Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 216 (2023) 213–220

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Conference on Computer Science and Computational Intelligence 2022

Long short-term memory (LSTM) model-based reinforcement learning for nonlinear mass spring damper system control

Santo Wijaya[a*], Yaya Heryadi[a], Yulyani Arifin[a], Wayan Suparta[b], Lukas[c]

[a]*Computer Science Department, BINUS Graduate Program – Doctor of Computer Science,Bina Nusantara University, Jakarta 11480, Indonesia.*
[b]*Department of Electrical Engineering, Faculty of Industrial Technology, Institut Teknologi Nasional Yogyakarta, Yogyakarta 55281, Indonesia.*
[c]*Cognitive Engineering Research Group (CERG), Universitas Katolik Indonesia Atma Jaya, Jakarta 12930, Indonesia.*

Abstract

The Neural Networks (NN) model which is incorporated in the control system design has been studied, and the results show better performance than the mathematical model approach. However, some studies consider that only offline NN model learning and does not use the online NN model learning directly on the control system. As a result, the controller's performance decreases due to changes in the system environment from time to time. The Reinforcement Learning (RL) method has been investigated intensively, especially Model-based RL (Mb-RL) to predict system dynamics. It has been investigated and performs well in making the system more robust to environmental changes by enabling online learning. This paper proposes online learning of local dynamics using the Mb-RL method by utilizing Long Short-Term Memory (LSTM) model. We consider Model Predictive Control (MPC) scheme as an agent of the Mb-RL method to control the regulatory trajectory objectives with a random shooting policy to search for the minimum objective function. A nonlinear Mass Spring Damper (NMSD) system with parameter-varying linear inertia is used to demonstrate the effectiveness of the proposed method. The simulation results show that the system can effectively control high-oscillating nonlinear systems with good performance.

Keywords: Model-based Reinforcement Learning; Random-Shooting Policy; LSTM; MPC; Nonlinear System

* Corresponding author. Tel.:+62-812-9906-4921.
E-mail address: santo.wijaya001@binus.ac.id

1877-0509
10.1016/j.procs.2022.12.129

1. Introduction

Research into NN has a lengthy history. Psychologist Donald Hebb [1] invented the oldest description of NN and created the Hebbian Learning scheme based on the brain plasticity process in the 1940s. In 1958, the perceptron, which is also known as the primary component of neural networks today and constructed as a two-layer neural network without a training method, was invented by Frank Rosenblatt [2]. In 1975, Paul Werbos created and published in his Ph.D. thesis of the back-propagation algorithm that is now the most extensively used [3]. In the following years, several NN architectures emerged, such as feedforward neural networks [4], Recurrent Neural Networks (RNN) [5], and others. A specific derivative model from RNN to solve the vanishing gradient was invented by Schmidhuber, which is called Long Short-Term Memory (LSTM) [6]. The LSTM is of great interest in the control system because they are particularly well-suited for prediction based on time series data, as there may be unpredictable delays or uncertainties between events in time series.

Pisa *et al*. [7] proposed the use of LSTM as a model in the Internal Model Control (IMC) scheme applied to a wastewater treatment plant. The result showed that the LSTM-based IMC approach improves the control metrics compared to the traditional PI controller. However, the authors used an offline learning approach to train the LSTM model, which is not robust enough to cope with the uncertainties that may occur in the system, especially uncertainties in the nonlinear system that might degrade control performance further. Sanaz *et al*. [8] proposed that the Mf-RL method be used to identify a dynamic model of a three-phase power converter by using the state-space Neural Network (ssNN) with Particle Swarm Optimization (PSO) to update the weights under parameter mismatch and incorporate them into a Model Predictive Control (MPC) scheme for control of a three-phase power converter. In the paper, the Mf-RL method succeeded in increasing the accuracy of the model in MPC model-based control. However, this approach did not directly calculate the control actions required in the control system.

The Reinforcement Learning (RL) method is a research topic that has been intensively investigated recently. The RL may be utilized to learn from the limited labeled data and derive additional essential insights from the unlabeled data [9]. Nagabandi *et al.* [10] proposed combining Mb-RL and Mf-RL methods to accomplish various MuJoCo [11] locomotion tasks. The author first utilizes the Mb-RL method combined with MPC for the initial movement task of the locomotion model to achieve robustness control with minimum sample data. The Mb-RL approach enables online learning to update a model based on reward, and the model can be used directly in the control system.

