Adaptive Robust Model Predictive Control for Bilateral Teleoperation

Fadi Alyousef, Hassan Omran, Chao Liu, Bernard Bayle

Adaptive Robust Model Predictive Control for Bilateral Teleoperation

Fadi Alyousef Almasalmah[1], Hassan Omran[1], Chao Liu[2], Bernard Bayle[1]

***Abstract*— In this work, we use recent developments in the field of adaptive robust Model Predictive Control (MPC) to build a controller for bilateral teleoperation systems. To guarantee robust constraint satisfaction, we incorporate polytopic tube controllers in the MPC design. In addition, we use online learning methods to learn the environment model. Namely, we use set membership learning to learn the parametric uncertainty bounds and reduce the conservatism of the robust controller, and we combine it with least mean square method to learn a point estimate of the model parameters, which enhances the controller performance. Our simulation demonstrates the effectiveness of the proposed approach in maintaining robust constraint satisfaction and enhancing performance by learning during teleoperation tasks.**

I. INTRODUCTION

Having force feedback in teleoperation enhances the human perception of the remote environment and makes the process more safe and intuitive. Many control schemes have been used to create safe controllers that maximize the transparency of teleoperated systems. Such schemes include robust control [1], adaptive control [2], time-domain passivity [3], energy-tanks [4], and Model Predictive Control (MPC) [5].

Recently, MPC has gained more attention in teleoperation applications due to its success in solving practical problems, namely the ability to respect constraints, which is essential for guaranteeing safety. In addition, MPC is an online optimization-based method, which has the potential to manage online the compromise between safety and transparency in a better way than the controllers that separate those two goals [5]. MPC has been used in various ways in the literature of teleoperation; for example, in [6] the author used the control sequence generated by an MPC to compensate for large time delays in a unilateral teleoperation scenario. The work in [7] built a single MPC on the operator side that generates a reference velocity for the remote robot's PI controller, and combined the MPC with a Smith-predictor to estimate the state of the remote robot and compensate for communication delay. In a recent paper [5], the authors designed a nonlinear MPC with energy tanks for both the operator and remote robots, and used the MPC to reduce the chattering that happens when the tank has low energy. Later in [8], the authors used the same framework and added a force constraint on the remote robot to prevent it from damaging the environment. The method used recursive least square to learn the environment model, starting from an overestimation of the model parameters. The authors expanded their work in [9] by using a hybrid linear MPC to reduce the number of switches between free motion and contact state, which cause tool bouncing at the moment of contact. However, a major limitation in the previously mentioned works is that the controller is prone to violate the constraints during the transient stage of parameters learning, or if the parameters are not learned accurately. In addition, the online optimization is not guaranteed to keep being feasible during the execution, which means that the problem might become infeasible at some point and the controller might fail.

To overcome these issues, robust learning-based MPC methods could be used. The idea of robust MPC methods is to consider a range of uncertainty in the model parameters and additive noise, and then to take cautious control actions to account for all possible values of uncertainty. Such methods include min-max optimization [10], and tube controllers [11]. The tube controller is the most widely used method due to its simplicity and lower computation complexity. The idea of this controller is to predict a nominal trajectory of the system by neglecting the uncertainties, and then to find a tube around the trajectory that contains all the possible values of the near-future states under the effect of uncertainty.

Robust MPC methods are quite conservative in the case of parametric uncertainty, which is the reason why many works combine it with learning from observations to reduce the conservatism. In [12], a tube controller was used for robustness alongside with model learning. However, the uncertainty model was fixed, so the method works only for small and additive uncertainty. In [13], [14], a method that combines tubes with Set Membership Learning (SML) is used to learn a tighter uncertainty description from observations and reduce the conservatism of the tubes. The method also uses an additional Least Mean Square (LMS) parameter estimate for boosting the performance. While the method in [14] is more conservative than the one in [13], it requires a much lower number of optimization variables, and hence, it is more suitable for real-time teleoperation.

In this paper, we design an Adaptive Robust MPC (AR-MPC) for bilateral teleoperation, which robustly and recursively guarantees constraint satisfaction. The method, which is based on the work in [14] learns the environment model from observations and reduces the conservatism and enhances the performance of the MPC.

