Preprints of the 23rd IFAC World Congress
Busan, Republic of Korea, August 23-28, 2026

DQN-based Computationally Efficient Switching Model Predictive Control for Nonlinear Systems

Liqian Huang * **Tiantian Wu** * **Zhuofan Fu** * **Shuheng Zhao** *
Zhou Tian * **Jingyi Lu** *

* *MOE Key Laboratory of Smart Manufacturing in Energy Chemical Process, East China University of Science and Technology, Shanghai 200237, P.R. China (e-mail: Y30241017@mail.ecust.edu.cn, Y10230112@mail.ecust.edu.cn, Y10240126@mail.ecust.edu.cn, Y16240155@mail.ecust.edu.cn, tianzhou@ecust.edu.cn, corresponding author: Jylu_cise@ecust.edu.cn).*

Abstract:
Model predictive control (MPC) is a widely adopted control strategy in the process industries. However, its reliance on online optimization can lead to performance degradation under limited computational resources, particularly for highly nonlinear systems. Learning-based MPC addresses this issue to some extent through offline policy training, but it lacks accuracy in untrained regions. To mitigate this limitation, this paper proposes a switched MPC strategy within a reinforcement learning (RL) framework. Specifically, a Deep Q-Network (DQN) was trained to switch between an exact MPC controller and a learning-based MPC controller, using real-time system state information and uncertainty quantification (UQ) from the learning-based control policy. Numerical experiments on the control of a nonlinear continuous stirred tank reactor (CSTR) demonstrate that the proposed method achieves significantly reduced computation time and lower tracking error compared to conventional methods.

Keywords: model predictive control, learning-based control, reinforcement learning, switched control, ensemble learning, uncertainty quantification.

1. INTRODUCTION

MPC uses a dynamic model to predict future system behavior and compute control inputs in a receding-horizon manner. Owing to its ability to handle multivariable interactions, hard constraints, and nonlinear dynamics, MPC has been widely applied in process industries and robotics Poignet and Gautier (2000). However, solving an online optimization problem can impose a prohibitive computational burden on high-dimensional or fast dynamical systems.

To alleviate the computational burden of MPC in real-time applications, existing studies have primarily followed two directions. The first seeks to accelerate online optimization through warm-start strategies Marcucci and Tedrake (2020) and fast MPC schemes, such as real-time iteration Diehl et al. (2005), advanced-step MPC Yang and Biegler (2013), and path-following methods Ohtsuka (2004). Although these methods can reduce computation time, they do not eliminate the need for online optimization, which remains costly for problems with complex dynamics or long prediction horizons. The second strategy avoids online optimization by approximating or precomputing the MPC control law offline. Representative methods include explicit MPC Tondel et al. (2002), learning-based approximations of the MPC control law Chen et al. (2018), and end-to-end controllers Amos et al. (2018). Among these methods, learning-based control methods offer greater flexibility for nonlinear systems and high-dimensional settings, but their reliability may degrade when the system encounters states that are insufficiently represented in the training data.

To enhance the reliability of learning-based MPC surrogate controllers, switching strategies have been proposed as supervisory mechanisms that transfer control authority to a backup controller when the learned policy is deemed unreliable. To determine when switching should occur, existing methods typically rely on online feasibility checks, performance certificates, or manually specified triggering criteria Hose et al. (2023); Zhang et al. (2021). However, such predefined switching mechanisms usually require manual calibration and may not adapt well to changing operating conditions. To address this limitation, we propose an RL framework for switching between exact MPC and learning-based MPC. For this purpose, we adopt the ensemble-based uncertainty estimation method proposed in Lakshminarayanan et al. (2017) to quantify the online reliability of the learning-based MPC surrogate. This uncertainty estimate is included in the RL state to inform the policy of regions where the surrogate may be inaccurate. In addition, the reward function jointly accounts for control accuracy, uncertainty, and computational cost. This allows the DQN to learn a switching policy that balances the accurate but computationally expensive exact MPC with the efficient but approximate learning-based

©2026 the authors. Accepted by IFAC for publication under a Creative Commons Licence CC-BY-NC-ND

MPC. As a result, the controller can adapt its switching decisions to varying operating conditions, thereby improving both control performance and computational efficiency.

