2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Learning-Based Model Predictive Control with Application in Robotic Trajectory Tracking

Hongyu Zhu, Mengna Liu and Dan Yu

Abstract— **This paper studies the learning-based model predictive control problem for nonlinear systems with model uncertainties and control constraints. First, a prediction model is constructed offline. The prediction model is composed of a nominal model derived using the first principle with known parameters, and a learning model constructed via the LSTM network to account for model uncertainties and unknown disturbances. Then control input increments are optimized using an online model predictive controller with constraints. Simulation results for trajectory tracking with a robotic arm are presented to verify the robustness and feasibility of the proposed approach.**

I. INTRODUCTION

Robotic arms have been widely used to assist people with high-intensity, repetitive tasks [1], [2]. In these applications, high-precision autonomous trajectory tracking is one of the fundamental tasks. However, designing the trajectory controller for robotic arms is difficult due to the existence of modeling errors, unknown parameters, and external disturbances. Moreover, in practice, there is a high demand for the real-time response speed of the robotic system, and the system is also subject to different constraints, which lead to challenges in controller design.

A combination of learning-based algorithms and model predictive control (MPC) has shown great potential in addressing these challenges, and has received a lot of attention recently [3]. MPC can handle state and control constraints and is simple to implement, while learning-based algorithms can refine models of the system and improve system performance [4].

MPC is an online control method that solves an open-loop optimal control problem over a finite horizon and then applies the first optimal control action to the plant [5]. By repeating the process recursively, MPC achieves online control with hard state and control constraints, and hence, has been widely used in industry process control [6], formation flight [7], robotic motion planning [8], and so on.

For deterministic systems, MPC is equivalent to solving the original Dynamic Programming (DP) problem online recursively, and hence, the stability and optimality can be guaranteed [9]. However, it is difficult to use an open-loop control strategy to optimize the performance of uncertain systems. To deal with uncertainties, robust MPC and stochastic MPC have been proposed. For example, a robust model predictive controller is designed for a robot manipulator to track the time-varying trajectory in [10]. The robot manipulator is assumed to be affected by bounded disturbances, and a feedback control law is designed with the guarantee of robust constraint satisfaction. In [11], a stochastic MPC method for trajectory tracking of wheeled mobile robots is proposed. The probability distribution of the stochastic disturbance is assumed to be known, and the tracking error of the closed-loop system is asymptotically average bounded. Robust MPC is over conservative when handling uncertainties, while stochastic MPC may violate the constraints with a small probability.

Another challenge of MPC is the lack of a reliable prediction model. Traditional MPC relies on the dynamic model of the system, which is difficult to know precisely. Prediction using an inaccurate model would result in performance degradation or even unsafety. Therefore, to better fit the real system, data-driven approaches can be employed where input-output data could be used to learn the system model or control policy, resulting in the development of the data-driven or learning-based MPC (LB-MPC) [12], [13] methods.

One major class of LB-MPC focuses on data-based adaption of the prediction model. For example, in [14], the Gaussian Process (GP) is used to learn the periodic error, and prediction from the GP model is used to design the online MPC controller. In [4], a GP is trained offline to estimate the discrepancies between the actual and analytical model, and the extended Kalman filter is used to estimate the residual model mismatch online. The performance of the proposed algorithm is validated for trajectory tracking using a 6 degree of freedom robotic arm. Problems with the online updates of GP model is that GP keeps tracking all measurement data, which would be computationally infeasible over time. Moreover, the mean and covariance function of a GP need to be chosen a priori, which will also affect the performance of the model. Koopman operator theory provides a data-driven approach to construct a linear model of a nonlinear controlled dynamical system based on the input and output data. For example, a linear data-driven model is constructed via the Koopman operator for the soft robots, and different MPC controllers are developed in [15]. In [16], the Koopman operator-based MPC scheme is proposed for trajectory tracking control of an omnidirectional mobile manipulators. However, proper basis functions need to be chosen carefully,

This work was supported by Nanjing University of Aeronautics and Astronautics, Grant no. xcxjh20221509, National Natural Science Foundation of China, Grant No.61801213

Hongyu Zhu and Mengna Liu are postgraduates in College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China

Dan Yu is with the College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China yudan@nuaa.edu.cn

and training for an accurate model depends on trials and errors, which limits applications of the Koopman operator based-MPC.

Neural Networks have been extensively studied in the Control Systems community as black-box models of the plants. The performance and applicability of NNs to general nonlinear systems have been demonstrated in many applications. Among current state-of-the-art NN architectures, Long Short Term Memory (LSTM) network has gained increasing attentions in control-related applications [17]. LSTM network has a unique memory cell, which makes it better suited for learning dynamical systems. For example, a LSTM-based MPC controller is designed to optimize the energy consumption of a building in [18]. The LSTM model is trained offline, and simulation results indicate that the computational time using LSTM model is reduced when compared to the reference detailed model, and the control performance using LSTM model is improved. In [19], the LSTM network is used as a predictive model for multimode process, and theoretical analysis of the stability and feasibility of the proposed LSTM-MPC method has been presented.