The paper proposed a combined offline and online learning approach utilizing the Mb-RL method. In offline learning, the LSTM architecture will be determined to have minimum weight parameters based on the dataset obtained from the controlled object of the system. Simple LSTM architecture is needed to learn faster in the online learning approach. Then online learning is done for every defined batch size to update the LSTM model online based on the given control action (u) and the measured state response of the system (y). We consider an MPC scheme with a random-shooting policy to search for the minimum objective function of trajectories error by calculating a set of random control actions within the given prediction horizon. The first control action of a set of random control actions that minimize the objective function is applied to the system every time step. We also consider the LSTM to learn from local dynamics, which is the rate of changes at each time step, to overcome the difficulties of learning when the difference of the state is too small, especially when the time step difference is small [10]. The NMSD system is used as an object to be controlled to show the effectiveness of the proposed method. The paper's primary contribution is two-fold: The online learning of local dynamics using the Mb-RL method utilizing LSTM as model-based in the MPC scheme with the Random-Shooting policy; Demonstrates the effectiveness of the proposed method for trajectory control of the NMSD system with parameters varying in the spring inertia and random disturbance.

2. Methods

The research method conducted in this paper consists of three phases: the initial phase, the design phase, and the simulation phase, as shown in Fig. 1. In the initial phase, we build the dynamic simulator using an NMSD system, then run it to obtain the dynamics data of systems input and outputs. The data will be divided into training, testing, and validation dataset to determine the minimum nodes of LSTM architecture with offline training. After that, in the design phase, the MPC is designed to use the LSTM model as a predictor to calculate the optimized action to

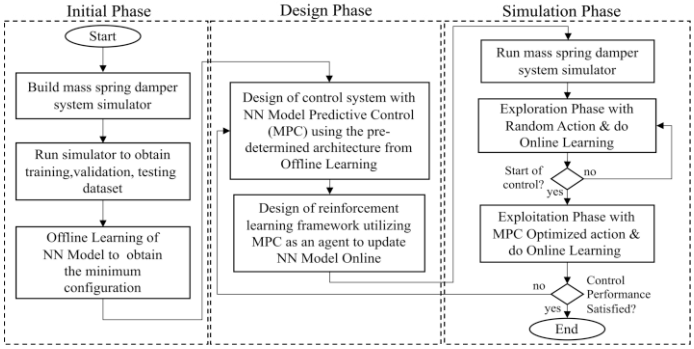


Fig. 1. Overall research method of the paper

minimize the objective function in the pre-determined horizon window. In addition, the MPC acts as an agent in the RL framework designed with a random-shooting policy. Lastly, in the simulation phase, we show the effectiveness of the proposed method in controlling the NMSD system to accomplish a regulatory objective.

2.1. Nonlinear Mass Spring Damper System

The NMSD system is an oscillation system consisting of a mass spring and damper elements that store kinetic and potential energy. This model can be used in many modeling approaches to model the system's dynamics, such as Flexible Bodies [12] and Mechanical Systems [13]. The NMSD system is particularly a mechanical system studied intensively in the control theory. However, it is still unavailable in the benchmark environment, such as MuJoco.

In the paper, the arbitrary free-body diagram of the NMSD system satisfying the motion of Newton's second law is shown in Fig. 2. It is assumed that the NMSD system consists of three masses or inertia of m_1, m_2, m_3 (kg), in this case identical masses considered where $m_1 = m_2 = m_3 = 0.5$ kg with initial positions y_1, y_2, y_3 (m). Damper damping constant also assumed to be identical with $d_1 = d_2 = d_3 = 0.5$ N. s/m respectively. Since the nonlinear spring is considered, the stiffness function is given by $k_1 = k_2 = k_3 = k_L x + k_{NL} x^3$ with parameter varying linear inertia of $k_L = 200 \pm 5$ N/m and nonlinear inertia of $k_{NL} = 65$ N/m^3. The control inputs of force to the NMSD system are u_1 and u_3. Then, $dist_3$ is a disturbance that disrupts the motion of the NMSD system; in this case, it is considered a random value between -200 N to 200 N. It is assumed that there is no measurement data for the position y_2, and the control actions are bounded between -1000 N to 1000 N.

