function or kernel $k(\cdot,\cdot)$. Finally, the surrogate model can be modeled as:

$$\hat{\mathcal{J}}(\theta) \sim \mathcal{GP}\left(\mu(\theta), k(\theta, \theta')\right) \tag{8}$$

Based on the set of observations $\mathcal{D}_n$, and assuming a zero prior mean, we can query the GP at a new point θ and get the predicted mean and variance as follows:

$$\begin{aligned} \mu_n(\theta) &= \mathbf{k}_n^\top \mathbf{K}^{-1}\mathbf{y} \\ \sigma_n^2(\theta) &= k(\theta,\theta) - \mathbf{k}_n^\top \mathbf{K}^{-1}\mathbf{k}_n \end{aligned} \tag{9}$$

with $\mathbf{k}_n$ a kernel vector with entries $[\mathbf{k}_n]_i = k(\theta, \theta^i)$, and the kernel matrix $\mathbf{K}$ with entries $[\mathbf{K}]_{i,j} = k(\theta^i, \theta^j) + \delta_{i,j}\sigma_\epsilon^2$, where $\delta_{i,j}$ is the Kronecker delta and $i, j = \{1, \ldots, n\}$. Note that calculating the predicted mean and variance at a new point depends directly on all previously measured points. Given the predictive distribution in (9), the algorithm proposes the next point θ^{n+1} to be evaluated by minimizing the acquisition function. We use the standard Upper Confidence Bound (UCB) function given by:

$$\alpha_{\mathrm{UCB}}(\theta|\mathcal{D}_n) = \mu_n(\theta) - \beta\sigma_n(\theta) \tag{10}$$

with the exploration hyper-parameter β, where higher values of β encourage the exploration of unknown regions with high uncertainty in the GP, and lower values encourage finding the minimum using the current learned surrogate model. For the GP kernel, we employ the standard Matérn kernel 3/2, and infer its hyperparameters and noise level by evidence maximization (Rasmussen and Williams, 2006). In order to search for the optimal weights of the MPC, we need to define a bounded search space and a performance metric. The algorithm builds a surrogate GP model that captures the relation between the MPC weights and the value of the performance metric, which is used iteratively to find the optimal weights.

4. PROPOSED APPROACH OF AUTO-TUNING

In this section, we present the proposed approach to automatically tune the MPC weights using the BO algorithm in bilateral teleoperation.

4.1 Performance Metric Design

To compare different weights of the MPC, we design a function that measures the performance based on an experiment (or simulation). One of the advantages of BO is the ability to handle complex non-convex performance metrics. Many examples of such functions were used in the literature in different applications. For example, in autonomous racing cars, Fröhlich et al. (2022) used lap time and the deviation from the center line as a performance metric. Catkin and Patoglu (2023) used human preference as a measure of performance to tune a haptic rendering system. Zahedi et al. (2022) used BO to adapt the damping in a physical Human-Robot Interaction (pHRI) scenario. The authors used a combination of agility measures, user effort, and stability-related measures such as overshoot and settling time. Dries et al. (2017) used compliance and force smoothness at the moment of contact to tune a robot that interacts with an uncertain environment. In teleoperation, the matching errors between the operator and the remote robots are often used. In this work, we use the matching errors of positions, velocities, and forces to measure the performance after each experiment i as:

$$\mathcal{J}^i = \frac{1}{T_{exp}} \sum_{j=1}^{T_{exp}} z_j^\top \mathcal{W} z_j \tag{11}$$

where $\mathcal{W}$ is a scaling matrix that takes into account measurement units and the relative importance between the signals as defined by the user, T_{exp} is the duration of the experiments measured in time steps, and z is defined in (5). In practice, we set an upper bound to prevent large values of the performance metric since a point with an excessively high value can deteriorate the accuracy of the GP fitting for neighboring low points. It is important to point out the difference between the MPC cost function (6) and the performance metric (11). Although both depend on the matching errors, the former minimizes the predicted errors based on the system model on a short horizon, while the latter evaluates the closed-loop measured errors on a much longer window in a model-independent manner. Note that the closed-loop errors are different from the predicted ones due to the limited prediction horizon and the potential model-plant mismatch. The MPC cost function also includes a cost for control inputs while the performance metric does not.