[1]Fadi Alyousef Almasalmah, Hassan Omran, and Bernard Bayle are with ICube lab, University of Strasbourg, France, (alyousef, homran, bernard.bayle)@unistra.fr

[2] Chao Liu is with LIRMM Lab, University of Montpellier-CNRS, France. chao.liu@lirmm.fr

II. System Modelling

A. Teleoperation system model

The teleoperation system consists of five components: the human operator, the operator robot (the haptic device), the communication channel, the remote robot, and the environment. We consider two single-axis linearized robots, as in [5]. The dynamics of the operator robot 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 operator robot motor; f_h is the force applied by the human. The dynamics of the remote robot and the environment during the contact are described by:

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

where a_r, v_r, x_r are the acceleration, velocity, and position of the remote robot, respectively; b_e, k_e are the damping coefficient and stiffness of the environment; 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. We assume that the environment model is Linear Time-Invariant (LTI), and that the environment parameters are not precisely known initially, so we consider the parametric uncertainty:

$$\theta = [\Delta k_e, \Delta b_e]^\top \tag{3}$$

which is bounded by Θ_0, a known compact polytopic set that contains the true value of the parameters θ^*. The discrete model of the system could be written in the state-space form using (1) and (2) with the uncertainty from (3):

$$\begin{aligned} x_{t+1} &= A_\theta x_t + B u_t + B_d f_{h,t} + d_t \\ y_t &= C_\theta x_t + D u_t + D_d f_{h,t} \end{aligned} \tag{4}$$

where $(\cdot)_t$ refers to the discretized variable $(\cdot)$ at time instant t; $x = [x_o, v_o, x_r, v_r]^\top$ is the state vector; $u = [u_o, u_r]^\top$ is the control input vector; B, B_d, D, D_d are known constant matrices; $d \in \mathbb{D}$ is the additive uncertainty which is bounded by a known convex 4-dimensional polytope $\mathbb{D}$; $y = [x_o, v_o, x_r, v_r, f_e]^\top$ is the measured output vector. The matrices A_θ, C_θ are affine in the parameter θ from (3):

$$\begin{aligned} A_\theta &= A_0 + A_1[\theta]_1 + A_2[\theta]_2 \\ C_\theta &= C_0 + C_1[\theta]_1 + C_2[\theta]_2 \end{aligned} \tag{5}$$

with A_i, C_i are known constant matrices, and $[\cdot]_i$ is the $i-th$ element of the vector.

One crucial aspect of MPC is the ability to respect constraints on the state, output, and input signals. In teleoperation, the constraints are chosen to ensure the safety of the system and the users, such as limiting remote robot position, which could represent a virtual fixture in surgical teleoperation [15] or limiting the velocity of both robots according to safety standards [16], [17]. Other works limit the interaction force to prevent damaging the environment [8]. We consider the joint state-input constraints as follows:

$$Fx + Gu \leq \mathbf{1} \tag{6}$$

where F, G are constant matrices that define q constraints; $\mathbf{1}$ is a vector of ones with a suitable size.

III. Background: Adaptive Robust MPC

In this section, we will summarize the main concepts of adaptive-robust MPC literature, and focus on the method described in [14].

A. Robust tube controller

The main idea of tube controllers is to predict a nominal trajectory (with no uncertainty) over a horizon N and to build a tube around it that contains the future states under all possible uncertainty realizations. Robust constraint satisfaction is then guaranteed by ensuring that the whole tube satisfies the constraints. The tube section is often a polytopic set $\mathcal{P}_0 = \{x \in \mathbb{R}^n : H_\mathcal{P} x \leq \mathbf{1}\}$, which is found offline based on invariant sets [14]. To build the online predicted tube, we find a sequence of N sets $\mathcal{P}_{k=1\ldots N}$, by translating $\mathcal{P}_0$ to each predicted state x_k and scaling the set by a scalar s_k. To minimize the growth of the tube and guarantee the stability of the system, a prestabilizing state feedback controller with gain K, and a terminal cost matrix P are computed offline using Linear Matrix Inequalities (LMIs). The MPC then deals with prestabilized dynamics $A_{cl,\theta}$, where:

$$A_{cl,\theta} = A_\theta + BK \tag{7}$$

and computes an optimal additional control input.

B. Learning the model

1) Learning uncertainty bounds - Set Membership Learning: To reduce the conservatism of the robust tube controller, the parametric uncertainty bounds can be learned and reduced from measurements. Assuming the additive disturbance is bounded by a known polytopic set $d \in \mathbb{D}$ and that we have at time step t the following state and input measurements: $x_t, x_{t-1}, u_{t-1}, f_{h,t-1}$, we can find a set that contains the model true parameter θ^*:

$$\Delta_t = \{\theta \in \mathbb{R}^2 : x_t - (A_\theta x_{t-1} + B u_{t-1} + B_d f_{h,t-1}) \in \mathbb{D}\}$$