The remainder of this paper is organized as follows. Section 2 formulates the control problem and introduces exact MPC and learning-based MPC. Section 3 presents the DQN-based switching strategy with ensemble-based uncertainty estimation. Section 4 describes the nonlinear CSTR simulation setup and compares different MPC schemes and switching strategies. Section 5 concludes the paper and discusses future research directions.

2. PROBLEM FORMULATION

2.1 Formulation of Model Predictive Control

Consider a discrete-time nonlinear system described by

$$x_{k+1} = F(x_k, u_k), \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ denote the state and control input, respectively. At each sampling time k, given the current state x_k, MPC computes an optimal control sequence $U_k = [u_{k|k}, u_{k+1|k}, \ldots, u_{k+N-1|k}]$ over a finite prediction horizon N by solving a constrained optimization problem. Only the first element of this sequence is applied to the system, and the optimization is repeated at the next sampling time. The finite-horizon optimal control problem is formulated as:

$$\min_{U_k} J(x_k, U_k) = \sum_{j=0}^{N-1} \ell(x_{k+j|k}, u_{k+j|k}) + V_f(x_{k+N|k}), \tag{2}$$

subject to:

$$\begin{aligned} x_{k|k} &= x_k, \\ x_{k+j+1|k} &= F(x_{k+j|k}, u_{k+j|k}), \quad j = 0, \ldots, N-1, \\ u_{k+j|k} &\in \mathbb{U}, \quad j = 0, \ldots, N-1, \\ x_{k+j|k} &\in \mathbb{X}, \quad j = 1, \ldots, N, \end{aligned}$$

where $x_{k+j|k}$ denotes the predicted state at time $k+j$ based on the current state x_k and the input sequence U_k. The sets $\mathbb{U}$ and $\mathbb{X}$ denote the admissible control and state constraint sets, respectively. The stage cost function $\ell(\cdot,\cdot)$ penalizes both the tracking error and the control effort. It is typically chosen as a quadratic form, $\ell(x_k, u_k) = \|x_k - x_{\text{ref}}\|_Q^2 + \|u_k - u_{\text{ref}}\|_R^2$, where Q is a positive semidefinite state weighting matrix and R is a positive definite input weighting matrix. The quadratic terminal cost $V_f(\cdot)$ penalizes the terminal state deviation:

$$V_f(x) = \frac{1}{2}(x - x_{\text{ref}})^\top P (x - x_{\text{ref}}), \tag{3}$$

where P is the terminal weight matrix. Due to the nonlinear dynamics, the MPC problem is formulated as a nonlinear programming (NLP) problem. Tools such as CasADi Andersson et al. (2019) can be used to formulate and solve such NLP problems efficiently. However, solving the NLP online remains computationally demanding for high-dimensional or fast-sampling systems, which motivates learning-based MPC.

2.2 Learning-Based MPC

Learning-based MPC replaces online optimization with a parameterized policy π_θ that approximates the mapping from the current state x_k to the optimal MPC control input u_k^*. Given a dataset $\mathcal{D} = \{(x^{(i)}, u^{*(i)})\}_{i=1}^{N_s}$ generated by an exact MPC solver, the parameters of the MPC policy are learned by minimizing

$$\mathcal{L}(\theta) = \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \pi_\theta(x^{(i)}) - u^{*(i)} \right\|_2^2. \tag{4}$$

The surrogate controller is implemented using a multilayer perceptron (MLP). During offline training, the MLP can be optimized with mini-batch gradient-based optimization using the mean squared error loss in (4). After training, the learned policy directly computes the control input online through a forward pass, thereby avoiding repeated solution of the MPC optimization problem at each sampling instant. However, the reliability of the learned policy may degrade outside the distribution of the training data, which motivates the uncertainty-aware switching strategy developed in the next section.

3. DQN-BASED SWITCHING STRATEGY

This section presents an uncertainty-aware DQN framework for switching between learning-based MPC and exact MPC. As shown in Fig. 1, ensemble-based uncertainty estimates are used to guide the DQN switching policy. The following subsections describe the uncertainty estimation method and the DQN-based switching mechanism.