Equation (1), (2), and (3) shows the NMSD system nonlinear differential equation function $g(y_k, u_k, dist_k)$ after derivation with Newton's law. SciPy Library will be used in the paper to obtain the dynamics of the NMSD differential equations using the *scipy.integrate.odeint* function with simulation time step $\Delta k = 0.001$ seconds, absolute error = 1.0×10^{-8} and relative error = 1.0×10^{-6}.

$$m\ddot{y}_1 = k_L(-2y_1 + y_2) + k_{NL}(-{y_1}^3 + (y_2 - y_1)^3) + d(\dot{y}_2 - 2\dot{y}_1) + u_1 \quad (1)$$

$$m\ddot{y}_2 = k_L(y_1 - 2y_2 + y_3) + k_{NL}((y_3 - y_2)^3 - (y_2 - y_1)^3) + d(\dot{y}_1 - 2\dot{y}_2 + \dot{y}_3) \quad (2)$$

$$m\ddot{y}_3 = k_L(y_2 - y_3) + k_{NL}(y_2 - y_3)^3 + d(\dot{y}_2 - \dot{y}_3) + u_3 - dist_3 \quad (3)$$

2.2. LSTM Neural Network Model

The LSTM model based on RNN is well-suited for learning time series or sequence data. Previous works [14, 15] comprehensively describe the LSTM network. This paper's LSTM model is constructed with Tensorflow and Keras Library [16] in the Google Colaboratory environment. Three layers are constructed, the Lambda layer, the LSTM layer and the Dense layer. The LSTM parameters are setup as follows, #return_sequence = True, #activation_function = Leaky ReLU ($\alpha = 0.5$), #recurrent_activation = Sigmoid, #look_back = 3, #loss_function = Mean Square Error, #optimizer = Adam (learning rate = 0.001), #epochs = 100, #batch_size = 16.

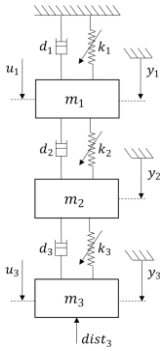


Fig. 2. Free-body diagram of the mass spring damper model

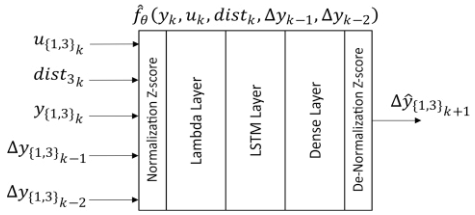


Fig. 3. LSTM Neural Network Model with Input – Output Parameters

Parameterization of an LSTM dynamics function is defined as $\hat{f}_\theta(y_k, u_k, dist_k)$, where θ denotes the vector of network weights. The input to the model is the control action (u_{1_k}, u_{3_k}), the measured disturbance $(dist_{3_k})$ and the measured state of the NMSD system (y_{1_k}, y_{3_k}) at every time step (k). The output of the model is the predicted state response $(\hat{y}_{1_{k+1}}, \hat{y}_{3_{k+1}})$ at next time step $(k+1)$. The paper also employed learning a dynamics function based on the rate of change in measurement data, $(\Delta y = y_k - y_{k-1})$, proposed by Nagabandi *et al*. [10], for better prediction results in the case of the small-time step (Δk). In this case, the predicted next state becomes $\Delta\hat{y}_{k+1} = \hat{y}_{k+1} - \hat{y}_k$. We considered the learned dynamics function is defined as $\hat{f}_\theta(y_k, u_k, dist_k, \Delta y_{k-1}, \Delta y_{k-2})$. The additional input to the model is the rate of change $\Delta y_{k-1} = y_k - y_{k-1}$, and $\Delta y_{k-1} = y_k - y_{k-2}$. Following that, the output of the model is changed to the rate of change of the predicted state response $(\Delta\hat{y}_{1_{k+1}}, \Delta\hat{y}_{3_{k+1}})$ at next time step $(k+1)$. Thus, totally the LSTM model require 9 input neurons in the Lambda layer and 2 output neurons in the Dense layer.