In other words, Δ_t is the set of all environment parameters $\theta = [\Delta k_e, \Delta b_e]^\top$ that could have caused the state to move from x_{t-1} to x_t under the influence of $u_{t-1}, f_{h,t-1}$, and an additive disturbance bounded by $\mathbb{D}$. Δ_t can be found by simple matrix operations. We then define the feasible parameter set Θ_t as the polytopic set that robustly contains the true parameters θ^*, which is given by:

$$\Theta_t = \Theta_0 \bigcap_{i=0}^{t} \Delta_i$$

If the bound $\mathbb{D}$ is tight enough, then Θ_t is likely to shrink with time. However, the number of facets of Θ_t grows due to the intersection, which becomes computationally expensive. This is usually mitigated by limiting the number of facets

[18], or using a moving window algorithm [14] as we do in the following. We overapproximate Θ_t by a hypercube Θ_t^{HC}, represented by the center $\bar{\theta}_t$ and a scalar cube size η_t:

$$\Theta_t^{HC} = \bar{\theta}_t \oplus \eta_t \mathbb{B}_2$$

where $\oplus$ is the Minkowski sum, and $\mathbb{B}_2 = \{\epsilon \in \mathbb{R}^2 : |\epsilon|_\infty \leq 0.5\}$ is the unit box in $\mathbb{R}^2$.

2) Least Mean Square parameter estimation: While SML learns parametric uncertainty bounds, MPC performance relies on having a good prediction model for the performance. Hence, LMS is often used as a point estimate method in conjunction with SML. LMS also ensures $\mathcal{L}_2$-stability of the closed loop under some assumptions [14], [13]. The estimated parameter $\hat{\theta}$ has to be projected on the set Θ_t since the latter robustly contains the true parameter θ^*.

IV. Adaptive Robust MPC for Bilateral Teleoperation

Fig.1 shows the general architecture of the teleoperation system with the controller and the learning mechanisms. We measure the forces, positions, and velocities of both robots and send them to the centralized controller and learning components. The SML uses these measurements to learn the uncertainty bounds Θ_t and its overapproximation Θ_t^{HC}, which is then used by the AR-MPC safety constraints. The smaller Θ_t^{HC} gets, the less conservative these constraints get. The LMS-learning finds an estimate of θ^* and projects it on Θ_t, which is then used in the cost function for better performance when the system is far from the constraints. Finally, the optimizer calculates control inputs u_o, u_r for both robots to synchronize them and robustly satisfy the constraints. We consider negligible time delay, which is usually the case in surgical teleoperation where both robots are in the same room.

The future trajectories of the system are affected by the additive and parametric uncertainty, along with the change in f_h. While a slowly-varying f_h could be accounted for in the prediction as an additive uncertainty, we will consider a constant f_h in the following and leave the extension for future work.

A. The MPC optimization problem

We adapt the optimization problem in [14] for the teleoperation application. Assuming f_h is constant, at each time step t, we solve the following QP:

$$\min_{\zeta_\cdot, w_\cdot} \sum_{k=0}^{N-1} \mathcal{J}_{Q,R}(\hat{x}_k, \hat{u}_k) + \mathcal{J}_P^f(\hat{x}_N) \tag{8a}$$

$$\text{s.t. } \bar{x}_0 = \hat{x}_0 = x_t,\ s_0 = 0, \tag{8b}$$

$$\hat{x}_{k+1} = A_{cl,\hat{\theta}_t}\hat{x}_k + B\zeta_k + B_d f_h, \tag{8c}$$

$$\bar{x}_{k+1} = A_{cl,\bar{\theta}_t}\bar{x}_k + B\zeta_k + B_d f_h, \tag{8d}$$

$$s_{k+1} = \rho_{\bar{\theta}_t} s_k + w_k, \tag{8e}$$

$$F_j\bar{x}_k + G_j\bar{u}_k + c_j s_k \leq 1,\ j = 1, \ldots, q, \tag{8f}$$

$$\bar{u}_k = \zeta_k + K\bar{x}_k,\ \hat{u}_k = \zeta_k + K\hat{x}_k, \tag{8g}$$