In this paper, we propose a learning-based model predictive control approach for nonlinear system with uncertainties and control constraints. The prediction model is composed of a nominal model and a learning model. The nominal model is derived from the first principle with known parameters. The learning model is constructed via training the LSTM network offline, and is used to compensate the modeling errors and external disturbances. The MPC controller is designed online to optimize the input increments under constraints. The primary contributions of the proposed approach are:

1) A novel LSTM-based MPC method is proposed for the robotic trajectory tracking problem. Compared with existing LSTM-based MPC approaches, the proposed LSTM model is used to learn the discrepancies between the actual system and the analytical model, such that the performance of the proposed controller can be improved.

2) Implementation using the proposed approach is simple, where a standard MPC scheme is used. The training of the LSTM network is done offline, which reduces the online computational time.

3) The online optimal control problem is reformulated to optimize the control input increments, constraints on both control input and control input increments need to be satisfied, which makes the proposed approach more feasible in practice.

The rest of the paper is organized as follows. In Section II, a short introduction to LSTM network is provided. In Section III, the LB-MPC algorithm is proposed. Computational results for trajectory tracking with a double link rigid-flexible robotic arm are shown in Section IV. Finally, Section V concludes the paper.

II. LONG SHORT-TERM MEMORY NETWORK

In this section, we briefly review the Long Short-Term Memory (LSTM) Network. The LSTM network is a special Recurrent Neural Network that has been widely used in temporal modeling problems.

Assume that at each time $t_k = k\Delta t$, we collect a sequence of data $\tilde{X}_k = [\tilde{\boldsymbol{x}}_{\boldsymbol{k-p+1}}, \tilde{\boldsymbol{x}}_{\boldsymbol{k-p+2}}, \cdots, \tilde{\boldsymbol{x}}_{\boldsymbol{k}}]$, where Δt is the sampling time, $\tilde{\boldsymbol{x}}_{\boldsymbol{k}} \in \mathcal{R}^N$ represents the measurement data at time t_k, and p is the length of the moving horizon. The LSTM network is used to predict $\tilde{\boldsymbol{x}}_{\boldsymbol{k+1}}$ based on the past p measurements $\tilde{X}_k$, i.e., the output of the LSTM network $\boldsymbol{Y_k} = \tilde{\boldsymbol{x}}_{\boldsymbol{k+1}}$.

The LSTM network is composed of one input layer, several hidden layers, and one fully connected output layer. The structure of the LSTM network with two hidden layers is shown in Fig. 1.

Each hidden layer contains several LSTM cells. Denote the cell state and cell output at time step k as $\boldsymbol{c_k} \in \mathcal{R}^N$ and $\boldsymbol{H_k} \in \mathcal{R}^N$, respectively. At each time step k, the cell state $\boldsymbol{c_{k-1}}$, cell output $\boldsymbol{H_{k-1}}$ and measurement data $\tilde{\boldsymbol{x}}_{\boldsymbol{k}}$ are passed through the forget gate, and the outputs of the forget gate is

$$\boldsymbol{f_k} = \sigma(W_{xf}\tilde{\boldsymbol{x}}_{\boldsymbol{k}} + W_{hf}\boldsymbol{H_{k-1}} + \boldsymbol{b_f}), \tag{1}$$

where $W_{xf} \in \mathcal{R}^{N\times N}, W_{hf} \in \mathcal{R}^{N\times N}$ are some weight matrices, and $\boldsymbol{b_f} \in \mathcal{R}^N$ is the bias. $\sigma(.)$ represents the element-wise sigmoid activation function.

The input gate filters the training sample $\tilde{\boldsymbol{x}}_{\boldsymbol{k}}$ and output $\boldsymbol{H_{k-1}}$ to be passed to the memory cell, and is implemented as

$$\begin{aligned} \boldsymbol{i_k} &= \sigma(W_{xi}\tilde{\boldsymbol{x}}_{\boldsymbol{k}} + W_{hi}\boldsymbol{H_{k-1}} + \boldsymbol{b_i}), \\ \boldsymbol{q_k} &= tanh(W_{xc}\tilde{\boldsymbol{x}}_{\boldsymbol{k}} + W_{hc}\boldsymbol{H_{k-1}} + \boldsymbol{b_c}), \\ \boldsymbol{g_k} &= \boldsymbol{i_k} * \boldsymbol{q_k}, \end{aligned} \tag{2}$$

where $\boldsymbol{i_k}$ is the output of the input gate, $\boldsymbol{g_k}$ is the updated cell state candidate, $W_{xi}, W_{hi}, W_{xc}, W_{hc}$ are the corresponding weight matrices, $\boldsymbol{b_i}, \boldsymbol{b_c}$ are the corresponding bias. $tanh(.)$ is the hyperbolic tangent activation function, and $*$ represents the element-wise product.