The paper also preprocessed the input dataset based on the Z-score normalization method [17] to standardize mean = 0 and standard deviation = 1 for fast training of the LSTM model. It is formulated as $x' = (x - \mu)/\sigma$, where x' denotes the normalize data, x denotes the source date, μ denotes the dataset mean, and σ denotes the dataset standard deviation. The Z-score normalization is chosen because the dataset feature distribution obtained from the NMSD system does not contain extreme outliers. Fig. 3 shows the overall configuration of the LSTM neural network model with input-output parameters and data preprocessing.

2.3. Model Predictive Control

The MPC scheme is well known in the control theory and has been intensively investigated in combination with the Machine Learning method and Neural Network model [18, 19]. The MPC scheme utilizes a model to make the system dynamics prediction in the pre-determined prediction horizon (N_p) and calculate a sequence of control action inside the prediction window to minimize the objective function (J) by optimization, apply the first control action from the calculated sequence at each time step, and then re-do the algorithm at the next time step. The input to the MPC controller (φ_k) is the vector of $u_{\{1,3\}_k}, dist_{3_k}, y_{\{1,3\}_k}$ from the NMSD system.

Fig. 4 shows the design of the LSTM model-based MPC scheme. The learned dynamics function of the LSTM model $\hat{f}_\theta(y_k, u_k, dist_k, \Delta y_{k-1}, \Delta y_{k-2})$ received vector input (φ_k) and calculate the predicted state response rate of change $(\Delta\hat{y}_{k|h})$ inside the prediction window (h) where the length of the horizon is N_p every time step (k). Then the MPC scheme calculate the objective function (J) of error between reference trajectory (r_k) and the predicted state response $(\hat{y}_{k|h})$. The minimization problem of J is solved with optimization solver to calculate a set of optimized control actions within the prediction horizon $(u_{k|h})$. But, to get the solution using optimization directly is difficult considering the model's nonlinearity. In this case, we employed a Random-Shooting policy [20] where R set of control action vector (U^R) with N_p length are generated, recalculate to obtain a set of the objective function (J^R), then find the minimum objective function (Ω) within the J^R set, and get the first control action $(u_{k|1})$ of U^R in which has the minimum objective function, finally apply the $u_{k|1}$ to the system for each time step.

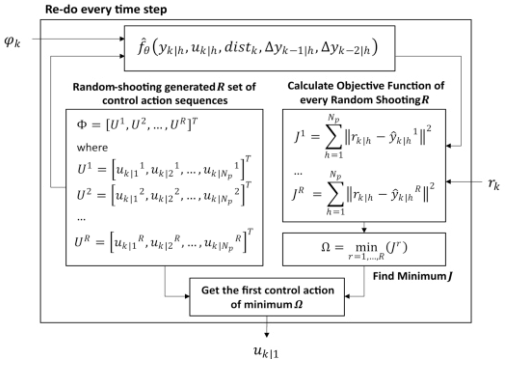


Fig. 4. Design of MPC Scheme with Random-Shooting Policy

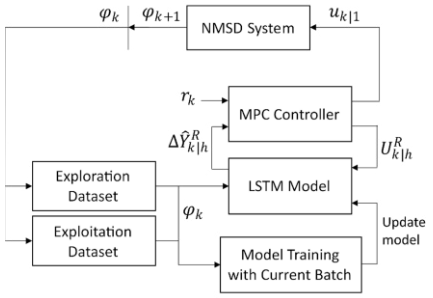


Fig. 5. The Model-based Reinforcement Learning Framework

2.4. Control Performance Evaluation

There are three control performance parameters in the paper. The trajectory deviation performance is calculated as a Root Mean Square Error $RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\hat{y}_k - y_{ref_k}\right)^2}$, where N denotes the simulation time steps, r denotes the reference values, ŷ denotes the predicted output values. The total impulse is formulated as $I_{tot} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(u_k)^2}$ and the action smoothness is formulated as $\Delta I_{tot} = \sqrt{\frac{1}{N}\sum_{k=1}^{N-1}(u_{k+1} - u_k)^2}$, where u denotes the control action.