$$(\bar{x}_N, s_N) \in \mathcal{X}_f, \tag{8h}$$

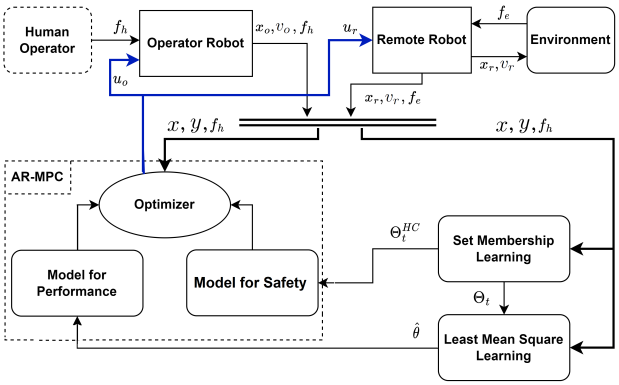

Fig. 1: Teleoperation system with the AR-MPC controller

where the subscript $(\cdot)_k$ means the prediction of $(\cdot)$ at the moment $(t+k)$; N is the prediction horizon; $\mathcal{J}_{Q,R}$ is a stage cost with weight matrices Q, R; $\mathcal{J}_P^f$ is a terminal cost function with weight matrix P; $(\bar{\cdot})$ are the variables related to the SML method; $(\hat{\cdot})$ are the variables related to the LMS method; matrices $A_{cl,\theta}, B, B_d$ are the model matrices defined in (4),(5) and (7); ζ_k is the additional optimal control input; s_k is the scale factor of the tube $\mathcal{P}$ and it is computed by the solver; $\rho_{\bar{\theta}_t}$ is the contraction rate of the tube at $\bar{\theta}_t$, updated at each time step t; w_k is an over-approximation of the uncertainty effect computed online by the solver (see [19] eq.(14e) for details); c_j are constraint tightening constants computed offline; $\mathcal{X}_f$ is the terminal set calculated offline; q is the number of constraints from (6). The previous optimization problem (8) has 3 types of propagated trajectories:

- (8c) describes $\hat{x}_k$, the state trajectory propagated based on the point-estimate $\hat{\theta}$, and it is used in the cost function for performance;
- (8d) describes $\bar{x}_k$, the state trajectory based on SML-learned $\bar{\theta}$, which is used for robustness guarantees;
- (8e) describes s_k, the tube scaling parameter, propagated based on the contraction rate of the tube $\rho_{\bar{\theta}}$ and w_k, which over-approximates the effect of parametric and additive uncertainty. At each time step t, the solver tries to build a tube by finding suitable trajectories of $\bar{x}_k, s_k$: $\mathcal{P} = \{\mathcal{P}_k = \bar{x}_k \oplus s_k\mathcal{P}_0\}_{k\in\{1\ldots N\}}$.

The inequality (8f) describes the tightened version of the constraints (6) to ensure that the whole predicted tube $\mathcal{P}$ satisfies the constraints. Finally, equality (8g) describes the control inputs, which consist of the precalculated state-feedback part (Kx_k) and the additional optimal input ζ_k.

B. The cost function and the terminal set

In [14], the cost function drives the system to the origin of the state space, and it could be modified to drive it to a different point. However, in teleoperation, the desired reference point is not constant, and the goal is to achieve system transparency by matching the positions and forces of the operator and the remote sides. To adapt the original algorithm to our application, we propose two solutions:

1) to calculate a virtual target point $(\tilde{x}, \tilde{u})$, which represents the target that the human is trying to reach, calculated using the applied human force f_h and the estimated environment stiffness $\hat{k}_e$. The cost function (8a) becomes:

$$\sum_{k=0}^{N-1} \left(\|\hat{x}_k - \tilde{x}\|_Q^2 + \|\hat{u}_k - \tilde{u}\|_R^2 \right) + \|\hat{x}_N - \tilde{x}\|_P^2$$

where $\tilde{x} = \left[\frac{f_h}{\hat{k}_e}, 0, \frac{f_h}{\hat{k}_e}, 0\right]^\top$, $\tilde{u} = \left[-f_h, \frac{f_h(\hat{k}_e + k_r)}{\hat{k}_e}\right]^\top$.

2) to minimize the position and force matching errors directly between the operator and remote sides. The cost function (8a) then becomes:

$$\sum_{k=0}^{N-1} \left(\|\hat{z}_k\|_{Q_z}^2 + \|\hat{u}_k\|_R^2 \right) + \|\hat{z}_N\|_{P_z}^2$$

where $\hat{z} = \left[\hat{x}_o - \hat{x}_r,\ \hat{v}_o - \hat{v}_r,\ f_h - \hat{f}_e\right]^\top$.