The current cell state $\boldsymbol{c_k}$ is

$$\boldsymbol{c_k} = \boldsymbol{f_k} * \boldsymbol{c_{k-1}} + \boldsymbol{g_k}, \tag{3}$$

and the output of the output gate $\boldsymbol{o_k}$ is

$$\boldsymbol{o_k} = \sigma(W_{xo}\tilde{\boldsymbol{x}}_{\boldsymbol{k}} + W_{ho}\boldsymbol{H_{k-1}} + \boldsymbol{b_o}), \tag{4}$$

where $W_{xo}, W_{ho}, \boldsymbol{b_o}$ are the corresponding weight matrices and bias, respectively.

Therefore, the output of the LSTM cell at time step k is

$$\boldsymbol{H_k} = \boldsymbol{o_k} * tanh(\boldsymbol{c_k}), \tag{5}$$

and the output of the fully connected layer is

$$\boldsymbol{Y_k} = W_{hy}\boldsymbol{H_k} + \boldsymbol{b_y}, \tag{6}$$

where W_{hy} and $\boldsymbol{b_y}$ are the weight matrix and bias, respectively.

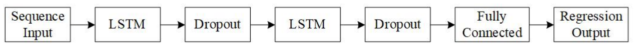


Fig. 1. Structure diagram of LSTM network

The mean square error (MSE) is used as the loss function, i.e.,

$$Loss = \frac{1}{S}\sum_{k=1}^{S}(\boldsymbol{y_k} - \boldsymbol{Y_k})^2, \tag{7}$$

where S is the number of samples, $\boldsymbol{y_k}$ is the actual output corresponding to the k^{th} sample, and $\boldsymbol{Y_k}$ is the predicted output. During the training process, the gradient descent method is used to reduce the loss function.

III. LEARNING-BASED MODEL PREDICTIVE CONTROLLER DESIGN

Model Predictive Control (MPC) solves an online optimization problem at each time step, implements the first control action, and then repeats the process at the next time step. A prediction model is needed to predict the system's behavior over a given horizon. However, in practice, the exact model is unknown due to the existence of parameter uncertainties and unknown disturbances. Therefore, in this section, we first introduce the learning-based prediction model, and then introduce the implementation details of the proposed LB-MPC controller.

A. Prediction Model

It is well-known that the dynamics of a robotic arm with multiple joints can be modeled as a second-order differential equation

$$M(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + H(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + K\boldsymbol{\theta} = \boldsymbol{u} + \boldsymbol{d}, \tag{8}$$

where $\boldsymbol{\theta} \in \mathcal{R}^n$ represents the joint angles, and $\dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}$ represent the joint angular velocities, and joint angular accelerations, respectively. $M(\boldsymbol{\theta}) \in \mathcal{R}^{n\times n}$ is the system inertial matrix, $H(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \in \mathcal{R}^{n\times n}$ is the coriolis and centrifugal matrix of the system, and $K \in \mathcal{R}^{n\times n}$ is the system elastic stiffness matrix. $\boldsymbol{u} \in \mathcal{R}^n$ is the control input vector, and $\boldsymbol{d} \in \mathcal{R}^n$ denotes the unknown disturbance acting on the joints.

In practice, the exact model parameters are not known, so the actual system matrices could be expressed as

$$\begin{aligned} M(\boldsymbol{\theta}) &= \bar{M}(\boldsymbol{\theta}) + \Delta M(\boldsymbol{\theta}), \\ H(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) &= \bar{H}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \Delta H(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}), \\ K &= \bar{K} + \Delta K, \end{aligned} \tag{9}$$

where $\bar{M}, \bar{H}, \bar{K}$ are matrices constructed using known parameters, $\Delta M, \Delta H, \Delta K$ represent the modeling errors induced by parameter uncertainties. Therefore, substituting (9) into (8) yields

$$\bar{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \bar{H}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \bar{K}\boldsymbol{\theta} = \boldsymbol{u} + \tilde{\boldsymbol{d}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}), \tag{10}$$

where $\tilde{\boldsymbol{d}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \boldsymbol{d} - \Delta M(\boldsymbol{\theta}) - \Delta H(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) - \Delta K$ takes into account the model uncertainties and unknown disturbances. Denote $\boldsymbol{x} = [\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}]'$, then from (10), we have

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{\boldsymbol{\theta}} \\ \bar{\boldsymbol{M}}^{-1}(\boldsymbol{\theta})(\boldsymbol{u} - \bar{H}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) - \bar{K}\boldsymbol{\theta}) \end{pmatrix} + \begin{pmatrix} 0 \\ \bar{\boldsymbol{M}}^{-1}(\boldsymbol{\theta})\tilde{\boldsymbol{d}}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \end{pmatrix}. \tag{11}$$

The discrete-time state space model is

$$\boldsymbol{x_{k+1}} = \bar{f}(\boldsymbol{x_k}, \boldsymbol{u_k}) + f_d(\boldsymbol{x_k}, \boldsymbol{u_k}), \tag{12}$$

where $\boldsymbol{x_k} = [\boldsymbol{\theta}(t_k), \dot{\boldsymbol{\theta}}(t_k)]'$ and $\boldsymbol{u_k} = \boldsymbol{u}(t_k)$ are the state vector and the control input vector at time $t_k = k\Delta t$, respectively. $\bar{f}(.)$ is a nonlinear function constructed using known parameters, and is denoted as the **nominal model**. $f_d(.)$ is an unknown nonlinear function which accounts for model uncertainties and is denoted as the **learning model**.