2.5. Model-based Reinforcement Learning Framework

In the paper, the agent is the MPC controller, the model is the LSTM model, and the control inputs represent the action to the environment. The maximum reward is awarded if the system follows the reference trajectory or is considered as a minimization of RMSE. A framework of the Mb-RL in the paper, as shown in Fig. 5 and Algorithm 1. By exploration with random actions applied to the system and obtaining the exploration dataset of the first k_{norm} to be normalized with Z-score, which can be used to train $\hat{f}_\theta$ of the LSTM model, re-train $\hat{f}_\theta$ for every N_{batch} size. At $k \geq k_{cont}$, the MPC controller start to calculate optimized actions, then applied them to the system and obtained the exploitation dataset, re-train $\hat{f}_\theta$ is continue using the exploitation dataset. Retraining use 1 epochs, and use the weights of $\hat{f}_\theta$ from the previous iteration. The algorithm stopped at maximum simulation time.

Algorithm 1 Mb-RL Framework

Initialize $\hat{f}_\theta$ based on offline learning.
Set simulation time (k_{max}), normalization time (k_{norm}), batch size (N_{batch}), control time (k_{cont}).
Generate random action data points (u_{rand}).
For $k = 0$ **to** k_{max}
  If *isInteger(k / k_{norm})* **then**
    Do normalization to obtain new μ and σ.
  End If
  If *($k \geq k_{norm}$) AND isInteger(k / N_{bacth})* **then**
    Retrain $\hat{f}_\theta$ and clear N_{batch}
  End If
  If *($k < k_{cont}$)* **then**
    Set $u_k = u_{rand_k}$
  Elseif *($k \geq k_{cont}$)* **then**
    Get the current dataset ($y_k, u_k, dist_k$).
    Generate random R sets of random action sequences vector U^R with length N_p, $U^R = (u_{k|1}{}^R, u_{k|2}{}^R, \dots, u_{k|N_p}{}^R)$.
    For $i = 0$ **to** R
      Applied $\hat{f}_\theta$ to predict the next time step state.
      Calculate objective func. $J^R = \sum_{h=1}^{N_p} \left\| y_{ref_{k|h}} - \hat{y}_{k|h}{}^R \right\|^2$.
      Find minimum objective function J.
    End For
    Get the first control action $u_{k|1}$ from the minimum J.
    Set $u_k = u_{k|1}$
  End If
  Simulate the NMSD system $y_{k+1} = g(y_k, u_k, dist_k)$ with ODEsolver $\Delta k = 0.001$ seconds.
  Populate dataset ($y_k, u_k, dist_k$), batch $N_{bacth} = N_{bacth} + 1$.
  Reset objective function J
End For

3. Results and Discussions

3.1. Offline Learning

At this phase, the LSTM minimum configuration will be determined. Then, random action data points are generated to analyze the NMSD system and collect training, validation, and testing datasets through the system dynamic simulator. Fig. 6 shows the response of the NMSD system using 20.000 data points for training data. Fig. 8 shows the response of the NMSD system using 100.000 data points for training data which can be generated again for testing data. Finally, the normalization using Z-score is done using 20.000 data points before applying the dataset to the model for training, μ and σ will be calculated using the training dataset as shown in Fig. 7.

Once the dataset is obtained, offline training of the LSTM model is conducted. Fig. 9 shows the results of training and validation of the LSTM model with hidden node layers: 3, 5, and 10. Several activation functions have been experimented with (tanh, sigmoid, ReLU, Leaky ReLU with $\alpha = 0.3$); the best activation function is Leaky ReLU with $\alpha = 0.5$, notably, the model has a reasonable convergence rate and good validation result with only five layers of hidden nodes. However, the result with activation function Leaky ReLU with $\alpha = 0.5$ only is shown due to the space limitation.

Employing online learning means that a test of model accuracy only needed within the length of the prediction horizon N_p of MPC compared to the testing dataset. In this case, we consider using $N_p = 10$, the model's accuracy is based on the predicted ten-step-ahead state output. Even with a short horizon, it has been proven that an MPC controller can successfully control a robot's movement in a MuJoco environment [10]. Fig. 10 shows the test result of the ten-step-ahead prediction output and the RMSE $= 1.62 \times 10^{-4}$. Thus, we selected the LSTM with five layers of hidden nodes. The weight parameters obtained in the offline training will warm-start the model in the simulation phase.