Since $\tilde{x}$ could change due to the learning of the environment stiffness $\hat{k}_e$, the terminal set $\mathcal{X}_f$ is translated with its center $\tilde{x}$. We need to guarantee that our predicted solution $(\bar{x}_N, s_N)$ will stay inside the new translated terminal set. In order to achieve that, we replace the condition (8h) with the following two conditions (for both proposed solutions):

$$\begin{aligned} (\bar{x}_N, s_N) &\in \tilde{x}(k_{e,max}) \oplus \mathcal{X}_f \\ (\bar{x}_N, s_N) &\in \tilde{x}(k_{e,min}) \oplus \mathcal{X}_f \end{aligned} \tag{9}$$

We differentiate between both solutions in robustness as follows: The first solution guarantees robustly both constraint satisfaction and $\mathcal{L}_2$-stability (assuming f_h is constant), which was proved in [19]. The second solution guarantees constraint satisfaction, but the stability proof has to be reformulated based on the specific matrices Q_z, R, which we leave for future work. It is worth noting that the performance of both solutions is dependent on the tuning of the controller, and studying the advantages of each solution is an interesting topic for future research. In the simulation section, we will adopt the second solution for its simplicity and intuitiveness.

Algorithm 1 describes the general steps of the controller design and execution process.

Algorithm 1 Controller design and execution

Offline:

1: **Given:** model; initial uncertainty Θ_0; constraints F, G; additive uncertainty bounds $\mathbb{D}$; weights Q, R.
2: find $K, P, \mathcal{P}_0$ ▷ eq. (33) in [19], and [20]
3: initialize the optimizer: find $\rho_{\bar{\theta}_0}, c_j, \mathcal{X}_f$. ▷ Alg. 2 [19]
4: **if** cond. (16) in [19] is false **or** no solution **then**
5: choose different (Q, R), or find tighter $(\Theta_0, \mathbb{D})$.
6: go back to line 1.
7: **end if**

Online:

8: **for** each time step $t > 1$ **do**
9: measure $f_h, x_t = [x_o, v_o, x_r, v_r]^\top$.
10: update Θ_t using the SML method.
11: overapproximate Θ_t to a hypercube Θ_t^{HC}.
12: update LMS estimate $\hat{\theta}_t$ and project it on Θ_t^{HC}.
13: update $A_{cl,\hat{\theta}_t}, A_{cl,\bar{\theta}_t}$. ▷ eq. (5)
14: update $\rho_{\bar{\theta}_t}$. ▷ eq. (6) in [19]
15: solve optimization problem (8),(9).
16: apply control input $u_t = \zeta_0^* + Kx_t$. ▷ eq. (8g)
17: **end for**

V. Simulation Results

The proposed method was validated in simulation in MATLAB 2022a, using *mpt3 Toolbox* for polyhedrons [21], *yalmip* for formulating the optimization problem [22], and *mosek* as a solver [23]. We assume to have prior knowledge about the possible range of the environment model parameters, so the controller is initialized with the following values:

$$k_e = 1000\ N/m,\ b_e = 10\ Ns/m$$

$$\Delta k_e \in [-200, +200] N/m,\ \Delta b_e \in [-3, +3] Ns/m$$

The values of operator and remote robots models are the same as in [1] (converted from rotational to translational values), as reported in Table I; the sampling period is $T_s = 0.005s$; the prediction horizon is $N = 20$, and the weight matrices Q_z, R chosen experimentally as:

$$Q_z = \begin{pmatrix} 5 \times 10^4 & 1 & 1 \\ 1 & 10^{11} & 1 \\ 1 & 1 & 10^5 \end{pmatrix} \quad R = \begin{pmatrix} 2 & 1 \\ 1 & 10 \end{pmatrix} \times 10^{-5}$$

TABLE I: Numerical parameters of robots

$m_o = 0.24$	$m_r = 0.23$	kg
$b_o = 1.34$	$b_r = 0.8$	Ns/m
$k_o = 23.3$	$k_r = 6.13$	N/m

A. Offline calculations

First, we designed the prestabilizing feedback controller K, and the terminal cost matrix P using the method in [19], and we designed the tube section as a $\rho-$contractive set using the method proposed in [20] with $\rho = 0.6$, and the result is a 4-D polytope represented by 14 inequalities. The terminal set $\mathcal{X}_f$ is found based on [19]. The additive noise bounds are $|d_1| < 0.0001 \times T_s; |d_2| < 0.4 \times T_s; |d_3| < 0.0001 \times T_s; |d_4| < 0.01 \times T_s$, where d_i is the i_{th} component of d. It is worth pointing out that having tight bounds on the additive noise accelerates the SML. However, the learning might fail if the noise exceeds the pre-specified bounds, which is why we initialized the SML with 4 times the expected bounds as a safety margin. We also note that scaling θ and x has a major effect on how easily K, P, and the tube could be found. In the following simulations, we use the second proposed cost function of subsection IV-B.