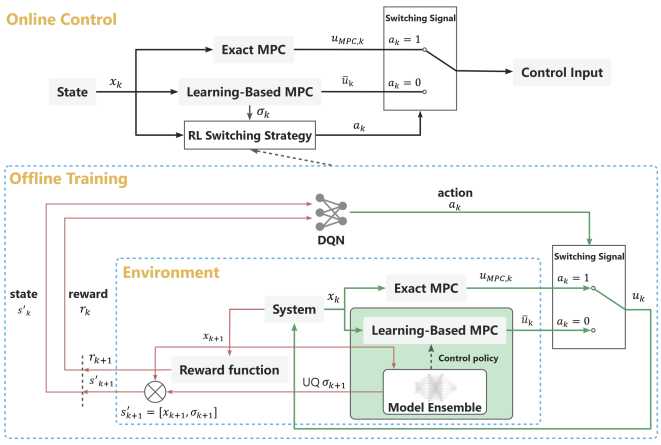


Fig. 1. Schematic illustration of the uncertainty-aware DQN switching framework.

3.1 Ensemble-Based Uncertainty Quantification

To guide the DQN switching policy, we quantify the uncertainty of the learning-based MPC using an ensemble of M independently trained neural policies $\{\pi_{\theta_i}\}_{i=1}^{M}$. For a given state, the disagreement among the control inputs predicted by the ensemble members is used as a measure of predictive uncertainty. Larger disagreement is interpreted as lower confidence in the learning-based MPC policy, indicating that exact MPC may be preferred. The ensemble is constructed using the training procedure described in Algorithm 1, where $M > 1$ neural networks are trained with different random initializations.

For each state x_k, the control predictions of all ensemble members are evaluated, and the empirical mean and a scalar disagreement measure are computed as

Algorithm 1 Ensemble Training for Learning-Based MPC Surrogates

Require: Training dataset $\mathcal{D} = \{(x^{(s)}, u^{*(s)})\}_{s=1}^{N_s}$, ensemble size M, number of epochs E, batch size B, learning rate α
Ensure: Trained ensemble $\{\pi_{\theta_i}\}_{i=1}^{M}$
for $i = 1$ to M **do**
  Initialize neural policy π_{θ_i} with random weights
  Initialize a gradient-based optimizer with learning rate α
  for epoch $= 1$ to E **do**
    Shuffle the training dataset $\mathcal{D}$
    for each mini-batch $\mathcal{B} \subset \mathcal{D}$ of size B **do**
      Compute predictions $\hat{u}^{(s)} = \pi_{\theta_i}(x^{(s)})$ for all $(x^{(s)}, u^{*(s)}) \in \mathcal{B}$
      Compute the loss

$$\mathcal{L}_i = \frac{1}{B} \sum_{(x^{(s)}, u^{*(s)}) \in \mathcal{B}} \left\| \hat{u}^{(s)} - u^{*(s)} \right\|_2^2$$

      Update θ_i using the optimizer
    end for
  end for
end for
return Trained ensemble $\{\pi_{\theta_i}\}_{i=1}^{M}$

$$\hat{u}_{i,k} = \pi_{\theta_i}(x_k), \quad i = 1, \ldots, M, \tag{5}$$

$$\bar{u}_k = \frac{1}{M} \sum_{i=1}^{M} \hat{u}_{i,k}, \tag{6}$$

$$\sigma_k = \sqrt{\frac{1}{M-1} \sum_{i=1}^{M} \|\hat{u}_{i,k} - \bar{u}_k\|_2^2}. \tag{7}$$

Here, $\bar{u}_k$ is used as the control input of the learning-based MPC, while σ_k measures the aggregate disagreement among ensemble predictions and serves as a practical indicator of predictive reliability. A large σ_k may suggest that the current state lies in a region poorly represented by the training data, implying a higher risk of control performance degradation, whereas a small σ_k suggests more reliable predictions.