4.2 Auto-Tuning the MPC

We perform n_{exp} experiments of interaction with an environment with parameters k_e, b_e. Each experiment lasts for T_{exp} seconds. The human operator applies a force profile on the operator robot, while the remote robot interacts with the environment. Before each experiment i, the BO algorithm proposes a certain θ^i that contains the diagonal values of Q and R. After T_{exp} time steps, we calculate the performance metric as in (11), and we update the surrogate model with the new noisy measurement of the performance metric $\left(\theta^i, \mathcal{J}(\theta^i) + \epsilon^i\right)$ where ϵ is the unknown noise value included in the measurement. Using the updated mean and kernel functions, the next promising point is found by maximizing the acquisition function from (10). After testing n_{exp} different points, we can either take the best one as the optimal tuning, or we can find the minimizer of the learned surrogate (GP) model using gradient-based techniques with multiple restarts. The latter choice is preferred due to its robustness against noisy measurements and human input variability.

In this work, we assume that we know a range of parameters that can stabilize the system, which is used as the search space for the algorithm. In practice, theoretical stability guarantees can be achieved through other means, for instance, using so-called energy tanks (Piccinelli and Muradore, 2020). Fig. 1 shows the architecture of the teleoperation system with the proposed algorithm.

5. SIMULATION AND EXPERIMENT RESULTS

In this section, we show the results of the proposed algorithm for auto-tuning. We repeat the same process with a similar setup in simulation and hardware experiments and compare their results. The simulation is done to validate the proposed method under perfect conditions.

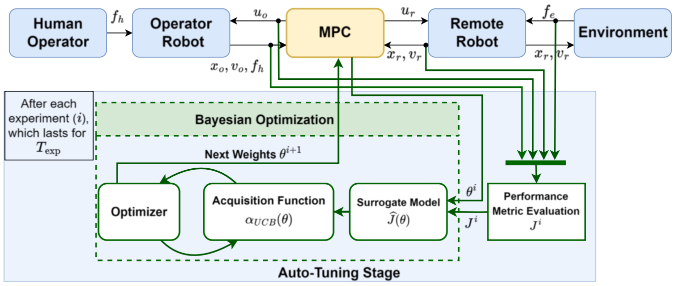

Fig. 1. Block diagram of the proposed auto-tuning approach with the teleoperation system

Each experiment (or simulation) involves 3 seconds of interaction with a low-frequency sinusoidal-like human force as input to the system ($1 \sim 2$ Hz). At the end of each interaction, we calculate the performance metric value $\mathcal{J}(\theta^i) + \epsilon^i$ and we assign it to the corresponding MPC weights, where the measurement noise ϵ^i can be caused, for instance, by human variability. The BO algorithm updates the surrogate model with the noisy measured value, and the acquisition function proposes new weights to test, which are then used in the next experiment. The process is repeated for n_{exp} trials until acceptable tuning is found.

5.1 Experiment and Simulation Setup

The goal of the tuning is to minimize the performance metric in (11). We set $\mathcal{W}$ from (11) as $\mathcal{W} = diag(1500, 5, 500)$, and we impose an upper bound of 10 on the performance metric value. In the BO algorithm, we set the exploration parameter in (10) to $\beta = 3$ initially, and decrease it linearly until 0.5 at the last point to encourage the exploitation. The BO algorithm also requires defining the bounds of the parameters, which is set as the element-wise inequalities:

$$[0, 0, 0] \leq [\theta_1, \theta_2, \theta_3] \leq [10^4, 10, 10^4]$$
$$[1, 1] \leq [\theta_4, \theta_5] \leq [10^3, 10^3]$$

where the search space of stabilizing parameters results from empirical tests and scaling of measuring units. The code is implemented in ROS2 framework under Linux. Acados library (Verschueren et al., 2021) is used to formulate and implement the MPC solver in C++. The BO code is based on the software in Fröhlich and Carron (2021).