B. Simulation 1: The effectiveness of SML and robust constraint satisfaction

The goal of this simulation is to demonstrate the controller ability to robustly satisfy the constraints, and to show the role of SML in reducing the controller conservatism. We set a

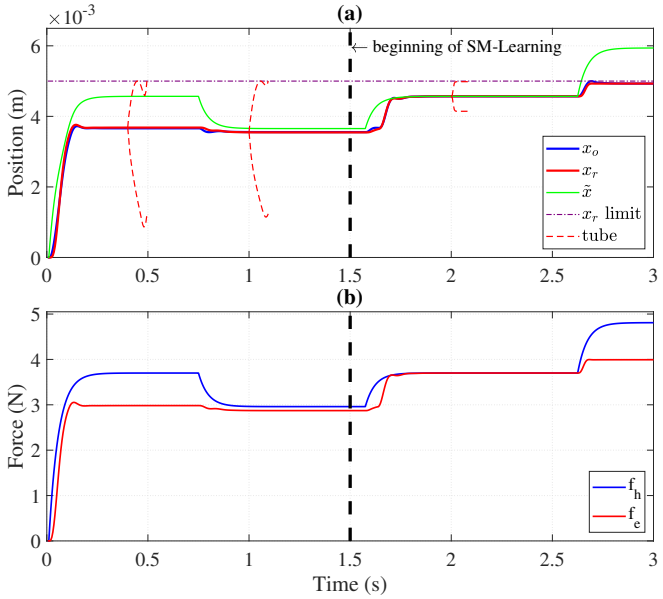

Fig. 2: Simulation 1, soft contact with a position constraint on the remote robot. The SML starts at $t = 1.5$. (a) The position of the operator (blue) and environment (red); the virtual target position based on f_h and k_e (green); the constraint on x_r (violet); the predicted tube around x_r at $t \in \{0.4, 1, 2\}s$ (dashed-red). (b) The human force (blue); the environment force (red).

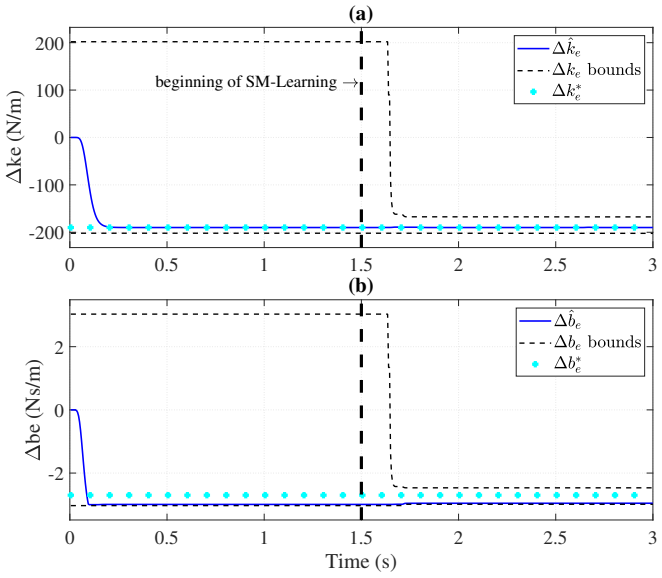

Fig. 3: Simulation 1, LMS and SML learning of the environment parametric uncertainty. The figure shows the projection of Θ_t^{HC} on each of the parameters (a) environment stiffness k_e; (b) environment damping coefficient b_e. On each figure, the estimated parameter by LMS is in blue; the boundaries of the uncertainty are in black-dashed lines, learned by SML; the true value of the parameter is in cyan. The SML starts at $t = 1.5s$

position constraint on the remote robot as: $x_r \leq 5 \times 10^{-3} m$. This type of constraint is usually used in medical robots as a virtual fixture for the safety of the patient. The SML starts only at $t = 1.5s$, while the LMS-learning starts from $t = 0s$.

First stage, before SML: At $t = 0s$, the remote robot is in contact with the environment (i.e. $x > 0$). The human operator applies a filtered step force, as shown in Fig. 2b (blue line). This force moves the operator robot, and the remote robot follows closely (red line) until the remote robot gets close to the constraint limit, that is when the controller prevents it from continuing the motion, and the operator robot starts resisting the human hand to keep its position matched with the remote robot. The green line represents the virtual target position that the human is trying to reach (based on f_h and $\hat{k}_e$). In Fig. 2a, we show the boundaries of the tube projected on x_r (red-dashed lines) at $t \in \{0.4s, 1s, 2s\}$. The tube is predicted over a horizon of 100 ms. All possible state trajectories under the uncertainty are bounded between the two red dashed lines of the tube. During this stage, the position matching error is small, but the reflected force to the human operator does not match the environment force as shown in Fig. 2b because the remote robot is not allowed to go deeper in the environment.