3.2 DQN-Based Controller Switching

This subsection presents a DQN-based controller switching strategy, where the switching problem is modeled as a Markov decision process. At each time step k, the RL agent observes the augmented state $\mathbf{s}_k = [x_k, \sigma_k]$. Based on the augmented state, the switching policy can switch to exact MPC when the learning-based MPC is predicted to provide insufficient control performance. The switching action is selected according to the Q-values generated by the DQN:

$$a_k = \arg \max_{a \in \{0,1\}} Q_\phi(\mathbf{s}_k, a), \tag{8}$$

where $Q_\phi(\cdot, \cdot)$ denotes the action-value function parameterized by the Q-network. The actions $a_k = 0$ and $a_k = 1$ correspond to selecting learning-based MPC and exact MPC, respectively. The resulting control input is then given by

$$u_k = \begin{cases} \bar{u}_k & \text{if } a_k = 0, \\ u_{\mathrm{MPC},k} & \text{if } a_k = 1, \end{cases} \tag{9}$$

Algorithm 2 DQN Training for Controller Switching

Require: Environment *env*, discount factor γ, exploration parameters $\varepsilon_{\max}$, $\varepsilon_{\min}$, $\varepsilon_{\mathrm{decay}}$
Initialize online network Q_ϕ, target network $Q_{\phi'} \leftarrow Q_\phi$, replay buffer $\mathcal{R}$
Initialize target update frequency T, batch size B, $\varepsilon \leftarrow \varepsilon_{\max}$
for episode $= 1$ to N **do**
  Reset environment and observe initial state x_0
  for $k = 0$ to $K - 1$ **do**
    Construct augmented state $\mathbf{s}_k = [x_k, \sigma_k]$
    $a_k \leftarrow \begin{cases} \text{random action} & \text{w.p.} \varepsilon \\ \arg\max_{a \in \{0,1\}} Q_\phi(\mathbf{s}_k, a) & \text{otherwise} \end{cases}$
    Execute a_k, observe r_k, x_{k+1}, and construct $\mathbf{s}_{k+1}$
    Store $(\mathbf{s}_k, a_k, r_k, \mathbf{s}_{k+1})$ in $\mathcal{R}$
    if $|\mathcal{R}| > B$ **then**
      Sample batch $\{(\mathbf{s}_i, a_i, r_i, \mathbf{s}_{i+1})\} \sim \mathcal{R}$
      $y_i \leftarrow r_i + \gamma \max_{a' \in \{0,1\}} Q_{\phi'}(\mathbf{s}_{i+1}, a')$
      $\mathcal{L} \leftarrow \frac{1}{B} \sum_i (y_i - Q_\phi(\mathbf{s}_i, a_i))^2$
      Update ϕ using $\nabla_\phi \mathcal{L}$
    end if
    if $k \mod T = 0$ **then**
      $\phi' \leftarrow \phi$
    end if
    $\varepsilon \leftarrow \max(\varepsilon_{\min}, \varepsilon \cdot \varepsilon_{\mathrm{decay}})$
  end for
end for

where $\bar{u}_k$ is the ensemble mean defined in (6), and $u_{\mathrm{MPC},k}$ is obtained by solving the exact MPC problem at time step k. Having specified the action-to-control mapping, we next define the reward function used to train the switching policy. The reward is designed to encourage tracking improvement while penalizing the current tracking error, surrogate uncertainty, and the computation time incurred by exact MPC:

$$\begin{aligned} r_k = {} & w_i \cdot \max(0, \Delta e_k) - \eta \sigma_k \cdot \mathbb{I}\{a_k = 0\} \\ & - w_e \cdot e_k - w_t \cdot \max(0, t_{m,k} - \bar{t}_m) \cdot \mathbb{I}\{a_k = 1\}, \end{aligned} \tag{10}$$

where w_i, η, w_e, and w_t are constant weights associated with tracking improvement, uncertainty penalty, current tracking error, and computational cost, respectively. The tracking error e_k is defined as the weighted normalized distance between the current state and the reference state:

$$e_k \triangleq \sum_{i=1}^{n_x} v_i \left| \frac{x_{k,i} - x_i^{\mathrm{ref}}}{s_i} \right|, \qquad v_i > 0,\ s_i > 0, \tag{11}$$

where v_i is the weight assigned to the i-th state variable, and s_i is the corresponding normalization scale. The error improvement term is defined as $\Delta e_k \triangleq e_{k-1} - e_k$, which rewards reductions in the tracking error. The computation time term $t_{m,k} - \bar{t}_m$ penalizes the use of exact MPC when its solution time exceeds the prescribed threshold, where $t_{m,k}$ is the measured MPC solve time and $\bar{t}_m$ is the computation time threshold. The uncertainty penalty $\eta \sigma_k$ is applied when the learning-based MPC controller is selected.

Using the augmented state, the DQN accounts for both the current operating condition and the prediction uncertainty of the learning-based MPC when selecting the controller. Together with the proposed reward function, this enables

the agent to evaluate the trade-off between computational efficiency and control accuracy under varying operating conditions. The complete training procedure is detailed in Algorithm 2.

4. SIMULATION

4.1 System Model

A nonlinear CSTR benchmark from Fukatsu et al. (2024) is considered in this study. Its continuous-time dynamics are given by

$$\dot{C}_A = -k_1 C_A - k_2 C_A^2 + (C_{A,\text{in}} - C_A)u_1, \tag{12}$$

$$\dot{C}_B = k_1 C_A - k_1 C_B - C_B u_1, \tag{13}$$

$$\dot{T}_R = -\delta\left(k_1 C_A H_{AB} + k_1 C_B H_{BC} + k_2 C_A^2 H_{AD}\right) + \alpha(T_C - T_R) + (T_{\text{in}} - T_R)u_1, \tag{14}$$

$$\dot{T}_C = \beta(T_R - T_C) + \gamma u_2. \tag{15}$$

The temperature-dependent reaction rate constants are modeled as:

$$k_i = k_{i0} \exp\left(-\frac{E_i}{T_R + 273.15}\right), \quad i = 1, 2. \tag{16}$$

The key model parameters, operating conditions, and operational constraints used in this study are summarized in Table 1. For numerical simulation and controller implementation, the continuous-time model in (12)–(15) is discretized using the fourth-order Runge–Kutta method. The control objective is to regulate C_A, C_B, T_R, and T_C to their respective reference setpoints by manipulating the two input variables u_1 and u_2.

Table 1. Key CSTR Parameters and Operating Conditions

Symbol	Value/Range	Unit
k_{10}	1.287×10^{12}	1/h
k_{20}	9.043×10^{6}	1/h
E_1	9758.3	K
E_2	8560.0	K
α	30.828	1/h
β	86.688	1/h
δ	3.522×10^{-4}	–
γ	0.1	–
$C_{A,\text{in}}$	5.1×10^{3}	mol/m^3
T_{in}	104.9	°C
H_{AB}	4.2	–
H_{BC}	−11.0	–
H_{AD}	−41.85	–
C_A^{ref}	1.37×10^{3}	mol/m^3
C_B^{ref}	0.95×10^{3}	mol/m^3
T_R^{ref}	110.0	°C
T_C^{ref}	108.6	°C
u_1	$[3, 35]$, ref. 5.0	–
u_2	$[-9000, 0]$, ref. −1190.0	–

4.2 Surrogate and Switching Policy Training

The ensemble learning-based MPC models are trained and evaluated using datasets generated by uniformly sampling states within the ranges listed in Table 2 and computing the corresponding exact MPC control inputs using CasADi with IPOPT. The training set contains one million samples from the full operating range and is supplemented by 200,000 samples generated around the nominal setpoint using a gradient-divergent sampling strategy to improve near-steady-state accuracy and uncertainty quantification. An independent test set of 10,000 samples is drawn from an extended operating range. The ensemble consists of five learning-based MPC models with independent random initializations, each trained with Adam to minimize the mean squared error between the predicted and exact MPC control inputs.