In this paper, the prediction model (12) is composed of a nominal model and a learning model. The nominal model $\bar{f}(.)$ is derived using the first principle with known parameters, and the learning model $f_d(.)$ is constructed via training the LSTM network offline.

From (12), let

$$\Delta\boldsymbol{x_k} \triangleq \boldsymbol{x_k} - \bar{f}(\boldsymbol{x_{k-1}}, \boldsymbol{u_{k-1}}), \tag{13}$$

where $\Delta\boldsymbol{x_k}$ represents the differences between actual measurements $\boldsymbol{x_k}$ and predictions from the nominal model at time step k. Let $\tilde{\boldsymbol{x}}_{\boldsymbol{k}} = [\Delta\boldsymbol{x}'_{\boldsymbol{k}}, \boldsymbol{u}'_{\boldsymbol{k}}]'$, and the objective is to train the LSTM network, such that

$$\boldsymbol{Y}_k = \tilde{\boldsymbol{x}}_{\boldsymbol{k+1}} = \tilde{f}_d(\tilde{X}_k), \tag{14}$$

where $\tilde{X}_k = [\tilde{\boldsymbol{x}}_{\boldsymbol{k-p+1}}, \tilde{\boldsymbol{x}}_{\boldsymbol{k-p+2}}, \cdots, \tilde{\boldsymbol{x}}_{\boldsymbol{k}}]$ include the past p information, p is the design parameter.

B. Cost Function

The objective is to control a robotic arm to track a reference trajectory under constraints and uncertainties. In order to ensure smooth changes in control inputs, in this paper, we design the learning-based model predictive controller using the increments of control inputs, i.e., at each time step k, given the current state $\boldsymbol{x_k}$ and control input $\boldsymbol{u_{k-1}}$, MPC solves the optimal control problem

$$\begin{aligned} &\min_{\{\Delta\boldsymbol{u_{k+l}}\}_{l=0}^{H-1}} J(\boldsymbol{x_k}, \{\Delta\boldsymbol{u_k}\}_{l=0}^{H-1}) \\ &= \min_{\{\Delta\boldsymbol{u_{k+l}}\}_{l=0}^{H-1}} c_H(\boldsymbol{x_{k+H|k}}) + \sum_{l=0}^{H-1} c_l(\boldsymbol{x_{k+l|k}}, \Delta\boldsymbol{u_{k+l}}), \end{aligned} \tag{15}$$

where H is the length of the moving horizon, and $c_0, \cdots, c_H$ represents the transient and terminal cost, respectively. Let

$$\Delta\boldsymbol{u_{k+l}} \triangleq \boldsymbol{u_{k+l}} - \boldsymbol{u_{k+l-1}}, \forall l = 0, 1, \cdots, H-1, \tag{16}$$

represents the input increment at time step $k+l$.

The predicted state is computed recursively as

$$\begin{aligned} \boldsymbol{u_{k+l}} &= \boldsymbol{u_{k+l-1}} + \Delta\boldsymbol{u_{k+l}}, \\ \boldsymbol{x_{k+l+1|k}} &= \bar{f}(\boldsymbol{x_{k+l|k}}, \boldsymbol{u_{k+l}}) + \Delta\boldsymbol{x_{k+l+1}}, \end{aligned} \tag{17}$$

where $\boldsymbol{x_{k|k}} = \boldsymbol{x_k}$, and

$$\begin{pmatrix} \Delta\boldsymbol{x_{k+l+1}} \\ \boldsymbol{u_{k+l+1}} \end{pmatrix} = \tilde{f}_d(\begin{pmatrix} \Delta\boldsymbol{x_{k+l-p+1}} & \cdots & \Delta\boldsymbol{x_{k+l}} \\ \boldsymbol{u_{k-p+1}} & \cdots & \boldsymbol{u_{k+l}} \end{pmatrix}). \tag{18}$$

with $0 \leq l \leq H-1$.

Remark 1: Note that at time step k, the actual measurements $\boldsymbol{x}_{k+1}, \cdots, \boldsymbol{x_{k+H}}$ are unknown, and hence, are approximated via (17). After the optimal control action $\boldsymbol{u_k}$ is implemented, then $\Delta\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k+1} - \bar{f}(\boldsymbol{x_k}, \boldsymbol{u_k})$ is updated.