The experimental setup includes two identical 1-DOF robots. Each robot is composed of a *Maxon* motor, which is controlled by *EPOS3* driver board, and connected to a high-precision encoder with 4000 readings per rotation. The motor shaft is connected to the robot joint by a cable-driven capstan mechanism with a reduction ratio of 10, as shown in Fig. 2. Finally, a load cell force sensor is embedded in the robot handle to measure the external force. The remote robot is in contact with a linear spring, which could be approximated by a rotational spring in the range of motion with stiffness $k_e = 0.7$ Nm.rad^{-1}.

The robot models were identified using *MATLAB* Identification Toolbox. The model parameters for (1) and (3) are: $m_o = m_r = 0.00092$ kg.m^2, $k_o = k_r = 0.015$ Nm.rad^{-1}, $b_o = b_r = 0.0032$ Nm.s.rad^{-1}. The remote robot is interacting with a spring with stiffness $k_e = 0.7$ Nm.rad^{-1}, and $b_e = 0$ Nm.s.rad^{-1}. The identified model parameters values are used in the simulation for comparison. All

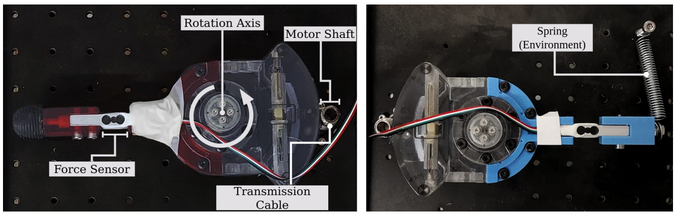

Fig. 2. The hardware setup. The operator robot is on the left, and the remote robot is on the right. The remote robot is attached to a spring (the environment).

forces and torques are projected on the joint axis and measured as torques (Nm). The sampling time is set to $T_s = 0.002$ s, and the MPC prediction horizon is set to 15 steps. We set a constraint on both control inputs as follows: -0.33 Nm $\leq u_o, u_r \leq 0.33$ Nm.

5.2 Simulation Results

The BO performs the auto-tuning using 9 simulated interactions, and the results are reported in Fig. 3. The top figure illustrates the performance metric computed at the end of each experiment. The first 3 points are chosen around the middle of the search space to initialize the surrogate model with no optimization yet. Afterward, a general downward trend can be noticed, indicating that the algorithm is finding better weights, with occasional rises due to exploration. In the bottom figures, we compare different tested weights, where on the left, position and force tracking are acceptable, and on the right, both position and force tracking are almost perfect with performance metric value close to zero. Fig. 4 displays the performance of the found optimal tuning by minimizing the learned surrogate model, where perfect tracking in position, velocity, and force matching can be observed. In the bottom right figure, we show the moving average of the performance metric, with a window of 3 seconds, which maintains a low value due to the good controller performance.

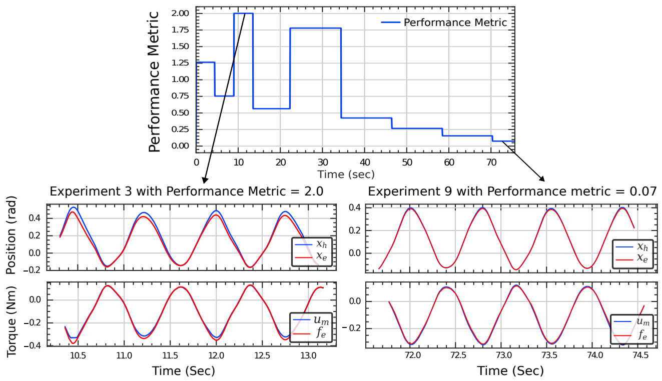

Fig. 3. Auto-tuning experiment in simulation.

5.3 Hardware Experiments Results

We show the efficiency of the algorithm in tuning the MPC through 14 experiments of 3 seconds each. Fig. 5 shows the performance metric, computed at the end of each experiment. The model is initialized with 3 experiments, after which the optimization starts and the performance metric starts decreasing. We show a comparison between different tested weights during the search for the optimal

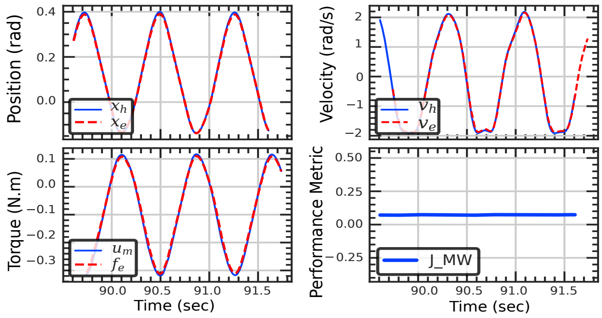

Fig. 4. Performance of the found optimal weights in simulation with near-perfect tracking.

value. The comparison displays the improvement in the matching between the robots positions (x_h, x_e) and forces (u_m, f_e). We present the performance of the found optimal controller in Fig. 6 with a 3-second moving average of the performance metric on the bottom right.