Table 2. Sampling Ranges of State Variables

Variable	x_{ref}	Training Range	Test Range
C_A	1370	$[770.0, 1970.0]$	$[660.0, 2080.0]$
C_B	950	$[750.0, 1150.0]$	$[710.0, 1190.0]$
T_R	110.0	$[100.0, 120.0]$	$[98.0, 122.0]$
T_C	108.6	$[98.6, 118.6]$	$[96.6, 120.6]$

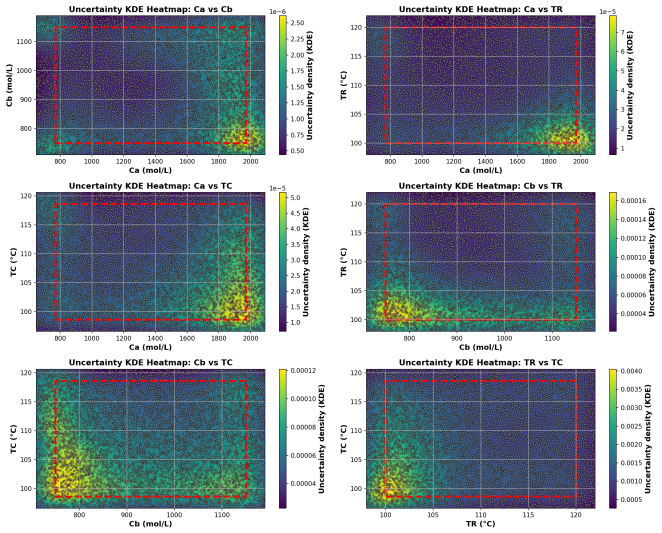


Fig. 2. Ensemble uncertainty heatmaps for pairwise state variables.

To examine whether the ensemble uncertainty reflects out-of-distribution (OOD) behavior, additional states are selected using Latin hypercube sampling from both the training range and the extended test range. Fig. 2 shows the resulting uncertainty heatmaps obtained using uncertainty-weighted two-dimensional kernel density estimation. The red dashed rectangles denote the training regions for each pair of state variables, and brighter regions indicate higher uncertainty. The results show that high-uncertainty regions are mainly located outside the training ranges, suggesting that the ensemble uncertainty can serve as a useful indicator of extrapolation risk.

After training the ensemble surrogate and constructing the uncertainty estimate, the DQN switching policy is trained for approximately 200,000 interaction steps. The reward parameters are set to $w_i = 2.5$, $\eta = 8.0$, $w_e = 3.5$, $w_t = 15.3$, and $\bar{t}_m = 0.005$ s. With the ensemble uncertainty estimate and the trained switching policy in place, the proposed framework is then evaluated in closed-loop simulations against several MPC and switching baselines.

4.3 Control Performance Comparison

This subsection compares the proposed DQN-switching MPC with exact MPC, single-model learning-based MPC, and two switching MPC baselines. The switching baselines include a threshold-based switching rule and the primal–dual verification method Zhang et al. (2021). The threshold-based method requires manual tuning of the threshold δ, whereas the primal–dual method is implemented on a locally parameter-varying linearized approximation of the CSTR and used as a heuristic baseline.

Table 3. Comparison of Threshold-Based and DQN Switching Policies

Strategy	Learning-based MPC Selection Rate (%)	Average Time (ms)	Average Tracking Error
Exact MPC	0.0%	10.894	178.6216
Threshold ($\delta = 1$)	0.21%	10.941	179.3240
Threshold ($\delta = 9$)	56.24%	5.127	181.8496
DQN with UQ	**64.39%**	**4.182**	**181.4134**
Threshold ($\delta = 20$)	79.84%	2.629	184.2416

The first comparison evaluates the proposed DQN-based switching strategy against the threshold-based switching method to examine state-dependent switching behavior. This comparison is conducted on the independent test set of 10,000 states drawn from the extended operating range. As shown in Table 3, the DQN switching strategy selects the learning-based MPC in 64.39% of the sampled states and achieves an average single-step computation time of 4.182 ms. Compared with the threshold-based policy with $\delta = 9$, it yields a lower tracking error while requiring less computation time. These results suggest that the proposed method can adaptively balance computational efficiency and tracking performance without manual threshold tuning.