C. Control Constraints

Both the control inputs $\boldsymbol{u_k}$ and the increments of control inputs $\Delta\boldsymbol{u_k}$ are constrained. The input incremental constraints are given as

$$\Delta\boldsymbol{u_{min}} \le \Delta\boldsymbol{u_k} \le \Delta\boldsymbol{u_{max}}, k = 1, 2, \cdots, \tag{19}$$

where $\Delta\boldsymbol{u_{min}}$ and $\Delta\boldsymbol{u_{max}}$ are lower and upper bounds of control input increments, respectively.

The transformation between the input constraints and the input incremental constraints is derived as follows. Given $\boldsymbol{u_{k-1}}$, at time step k,

$$\begin{aligned} \boldsymbol{u_k} &= \boldsymbol{u_{k-1}} + \Delta\boldsymbol{u_k} \\ \boldsymbol{u_{k+1}} &= \boldsymbol{u_{k-1}} + \Delta\boldsymbol{u_k} + \Delta\boldsymbol{u_{k+1}} \\ &\vdots \\ \boldsymbol{u_{k+H-1}} &= \boldsymbol{u_{k-1}} + \Delta\boldsymbol{u_k} + \cdots + \Delta\boldsymbol{u_{k+H-1}}. \end{aligned} \tag{20}$$

Rewrite the above equation in a matrix form as

$$\underbrace{\begin{pmatrix} \boldsymbol{u_k} \\ \vdots \\ \boldsymbol{u_{k+H-1}} \end{pmatrix}}_{U_k} = G \underbrace{\begin{pmatrix} \Delta\boldsymbol{u_k} \\ \vdots \\ \Delta\boldsymbol{u_{k+H-1}} \end{pmatrix}}_{\Delta U_k} + \underbrace{\begin{pmatrix} \boldsymbol{u_{k-1}} \\ \vdots \\ \boldsymbol{u_{k-1}} \end{pmatrix}}_{\tilde{U}_{k-1}}, \tag{21}$$

where

$$G = \begin{pmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I & I & \cdots & I \end{pmatrix} \tag{22}$$

and hence,

$$U_{min} \le U_k = G\Delta U_k + \tilde{U}_{k-1} \le U_{max}, \tag{23}$$

where U_{min} and U_{max} are stacks of the lower and upper bounds of the control inputs, respectively. Therefore,

$$G^{-1}(U_{min} - \tilde{U}_{k-1}) \le \Delta U_k \le G^{-1}(U_{max} - \tilde{U}_{k-1}). \tag{24}$$

Let $\Delta\tilde{U}_{min} = [\Delta\boldsymbol{u'_{min}}, \Delta\boldsymbol{u'_{min}}, \cdots, \Delta\boldsymbol{u'_{min}}]'$, and $\Delta\tilde{U}_{max} = [\Delta\boldsymbol{u'_{max}}, \Delta\boldsymbol{u'_{max}}, \cdots, \Delta\boldsymbol{u'_{max}}]'$, then at each time step k, MPC solves the optimal control problem (15) subject to input constraints

$$\begin{aligned} max\{G^{-1}(U_{min} - \tilde{U}_{k-1}), \Delta\tilde{U}_{min}\} \le \Delta U_k \\ \le min\{G^{-1}(U_{max} - \tilde{U}_{k-1}), \Delta\tilde{U}_{max}\} \end{aligned} \tag{25}$$

D. LB-MPC Algorithm

The learning-based model predictive control algorithm presented in this paper is summarized in Algorithm 1.

IV. COMPUTATIONAL RESULTS

In this section, we test the proposed learning-based model predictive controller using a double link rigid-flexible manipulator. Define the root mean square error (RMSE) as

$$RMSE = \sqrt{\frac{1}{T}\sum_{k=1}^{T}(a_k - \bar{a}_k)^2}, \tag{26}$$

where a_k and $\bar{a}_k$ represents the actual and predicted states at time step k, respectively.

The rigid-flexible manipulator is shown in Fig. 2.

The dynamic model is given in [20]. We assume that the actual model parameters are unknown, and parameters we use to construct the nominal model are:

$$\begin{aligned} &L_1 = 0.3m, m_{L1} = 26.07kg, L_2 = 1.7m, m_{L2} = 10kg, \\ &J_{L1} = 20.86kg \cdot m^2, I_h = 30kg \cdot m^2, \rho A = 54.98kg/m, \\ &EI = 16.485N \cdot m^2, M_j = 5kg, I_j = 15kg \cdot m^2, \end{aligned}$$

where L_1, m_{L1}, J_{L1} are the length, mass and moment of inertia of the rigid link, respectively. $L_2, m_{L2}, I_h, \rho, A, EI$ are the length, mass, moment of inertia, density, cross-sectional area and flexural rigidity of the flexible link, respectively. M_j and I_j are the mass and moment of inertia of the second joint, respectively. θ_1, θ_2 are the relative joint angle of the rigid and flexible link, respectively.