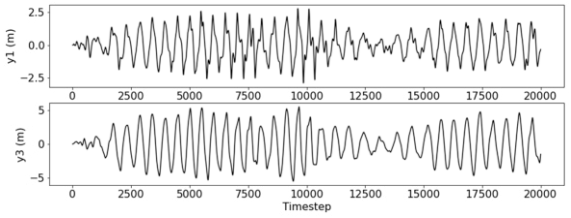


Fig. 6. Response of the NMSD system for training data

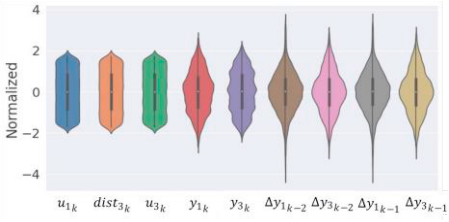


Fig. 7. Z-score normalization result of μ and σ

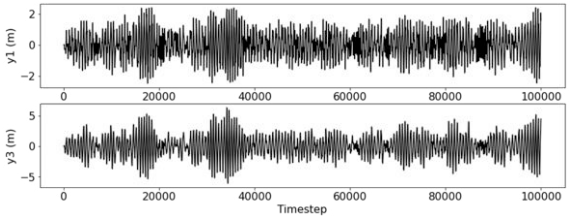
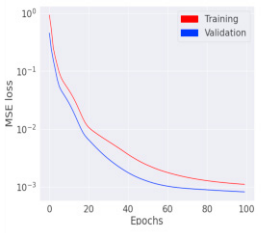


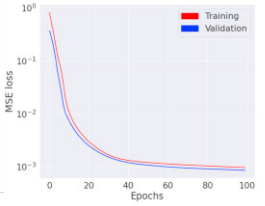
Fig. 8. Response of the NMSD system for validation data



LSTM Model with 3 Hidden Nodes

Layer (type)	Output_Shape	Param#
lambda (Lambda)	(None, 1, 9)	0
lstm (LSTM)	(None, 1, 3)	156
dense (Dense)	(None, 1, 2)	8

Total params: 164
Trainable params: 164
Non-trainable params: 0



LSTM Model with 5 Hidden Nodes

Layer (type)	Output_Shape	Param#
lambda(Lambda)	(None, 1, 9)	0
lstm (LSTM)	(None, 1, 5)	300
dense (Dense)	(None, 1, 2)	12