We further evaluate the proposed method against the primal–dual strategy under identical operating conditions. The evaluation consists of three parts: a single-step control comparison, a multi-step closed-loop trajectory comparison, and a steady-state performance comparison. Single-step evaluation is conducted on the same independent test set, where each controller produces a one-step control input for every sampled state. Multi-step performance is evaluated over 20 independently sampled initial states, each simulated for 30 control steps. The tracking error is averaged over all 30 steps of each trajectory, providing a finite-horizon measure of transient closed-loop performance. For both the single-step and multi-step comparisons, the computation time and tracking error are averaged over the corresponding state-control samples. The relative single-step time and tracking error are normalized by the exact MPC results, which serve as the 100% baseline. As shown in Table 4, exact MPC achieves the lowest tracking error but the highest computation time, while learning-based MPC is the opposite. Among the switching strategies, DQN with UQ achieves the best overall trade-off, with the lowest computation time and tracking error in both single-step and multi-step evaluations. For steady-state evaluation, 100 initial states are randomly sampled and simulated in closed loop for a sufficiently long horizon. Unlike the multi-step metric, the steady-state error is computed solely from the weighted tracking error of the terminal state. As shown in Table 4, the primal–dual method and DQN without UQ exhibit the same steady-state error as learning-based MPC (25.47), since both switching strategies rely on the same single MLP-based surrogate and tend to keep selecting the learning-based MPC controller near the reference. In contrast, DQN with UQ achieves a notably lower steady-state error (9.73), as the ensemble-mean control input provides improved accuracy without requiring exact MPC in steady state. The steady-state error of exact MPC is nonzero (4.62) because the MPC formulation employed in this study is not specifically tuned for offset-free tracking; incorporating integral action or an offset-free MPC formulation could further reduce this residual error.

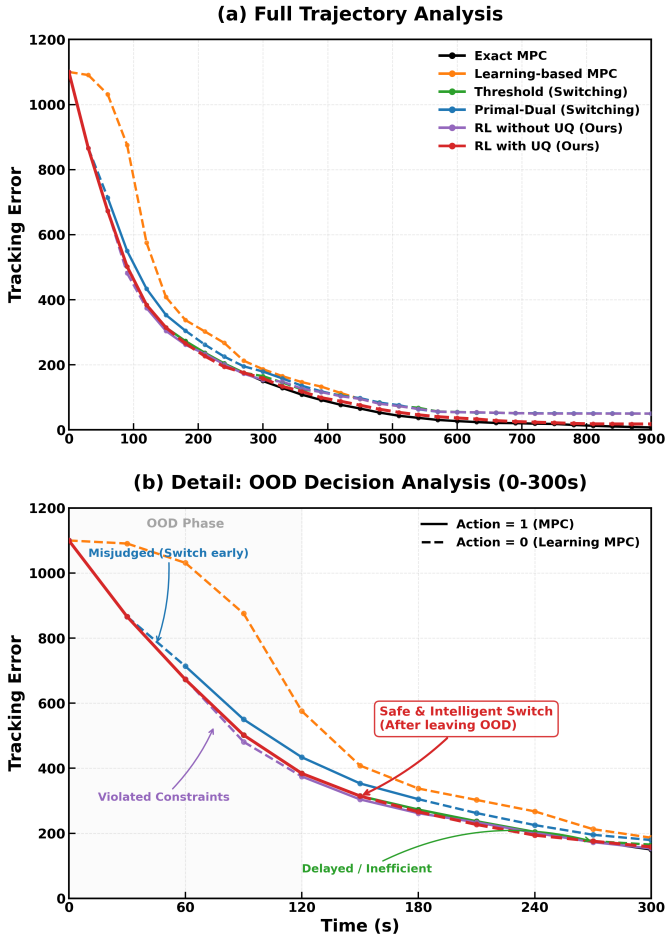


Fig. 3. OOD closed-loop comparison: (a) tracking error; (b) switching decisions.