The sampling time $\Delta t = 0.01s$ is used to discretize the system in time, and the control inputs are the torques applied to the joints. We consider the following input constraints $\|u_k\| \le 20$, and input incremental constraints $\|\Delta u_k\| \le 1.0$. The control objective is to find the optimal control sequence such that the end of the flexible link can track a given reference trajectory in 20 seconds. The reference trajectory is shown in Fig. 3.

First, we illustrate the training of the LSTM network. Assume that the model parameters of the actual systems are perturbed from the nominal with 20% uncertainties. The control input signal is chosen as $\boldsymbol{u_k} =$

Algorithm 1 LB-MPC Algorithm

```
1: Train the LSTM network (14) offline.
2: Set $k = 0$, given initial state $\boldsymbol{x_0}$, and set $\boldsymbol{u_{-1}} = 0$.
3: while $k \le T - 1$ do
4:     Set $\boldsymbol{x_{k|k}} = \boldsymbol{x_k}$, for $l = 0, 1, \cdots, H - 1$, predict
       states for $H$ steps via (17) and (18);
5:     Solves the optimal control problem (15) for optimal
       control increments $\Delta\boldsymbol{u^*_{k+l}}, l = 0, \cdots, H - 1$ subject to
       control constraints (25);
6:     Apply the first optimal control action $\boldsymbol{u^*_k} = \boldsymbol{u_{k-1}} +$
       $\Delta\boldsymbol{u^*_k}$ to the actual system (12) for $\boldsymbol{x_{k+1}}$;
7:     Update $\Delta\boldsymbol{x_{k+1}} = \boldsymbol{x_{k+1}} - \bar{f}(\boldsymbol{x_k}, \boldsymbol{u^*_k})$;
8:     $k = k + 1$.
9: end while
```

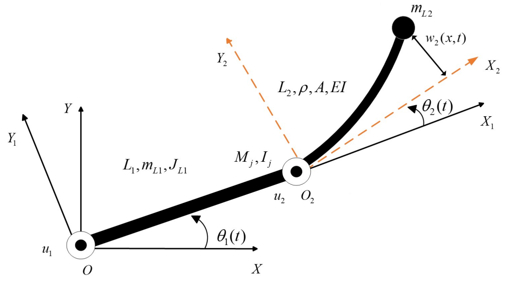


Fig. 2. Double link rigid-flexible manipulator

$[-6cos(0.1k\Delta t), -6cos(0.1k\Delta t)]$ with zero initial state. Data from the actual model for a total of 150 seconds are collected and normalized, where 80% of the data are used as the training data and 20% are used as the testing data. The length of the moving horizon $p = 5$, i.e., at each time step k, the past 5 measurements $[\tilde{\boldsymbol{x}}_{\boldsymbol{k-4}}, \tilde{\boldsymbol{x}}_{\boldsymbol{k-3}}, \cdots, \tilde{\boldsymbol{x}}_{\boldsymbol{k}}]$ are used as the inputs, and the output of the LSTM is $\tilde{\boldsymbol{x}}_{\boldsymbol{k+1}}$, where $\tilde{\boldsymbol{x}}_{\boldsymbol{k}} = [\Delta \boldsymbol{x}'_{\boldsymbol{k}}, \boldsymbol{u}'_{\boldsymbol{k}}]'$, and the schematic diagram is shown in Fig. 4.

The LSTM network used in this paper has two hidden layers. In order to prevent overfitting and improve the robustness of the model, some cells are randomly discarded from the network with a probability of 0.3. The number of neurons is 2^9 and the learning rate is 0.0005. The number of MaxEpochs is 6, and the size of mini batch is 64.

To test the accuracy of the LSTM model, we perturb the actual system with Gaussian white noise $\boldsymbol{w}_{\boldsymbol{k}} \sim N(0, 0.2)$ acting on the joints. The actual modeling errors are the differences between the actual data and the one computed using the nominal model. In Fig.5, we show comparisons of the actual modeling errors with modeling errors predicted using the LSTM model. It can be seen that the LSTM learning model could estimate the modeling uncertainties accurately.

In the following, we assume that parameters of the actual robotic arm system are perturbed by 20% and 40% from the nominal model, respectively, with Gaussian white noise $\boldsymbol{w}_{\boldsymbol{k}} \sim N(0, 0.2)$ acting on the joints. The prediction model used in LB-MPC consists of the nominal model and a learning model, where the learning model is trained from the corresponding actual system. In Fig. 6, we compare tracking errors of the flexible link between the proposed method and the decoupled feedback control method proposed in [20]. The same nominal model is used to design the open loop controller, and an LQR is designed for the close-loop control in the decoupled feedback control method. To test the performance of the proposed approach, we also compare with the standard MPC approach without consideration of modeling uncertainties, i.e., the controller in standard MPC is designed using the nominal model as the prediction model. Comparisons of input torques are shown in Fig. 7.