5.4 Discussion

Overall, the hardware experiment results are consistent with the simulation results. In both cases, the auto-tuning shows an improvement compared to the initial tuning, without prior knowledge about the role of each parameter on the closed-loop performance. The auto-tuning achieved nearly perfect tracking in simulation. In the hardware experiment, we observe that while tracking improves after tuning, it remains imperfect, due to unmodeled static friction. Addressing such nonlinearities requires more than tuning the MPC cost weights. However, since the goal of this work is to show the feasibility of the auto-tuning, handling unmodelled nonlinearities is beyond the paper's scope. We observe in Fig. 3 that even the initial performance is acceptable in simulation, unlike the initial weights in the hardware experiment in Fig. 5. This shows the importance of tuning directly on hardware compared to using the optimal weights found in simple simulations. Finally, the auto-tuning took around 80 seconds in hardware, which corresponds to 3 seconds for each experiment, plus 1.5 seconds of transient time between the experiments, and the rest is for the BO computations.

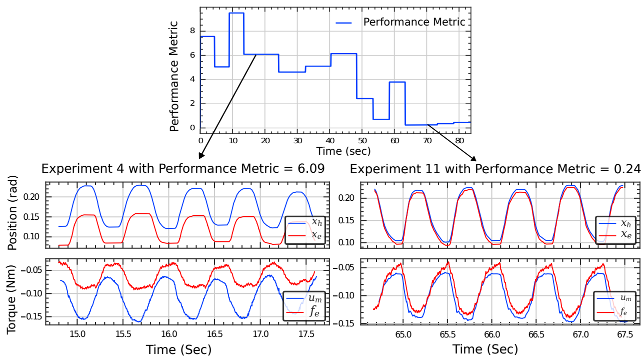

Fig. 5. Auto-tuning experiment with hardware.

6. CONCLUSION

In this work, we presented an approach for auto-tuning an MPC in a bilateral teleoperation scenario, which can be applied directly to hardware. We formulated the tuning problem as a black-box optimization and used the

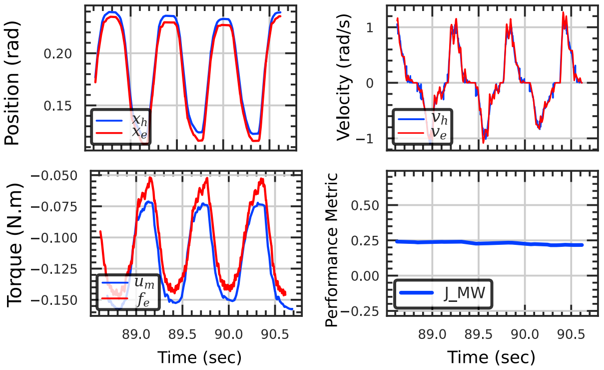

Fig. 6. Performance of the found optimal weights in hardware experiment.

Bayesian Optimization algorithm to efficiently search for the optimal MPC weights, which gave good results in a few minutes. In the future, we plan to extend the method to work with several environments, for example, by testing each proposed MPC weights on a range of environments and evaluating the performance metric accordingly. Another possibility is learning the optimal MPC weights as a function of the environment using Contextual Bayesian Optimization as was done by Fröhlich et al. (2022). Finally, incorporating the stability constraints in the BO problem itself would be an interesting perspective.

REFERENCES

Alfi, A., Khosravi, A., and Lari, A. (2014). Swarm-based structure-specified controller design for bilateral transparent teleoperation systems via μ synthesis. *IMA J. of Math. Control and Information*, 31(1), 111–136.