Total params: 312
Trainable params: 312
Non-trainable params: 0

Fig. 9. Training and validation result of the LSTM model

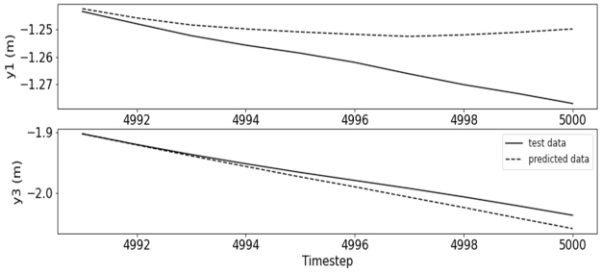


Fig. 10. Test result of 10 step-ahead prediction using LSTM 5 nodes

3.2. Online Learning

In the simulation phase, we setup $k_{max} = 40.900$ timesteps, $k_{norm} = 20.000$ timesteps, $N_{batch} = 16$ data points, $k_{cont} = 40.000$ timesteps, $N_p = 10$ horizons. It is also assumed that the actuators arbitrarily have a limit of $-500\ \text{N} \leq u_{\{1,3\}} \leq 500\ \text{N}$ before control action is started and $-1000\ \text{N} \leq u_{\{1,3\}} \leq 1000\ \text{N}$ after control action is started. Random disturbance of $dist_3$. Lastly, we also applied randomly distributed Gaussian noise to the measurement data of the NMSD system output, and there is a random parameter varying on the inertia of the spring $195\ \text{N/m} \leq k_L \leq 205\ \text{N/m}$. Then, there are two scenarios are considered for the random-shooting policy to generate R sets of random actions, U^R, randomly. Case A, to generate a set of U^R randomly between $250 \leq R \leq 350$ and case B to generate a set of U^R randomly between $50 \leq R \leq 150$. The control scenario is a regulatory objective to follow the pre-determined reference trajectory. There are three reference trajectories set up after control time (k_{cont}) is reached, (1) IF $k_{cont} \leq k < k_{cont} + 300$ THEN $y_{1,ref} = 1$ AND $y_{3,ref} = 2$, (2) IF $k_{cont} + 300 \leq k < k_{cont} + 600$ THEN $y_{1,ref} = -1$ AND $y_{3,ref} = 1$, (3) IF $k_{cont} + 600 \leq k < k_{cont} + 900$ THEN $y_{1,ref} = 0$ AND $y_{3,ref} = 0$.

As shown in Fig. 11, the simulation result indicated that the proposed method effectively controls the NMSD system according to the objective control scenario. Table 1 shows the control performance result based on Random-Shooting Policy (RSP) cases A and B. Notably, the RMSE and total Impulse performance in case A were better than in case B. In contrast, the performance of action smoothness in case B was better than in case A. The other important parameter is simulation time; case A, which has more extensive sets of R to search minimum objective function, produces better control performance but with a longer simulation time. We also compare the proposed LSTM model with a Multi-Layer Perceptron (MLP) model for better insights. In this comparison, case A of the Random-Shooting policy is selected. Furthermore, we selected the MLP model with 30 hidden nodes ($\theta = 362$) as it contains similar weight parameters as the LSTM model with five hidden nodes ($\theta = 312$). Table 2 shows the control performance comparison; notably, the LSTM model performs better in RMSE and action smoothness but with a longer simulation time.

Table 1. Control Performance of RSP Cases A and B

Case	$RMSE$	I_{tot}	ΔI_{tot}	Sim_Time
A	0.10875	736.97	269.31	20733.75
		740.79 (u_1)	271.66 (u_1)	
		733.14 (u_3)	266.95 (u_3)	
B	0.17674	754.97	255.60	6706.33
		746.38 (u_1)	266.31 (u_1)	
		763.55 (u_3)	244.89 (u_3)	

Table 2. Control Performance of RSP Case A of LSTM and MLP

Model	$RMSE$	I_{tot}	ΔI_{tot}	Sim_Time
LSTM 5 nodes	0.10875	736.97	269.31	20733.75
		740.79 (u_1)	271.66 (u_1)	
		733.14 (u_3)	266.95 (u_3)	
MLP 5 nodes	0.12348	718.45	279.63	10042.45
		727.14 (u_1)	266.98 (u_1)	
		709.75 (u_3)	292.27 (u_3)	

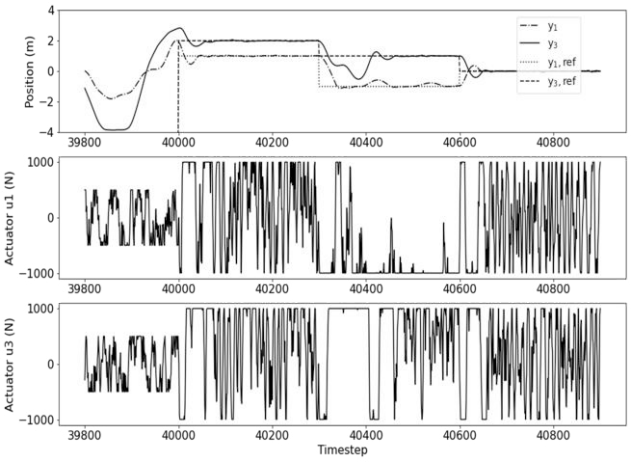


Fig. 11. Simulation result of Random Shooting Case A

4. Conclusions

Several studies considered only offline learning of the NN model and did not use the NN model directly in the control system. As a result, the environmental changes of the system over time might not be incorporated into the model; hence it may degrade the controller performance. The paper proposed online learning of local dynamics using the Mb-RL method utilizing the LSTM model to make the system more robust to environmental changes by enabling online learning. The MPC as an agent, provides a reward function calculated based on a random-shooting policy, which generates random control action and finds the minimum objective function through a closed-loop strategy. The online learning scheme retrains the LSTM model and minimizes model mismatch with the system. The NMSD system with varying parameters is employed to show the effectiveness of the proposed method. The simulation results showed that the proposed method effectively controls the system with good performance.