It can be seen that with model uncertainties and unknown disturbances, LB-MPC outperforms the other two methods, and tracking errors of the LB-MPC method is relatively small. Further, we compare the proposed LB-MPC controller with the decoupled feedback controller and standard MPC controller for systems with different model uncertainties. The average RMSE of the flexible link over the entire simulation time is shown in Fig.8. It can be seen that with increasing of model uncertainties, the proposed LB-MPC method still outperforms the other two methods.

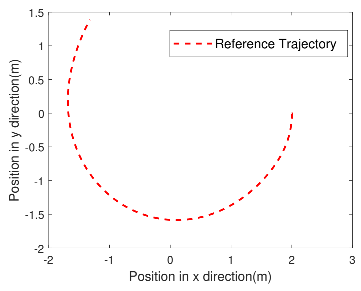


Fig. 3. Reference Trajectory

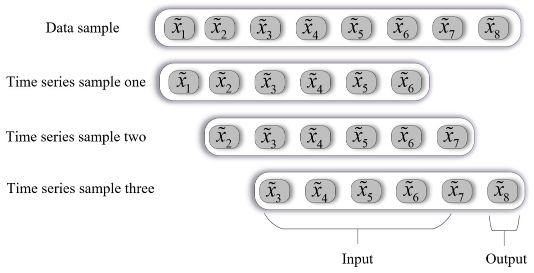

Data sample
$\tilde{x}_1$ $\tilde{x}_2$ $\tilde{x}_3$ $\tilde{x}_4$ $\tilde{x}_5$ $\tilde{x}_6$ $\tilde{x}_7$ $\tilde{x}_8$
Time series sample one
$\tilde{x}_1$ $\tilde{x}_2$ $\tilde{x}_3$ $\tilde{x}_4$ $\tilde{x}_5$ $\tilde{x}_6$
Time series sample two
$\tilde{x}_2$ $\tilde{x}_3$ $\tilde{x}_4$ $\tilde{x}_5$ $\tilde{x}_6$ $\tilde{x}_7$
Time series sample three
$\tilde{x}_3$ $\tilde{x}_4$ $\tilde{x}_5$ $\tilde{x}_6$ $\tilde{x}_7$ $\tilde{x}_8$
Input
Output

Fig. 4. Schematic diagram of time series construction

V. CONCLUSIONS

In this paper, we have proposed a learning-based model predictive control strategy with application to robotic motion planning problem. An online model predictive control framework is used, where the prediction model is composed of a nominal model and a learning model. The nominal model is derived using the first principle with known parameters, the learning model is constructed via training the LSTM network offline, and is used to taken into account unknown modeling errors and disturbances. MPC solves an online optimization problem for the optimal control input increments, with both constraints on the control inputs and control input increments. We have tested the proposed approach on a double link rigid-flexible manipulator, and showed the performance of the proposed approach. It can be seen that the proposed approach can track the reference trajectory accurately and is robust to the model uncertainties and random disturbances. Future work will generalize the proposed approach to partially observable systems.

REFERENCES

[1] T. Rybus, "Obstacle avoidance in space robotics: Review of major challenges and proposed solutions," *Progress in Aerospace Sciences*, vol. 101, pp. 31–48, 2018.

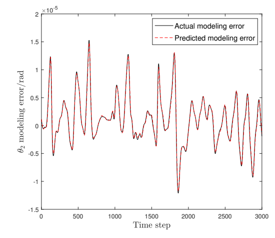
[2] B. S. Peters, P. R. Armijo, C. Krause, S. A. Choudhury, and D. Oleynikov, "Review of emerging surgical robotic technology," *Surgical Endoscopy and Other Interventional Techniques*, vol. 32, no. 4, pp. 1636–1655, 2018.

[3] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE conference on decision and control (CDC)*. IEEE, 2018, pp. 6059–6066.

[4] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[5] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[6] I. Jurado, P. Millán, D. Quevedo, and F. R. Rubio, "Stochastic mpc with applications to process control," *International Journal of Control*, vol. 88, no. 4, pp. 792–800, 2015.