Second stage: SML At $t = 1.5s$, the SML starts, and quickly, the bounds on the parametric uncertainty begin to shrink as seen in Fig. 3 (black dashed lines). At $t = 1.6s$, the operator increases the force to the same previous level, but this time, both robots move very closely to the constraints, since the uncertainty became much smaller. The tube shown at $t = 2s$ is much smaller than the one of $t = 0.4s$, which means that the controller is now able to find a smaller tube that lies inside the constrained set. At $t = 2.6s$, the human tries to violate the constraint with a bigger force, but the remote robot stays inside the safe region, and the operator robot resists the motion and keeps the positions matched. Fig. 3 shows how the LMS-learning converges fast to the true values of the parameters.

C. Simulation 2: The effect of LMS-learning

In case the bounds on the additive noise are not tight enough, SML might be slow. For this case, the parameters learned using the LMS method are used in the cost function to enhance the performance. In this simulation, we remove the position constraints and the SML. The controller is initialized with $k_e = 1000 \pm 200 N/m$, $b_e = 10 \pm 3 Ns/m$. The true parameters are $k_e^* = 1190 N/m, b_e^* = 12.7 Ns/m$. Fig.

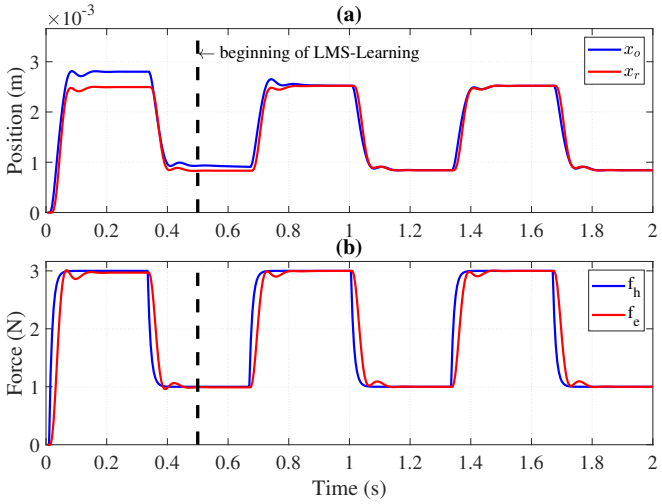

Fig. 4: Simulation 2, soft contact with LMS-learning and no constraints. (a) Position of the operator robot (blue) and the remote robot (red). (b) Human force (blue); environment force (red).

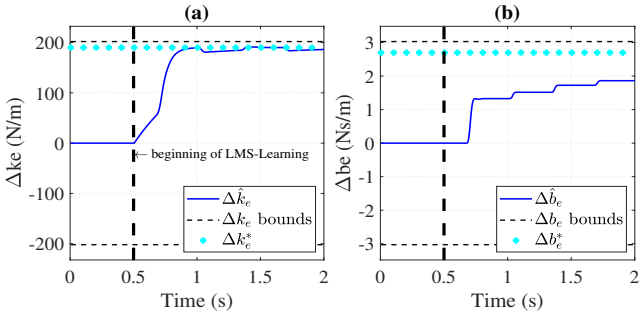

Fig. 5: Simulation 2, LMS parameter learning without SML. (a) Learning of environment stiffness uncertainty. (b) Learning of environment damping coefficient uncertainty.

4 shows that before learning, the position matching errors between both robots are large. At $t = 0.5s$, the LMS-learning starts and the parameters start converging to their true values as shown in Fig. 5. By $t = 0.8s$, the matching errors become much smaller since the controller is using a good estimate of the parameters in the cost function. This simulation shows how LMS-learning can enhance performance even if the bounds on parametric uncertainty are still wide, as long as the system is far from the constraints. In practice, we found that the damping coefficient b_e is more difficult to learn than the stiffness k_e, and it is learned only during the dynamic motions. This can be observed in Fig. 5.

This version of the controller guarantees robust constraint satisfaction and recursive feasibility after f_h becomes constant and the optimizer finds a first solution. This could be mitigated by considering the change in f_h as an additive uncertainty and taking it into account in $\mathbb{D}$, at the expense of more conservatism in the control actions.