Extending the proposed method to a real-world mechanical system and investigating how the framework works on the real-time control system are considered for future research. The proposed approach requires considerable computing power to shorten the calculation time. This issue motivates the adoption of cloud-based control structures, where the controller is operated remotely as a cloud-based service, enabling a real-time control system.

References

[1] Brown RE. (2020) "Donald O. Hebb and the organization of behavior: 17 years in the writing." *Molecular Brain* **13** (**1**): 55.

[2] Rosenblatt F. (1958) "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review* **65** (**6**): 386–408.

[3] Werbos P. (1974) *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Ph.D. Dissertation, Harvard University, Cambridge.

[4] Glorot X, Bengio Y. (2010) "Understanding the difficulty of training deep feedforward neural networks." *Journal of Machine Learning Research* **9**: 249–256.

[5] Dong A, Du Z, Yan Z. (2019) "Round trip time prediction using recurrent neural networks with minimal gated unit." *IEEE Communication Letter* **23** (**4**): 584–587.

[6] Yu Wang. (2017) "A new concept using LSTM neural networks for dynamic system identification". In: *2017 American Control Conference (ACC)* pp. 5324–5329.

[7] Pisa I, Morell A, Vicario JL, et al. (2020) "Denoising autoencoders and LSTM-based artificial neural networks data processing for its application to internal model control in industrial environments—The wastewater treatment plant control case." *Sensors* **20** (**13**): 3743–3773.

[8] Sabzevari S, Heydari R, Mohiti M, et al. (2021) "Model-free neural network-based predictive control for robust operation of power converters." *Energies* **14** (**8**): 1–12.

[9] Ge Z, Song Z, Ding SX, et al. (2017) "Data mining and analytics in the process industry: The role of machine learning." *IEEE Access* **5**: 20590–20616.

[10] Nagabandi A, Kahn G, Fearing RS, et al. (2018) "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning". In: *Proc. 2018 IEEE Int. Conf. on Robotics and Automation* pp. 7559–7566.

[11] Todorov E, Erez T, Tassa Y. (2012) "MuJoCo: A physics engine for model-based control." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 5026–5033.

[12] Subedi D, Tyapin I, Hovland G. (2020) "Modeling and analysis of flexible bodies using lumped parameter method." In: *Proc 2020 IEEE 11th Int Conf Mech Intell Manuf Technol (ICMIMT)* pp. 161–166.

[13] Tijsseling AS, Hou Q, Bozkuş Z. (2018) "Moving liquid column with entrapped gas pocket and fluid-structure interaction at a pipe's dead end: A nonlinear spring-mass system." In: *ASME Conf. 2018 Pressure Vessels and Piping Division*.

[14] Terzi E, Bonassi F, Farina M, et al. (2021) "Learning model predictive control with long short-term memory networks." *International Journal of Robust and Nonlinear Control* **31** (**18**): 8877–8896.

[15] Nabipour M, Nayyeri P, Jabani H, et al. (2020) "Deep learning for stock market prediction." *Entropy* **22** (**8**): 840–863.

[16] Géron A. (2019) *Hands-on machine learning with scikit-learning, keras and tensorfow*.

[17] Altman EI. (2018) "Applications of distress prediction models: What have we learned after 50 years from the z-score models?" *International Journal of Financial Studies* **6** (**3**): 70–85.

[18] Carlet PG, Tinazzi F, Bolognani S, et al. (2019) "An effective model-free predictive current control for synchronous reluctance motor drives." *IEEE Transactions on Industry Applications* **55** (**4**): 3781–3790.

[19] Masti D, Bemporad A. (2021) "Learning nonlinear state–space models using autoencoders." *Automatica* **129**: 109666.

[20] Rao A V. (2009) "A survey of numerical methods for optimal control." *Advances in the Astronautical Sciences* **135** (**1**): 497–528.