(a) Modeling error of θ_2

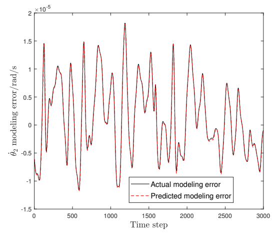
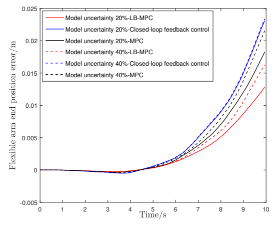

(b) Modeling error of $\dot{\theta}_2$

Fig. 5. Performance of the LSTM learning model


(a) Tracking error in x-direction

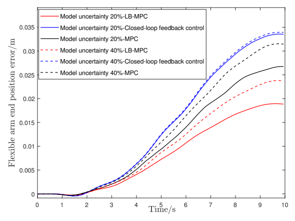
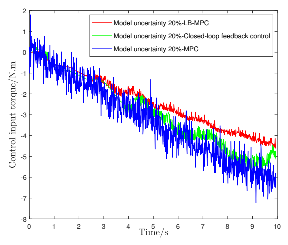
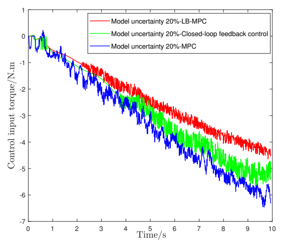
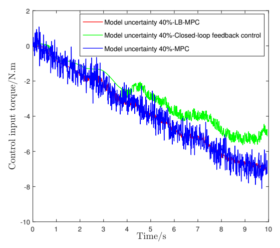

(b) Tracking error in y-direction

Fig. 6. Comparisons of tracking errors for the flexible link


(a) First joint input torque with model uncertainty 20%


(b) Second joint input torque with model uncertainty 20%


(c) First joint input torque with model uncertainty 40%

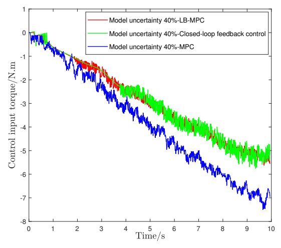
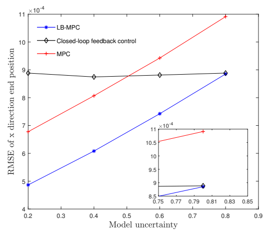

(d) Second joint input torque with model uncertainty 40%

Fig. 7. Comparisons of the joint input torque


(a) RMSE in x-direction

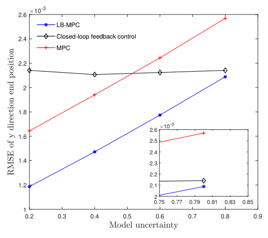

(b) RMSE in y-direction

Fig. 8. RMSE of the flexible link under different uncertainties

[7] T. Wakabayashi, Y. Suzuki, and S. Suzuki, “Dynamic obstacle avoidance for multi-rotor uav using chance-constraints based on obstacle velocity,” *Robotics and Autonomous Systems*, vol. 160, p. 104320, 2023.

[8] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, “Online trajectory generation with distributed model predictive control for multi-robot motion planning,” *IEEE robotics and automation letters*, vol. 5, no. 2, pp. 604–611, 2020.

[9] D. Q. Mayne, “Model predictive control: Recent developments and future promise,” *Automatica*, vol. 50, pp. 2967–2986, 2014.

[10] L. Dai, Y. Yu, D.-H. Zhai, T. Huang, and Y. Xia, “Robust model predictive tracking control for robot manipulators with disturbances,” *IEEE Transactions on Industrial Electronics*, vol. 68, no. 5, pp. 4288–4297, 2021.

[11] W. Zheng and B. Zhu, “Stochastic time-varying model predictive control for trajectory tracking of a wheeled mobile robot,” *Frontiers in Energy Research*, vol. 9, p. 767597, 2021.

[12] U. Rosolia, X. Zhang, and F. Borrelli, “Data-driven predictive control for autonomous systems,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 259–286, 2018.

[13] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.

[14] E. D. Klenske, M. N. Zeilinger, B. Scholkopf, and P. Hennig, “Gaussian process-based predictive control for periodic error correction,” *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 110–121, 2016.

[15] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, “Data-driven control of soft robots using koopman operator theory,” *IEEE transactions on robotics*, vol. 37, no. 3, pp. 948–961, 2021.

[16] X. Zhu, C. Ding, L. Jia, and Y. Feng, “Koopman operator based model predictive control for trajectory tracking of an omnidirectional mobile manipulator,” *Measurement and Control*, vol. 55, no. 9-10, pp. 1067–1077, 2022.

[17] F. Bonassi, M. Farina, J. Xie, and R. Scattolini, “On recurrent neural networks for learning-based control: Recent results and ideas for future developments,” *Journal of Process Control*, vol. 114, pp. 92–104, 2022.

[18] B.-K. Jeon and E.-J. Kim, “Lstm-based model predictive control for optimal temperature set-point planning,” *Sustainability*, vol. 13, p. 894, 2021.

[19] K. Huang, K. Wei, F. Li, C. Yang, and W. Gui, “Lstm-mpc: A deep learning based predictive control method for multimode process control,” *IEEE Transactions on Industrial Electronics*, pp. 1–10, 2022.

[20] D. Yu and M. Liu, “Dynamic modeling and stochastic feedback control of the space rigid-flexible manipulator,” in *2022 IEEE Conference on Guidance, Navigation and Control (ICGNC)*. Springer Nature, 2023, pp. 6304–6313.