VI. CONCLUSION AND FUTURE WORK

In this work, we have presented an adaptive robust MPC framework for bilateral teleoperation systems, which combines tube controllers with two online learning mechanisms to learn the environment model. The set membership method learns the bounds of parametric uncertainty to reduce the conservatism of the robust tube controller, and the least mean square method improves the prediction model. The framework ensures constraint satisfaction robustly and recursively, which means that if the optimization problem starts from a feasible point, it will be feasible in the future even under the worst-case uncertainty. Our simulations have shown that the proposed approach is effective in maintaining constraints, enhancing performance, and reducing controller conservatism during teleoperation tasks. A limitation of this work is that human force is considered constant, and that the tuning and offline calculations are not trivial. Future work will focus on considering a time-varying human force, and on the transition between free motion and contact states.

ACKNOWLEDGMENT

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

REFERENCES

[1] C. A. Lopez Martinez, I. Polat, R. Van De Molengraft, and M. Steinbuch, "Robust High Performance Bilateral Teleoperation Under Bounded Time-Varying Dynamics," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 1, pp. 206–218, 2015.

[2] G. Niemeyer and J. J. E. Slotine, "Stable adaptive teleoperation," *IEEE Journal of Oceanic Engineering*, vol. 16, pp. 152–162, 1991.

[3] J. H. Ryu, D. S. Kwon, and B. Hannaford, "Stable teleoperation with time-domain passivity control," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 365–373, 2004.

[4] F. Ferraguti, N. Preda, A. Manurung, M. Bonfe, O. Lambercy, R. Gassert, R. Muradore, P. Fiorini, and C. Secchi, "An energy tank-based interactive control architecture for autonomous and teleoperated robotic surgery," *IEEE Transactions on Robotics*, vol. 31, pp. 1073–1088, 2015.

[5] N. Piccinelli and R. Muradore, "A passivity-based bilateral teleoperation architecture using distributed nonlinear model predictive control," *IEEE International Conference on Intelligent Robots and Systems*, pp. 11466–11472, 2020.

[6] A. Bemporad, "Predictive control of teleoperated constrained systems with unbounded communication delays," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 2, pp. 2133–2138, 1998.

[7] T. Slama, D. Aubry, R. Oboe, and F. Kratz, "Robust bilateral generalized predictive control for teleoperation systems," in *Mediterranean Conference on Control & Automation*, pp. 1–6, 2007.

[8] N. Piccinelli and R. Muradore, "A bilateral teleoperation with interaction force constraint in unknown environment using non linear model predictive control," *European Journal of Control*, vol. 62, pp. 185–191, 2021.

[9] N. Piccinelli and R. Muradore, "Passivity-Based Teleoperation With Interaction Force Constraints Using Hybrid Linear Model Predictive Control," in *Mediterranean Conference on Control and Automation*, pp. 25–30, 2022.

[10] D. M. Raimondo, D. Limon, M. Lazar, L. Magni, and E. F. ndez Camacho, "Min-max Model Predictive Control of Nonlinear Systems: A Unifying Overview on Stability," *European Journal of Control*, vol. 15, no. 1, pp. 5–21, 2009.

[11] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[12] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[13] M. Lorenzen, M. Cannon, and F. Allgöwer, "Robust MPC with recursive model update," *Automatica*, vol. 103, pp. 461–471, 2019.

[14] J. Köhler, E. Andina, R. Soloperto, M. A. Müller, and F. Allgöwer, "Linear robust adaptive model predictive control: Computational complexity and conservatism," in *IEEE Conference on Decision and Control*, pp. 1383–1388, 2019.

[15] L. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *IEEE Virtual Reality Annual International Symposium*, pp. 76–82, 1993.

[16] *Medical electrical equipment — Part 2-77: Particular requirements for the basic safety and essential performance of robotically assisted surgical equipment, IEC 80601-2-77*, 2019.

[17] F. Benzi, F. Ferraguti, and C. Secchi, "Energy Tank-based Control Framework for Satisfying the ISO/TS 15066 Constraint," Apr. 2023. arXiv:2304.14059 [cs].

[18] M. Tanaskovic, L. Fagiano, R. Smith, and M. Morari, "Adaptive receding horizon control for constrained MIMO systems," *Automatica*, vol. 50, no. 12, pp. 3019–3029, 2014.

[19] J. Köhler, E. Andina, R. Soloperto, M. A. Müller, and F. Allgöwer, "Linear robust adaptive model predictive control: Computational complexity and conservatism – extended version," 2020. arXiv:1909.01813.

[20] B. Pluymers, J. Rossiter, J. Suykens, and B. De Moor, "The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty," in *American Control Conference*, pp. 804–809 vol. 2, 2005.

[21] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *European Control Conference*, pp. 502–510, 2013.

[22] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *CACSD Conference*, 2004.

[23] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.