The trajectories under an OOD initial condition further illustrate the adaptive and reliability-aware behavior of the proposed switching method. As shown in Fig. 3, DQN with UQ keeps exact MPC active during the high-uncertainty transient and switches to the learned controller only after the state enters a region of lower predictive uncertainty. In contrast, the primal–dual baseline and DQN without UQ switch earlier and exhibit less favorable transient behavior. These results suggest that uncertainty information improves switching decisions in OOD regimes while enabling a favorable trade-off between computational efficiency and control quality.

Table 4. Performance Comparison

Metric	Exact MPC	Learning-based MPC	Primal–Dual	DQN without UQ	DQN with UQ (proposed)
Avg. Single-step Computational Time (ms)	10.894	0.383	5.126	5.657	**4.182**
Avg. Single-step Tracking Error	178.6216	195.0274	183.0726	190.8071	**181.4134**
Relative Single-step Time (%)	100.00%	3.52%	47.05%	51.93%	**38.39%**
Relative Single-step Tracking Error (%)	100.00%	109.18%	102.49%	106.82%	**101.56%**
Avg. Multi-step Computational Time (ms)	7.317	0.673	3.360	2.831	**2.490**
Avg. Multi-step Tracking Error	36.53	44.54	44.02	43.19	**41.40**
Steady-State Error (e_{ss})	4.62	25.47	25.47	25.47	**9.73**

5. CONCLUSION

In this paper, we have proposed a hybrid learning-based MPC strategy that dynamically switches between an exact MPC solver and a pre-trained learning-based control policy. The switching logic is determined by a DQN, which uses the current system state along with uncertainty quantification from the learning-based policy as its input. This design effectively balances control performance against online computational burden. Numerical experiments on a nonlinear CSTR system demonstrated that the proposed method achieves a significant reduction in computation time while maintaining lower tracking errors compared to conventional approaches. For future work, we will focus on a rigorous theoretical analysis of closed-loop system stability and constraint satisfaction guarantees under the proposed switching framework. Furthermore, to address inevitable model mismatch, we plan to integrate controllers equipped with online parameter identification into the RL decision-making framework.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China under Grant 62394343, the Key Program of the National Natural Science Foundation of China under Grant 62136003, the Liaoning Provincial Science and Technology Program Joint Fund under Grant 2023JH26/10100003, the Programme of Introducing Talents of Discipline to Universities (111 Project) under Grant B17017, and the Fundamental Research Funds for the Central Universities under Grant 222202517006.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the authors used DeepSeek for assistance with grammar checking. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

REFERENCES

Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J.Z. (2018). Differentiable mpc for end-to-end planning and control. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

Chen, S., Saulnier, K., Atanasov, N., Lee, D.D., Kumar, V., Pappas, G.J., and Morari, M. (2018). Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American Control Conference (ACC)*, 1520–1527.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736.

Fukatsu, T., Hishinuma, T., and Aramaki, D. (2024). *Learning Model Predictive Control with Python and CasADi*. KS Series on Engineering and Technology. Kodansha.

Hose, H., Köhler, J., Zeilinger, M.N., and Trimpe, S. (2023). Approximate non-linear model predictive control with safety-augmented neural networks. abs/2304.09575.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. 6405–6416. Curran Associates Inc., Red Hook, NY, USA.

Marcucci, T. and Tedrake, R. (2020). Warm start of mixed-integer programs for model predictive control of hybrid systems. 66, 2433–2448.

Ohtsuka, T. (2004). A continuation/gmres method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4), 563–574.

Poignet, P. and Gautier, M. (2000). Nonlinear model predictive control of a robot manipulator. In *6th International Workshop on Advanced Motion Control. Proceedings (Cat. No.00TH8494)*, 401–406.

Tondel, R., Johansen, T., and Bemporad, A. (2002). An algorithm for multi-parametric quadratic programming and explicit mpc solutions.

Yang, X. and Biegler, L.T. (2013). Advanced-multi-step nonlinear model predictive control. *Journal of Process Control*, 23(8), 1116–1128.

Zhang, X., Bujarbaruah, M., and Borrelli, F. (2021). Near-optimal rapid mpc using neural networks: A primal-dual policy learning framework. *IEEE Transactions on Control Systems Technology*, 29(5), 2102–2114.