Almasalmah, F.A., Omran, H., Liu, C., and Bayle, B. (2023). Adaptive Robust Model Predictive Control for Bilateral Teleoperation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 7069–7074.

Barbé, L., Bayle, B., and de Mathelin, M. (2006). Towards The Autotuning of Force-Feedback Teleoperators. *IFAC Proceedings Volumes*, 39(15), 482–487.

Catkin, B. and Patoglu, V. (2023). Preference-Based Human-in-the-Loop Optimization for Perceived Realism of Haptic Rendering. *IEEE Trans. Haptics*, 16, 470–476.

Cheng, J., Abi-Farraj, F., Farshidian, F., and Hutter, M. (2022). Haptic Teleoperation of High-dimensional Robotic Systems Using a Feedback MPC Framework. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 6197–6204.

Dries, D., Englert, P., and Toussaint, M. (2017). Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations. In *IEEE Int. Conf. Rob. and Automation (ICRA)*, 902–907.

Fröhlich, L.P. and Carron, A. (2021). Bayesopt4ros: A Bayesian optimization package for the robot operating system. `https://github.com/IntelligentControlSystems/bayesopt4ros`.

Fröhlich, L.P., Küttel, C., Arcari, E., Hewing, L., Zeilinger, M.N., and Carron, A. (2022). Contextual Tuning of Model Predictive Control for Autonomous Racing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 10555–10562.

Garnett, R. (2023). *Bayesian Optimization*. Cambridge University Press, 1 edition.

Hashtrudi-Zaad, K. (2000). *Design, Implementation and Evaluation of Stable Bilateral Teleoperation Control Architectures for Enhanced Telepresence*. Ph.D. thesis, University of British Columbia.

Hewing, L., Wabersich, K.P., Menner, M., and Zeilinger, M.N. (2020). Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1), 269–296.

Holzmann, P., Maik Pfefferkorn, M., Peters, J., and Findeisen, R. (2024). Learning energy-efficient trajectory planning for robotic manipulators using bayesian optimization. *European Control Conference (ECC)*.

Kim, B.Y. and Ahn, H.S. (2009). Bilateral teleoperation systems using genetic algorithms. In *IEEE Int. Symposium on Computational Intelligence in Robotics and Automation - (CIRA)*, 388–393.

Kress, R. and Jansen, J. (1992). Automatic tuning for a teleoperated arm controller. In *IEEE Conf. on Decision and Control (CDC)*, 2692–2695.

Marco, A., Hennig, P., Bohg, J., Schaal, S., and Trimpe, S. (2016). Automatic LQR tuning based on Gaussian process global optimization. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 270–277.

Piccinelli, N. and Muradore, R. (2020). A passivity-based bilateral teleoperation architecture using distributed nonlinear model predictive control. *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 11466–11472.

Rasmussen, C.E. and Williams, C.K.I. (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass.

Said, A., Khalil, A., Petra, R., Yunus, S., Peng, A.S., and Khan, S. (2019). PID Controller Optimization of Teleoperated 2DOF Robot Manipulator Using Artificial Bee Colony Algorithm. In *IEEE Int. Conf. on Engineering Technologies and Applied Sciences (ICETAS)*, 1–6.

Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., and de Freitas, N. (2016). Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1), 148–175.

Sheng, J. and Spong, M. (2004). Model predictive control for bilateral teleoperation systems with time delays. In *IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE)*, volume 4, 1877–1880.

Shokri-Ghaleh, H. and Alfi, A. (2014). A comparison between optimization algorithms applied to synchronization of bilateral teleoperation systems against time delay and modeling uncertainties. *Applied Soft Computing*, 24, 447–456.

Talavatifard, H.A., Razi, K., and Menhaj, M.B. (2006). A Self-Tuning Controller for Teleoperation System using Evolutionary Learning Algorithms in Neural Networks. In B. Reusch (ed.), *Computational Intelligence, Theory and Applications*, 51–60. Springer, Berlin, Heidelberg.

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., and Diehl, M. (2021). acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*.

Zahedi, F., Chang, D., and Lee, H. (2022). User-Adaptive Variable Damping Control Using Bayesian Optimization to Enhance Physical Human-Robot Interaction. *IEEE Robotics and Automation Letters*, 7(2), 2724–2731.