Surrogate modeling for adaptive predictive control over parameter spaces

Hassan Iqbal[1]

Joint work with:
Xingjian Li[1], Tyler Ingebrand[1], Adam Thorpe[1], Krishna Kumar[1], Ufuk Topcu[1], Ján Drgoňa[2]

[1]The University of Texas at Austin
[2]Johns Hopkins University

Iqbal, H. et al. (2025). "Zero-Shot Function Encoder-Based Differentiable Predictive Control". In: *arXiv preprint arXiv:2511.05757*

Nov 13, 2025
IMSI, University of Chicago
Workshop on Reduced Order and Surrogate Modeling for Digital Twins

Reduced Order and Surrogate Modeling for Digital Twins

November 10 — 14, 2025

This event is part of
Digital Twins

View Details

Description

BACK TO TOP

In many digital twin (DT) applications, the complexity of the forward models, the high dimensionality of the inference parameter and decision variable spaces, the need for real-time response, and the imperative of accounting for uncertainties all conspire to make the underlying inverse and optimal control problems intractable using high fidelity forward models. Surrogates and reduced order models (ROMs) can make these tasks tractable, provided they are sufficiently accurate and can be constructed with sufficiently few forward model solves.

Specific challenges arising in the DT setting and that will be addressed in this workshop include: (1) The surrogates/ROMs need not represent the full spatiotemporal system dynamics well, but only the control objectives and data assimilation observables—how this "goal-orientation" is best done remains a challenge; (2) since DTs typically evolve the dynamics over long time periods, there is a need to make ROMs structure preserving (e.g., energy conserving); (3) Neural network representations have shown much promise as surrogates in high dimensions, but work remains to be done to provide guarantees of their trustworthiness, particularly in the few data regime; (4) the surrogates/ROMs must be parametric with respect to not just state space, but also control variable space and uncertain parameter space, since the DT framework executes data assimilation and control problems repeatedly over a moving horizon; (5) many methods for surrogates rely on an intrinsically low-dimensional map from parameters to outputs of interest, and for ROMs an intrinsically low-dimensional solution manifold, yet linear subspaces may not capture this low-dimensionality efficiently for certain classes of problems (e.g., high frequency wave propagation, advection-dominated flow and transport); and (6) using surrogates trained on samples of high-fidelity input–output maps and not their Jacobians can result in poor approximation of gradients, leading to inaccurate solutions of optimization problems underlying data assimilation and optimal control.

Model Predictive Control (MPC)

MPC solves online the following discrete-time optimal control problem (OCP),

$$\begin{aligned}
\min_{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}} \quad & \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + p_N(\mathbf{x}_N) \\
\text{s.t. } & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \;\; k \in \mathbb{N}_0^{N-1} \\
& h(\mathbf{x}_k) \leq 0 \\
& g(\mathbf{u}_k) \leq 0 \\
& \mathbf{x}_0 = \mathbf{x}(t)
\end{aligned}$$

Model Predictive Control (MPC)

MPC solves online the following discrete-time optimal control problem (OCP),

$$\begin{aligned}\min_{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}} \quad & \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + p_N(\mathbf{x}_N) \\ \text{s.t. } & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \;\; k \in \mathbb{N}_0^{N-1} \\ & h(\mathbf{x}_k) \leq 0 \\ & g(\mathbf{u}_k) \leq 0 \\ & \mathbf{x}_0 = \mathbf{x}(t)\end{aligned}$$

A typical reference-tracking objective with a reference $\mathbf{r}_k$ and control action penalty,

$$\ell(\mathbf{x}_k, \mathbf{u}_k, \mathbf{r}_k) = ||\mathbf{x}_k - \mathbf{r}_k||_2^2 + ||\mathbf{u}_k||_2^2$$

The system dynamics are assumed to be differentiable,

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \quad \text{or} \quad \mathbf{x}_{k+1} = \text{ODESolve}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)).$$

This is the discretize-then-optimize approach to OCP.

Differentiable Predictive Control (DPC)

DPC (Drgoňa et al. 2024) is sampling-based strategy with expected risk minimization loss for approximately solving the following OCP,

$$\begin{aligned}
\min_{\pi \in \Pi} \quad & \mathbb{E}_{\mathbf{x}_0 \sim P_{\mathbf{x}_0}, \boldsymbol{\xi} \sim P_{\boldsymbol{\xi}}, \boldsymbol{\nu} \sim P_{\boldsymbol{\nu}}} \left(\int_0^T \ell(\mathbf{x}(t), \mathbf{u}(t); \boldsymbol{\xi}) \mathrm{d}t + p_T(\mathbf{x}(T)) \right) \\
\text{s.t.} \quad & \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t); \boldsymbol{\nu}), \\
& \mathbf{u}(t) = \pi(\mathbf{x}(t); \boldsymbol{\xi}, \boldsymbol{\nu}), \\
& h(\mathbf{x}(t); \boldsymbol{\xi}) \leq 0, \\
& g(\mathbf{u}(t); \boldsymbol{\xi}) \leq 0,
\end{aligned}$$

Differentiable Predictive Control (DPC)

DPC (Drgoňa et al. 2024) is sampling-based strategy with expected risk minimization loss for approximately solving the following OCP,

$$\begin{aligned}
\min_{\pi \in \Pi} \quad & \mathbb{E}_{\mathbf{x}_0 \sim P_{\mathbf{x}_0}, \boldsymbol{\xi} \sim P_{\boldsymbol{\xi}}, \boldsymbol{\nu} \sim P_{\boldsymbol{\nu}}} \left(\int_0^T \ell(\mathbf{x}(t), \mathbf{u}(t); \boldsymbol{\xi}) \mathrm{d}t + p_T(\mathbf{x}(T)) \right) \\
\text{s.t.} \quad & \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t); \boldsymbol{\nu}), \\
& \mathbf{u}(t) = \pi(\mathbf{x}(t); \boldsymbol{\xi}, \boldsymbol{\nu}), \\
& h(\mathbf{x}(t); \boldsymbol{\xi}) \leq 0, \\
& g(\mathbf{u}(t); \boldsymbol{\xi}) \leq 0,
\end{aligned}$$

We use penalty method to include the constraints in the loss function,

$$\begin{aligned}
\min_{\mathbf{W}} \quad & \mathbb{E}_{\mathbf{x}_0 \sim P_{\mathbf{x}_0}, \boldsymbol{\xi} \sim P_{\boldsymbol{\xi}}, \boldsymbol{\nu} \sim P_{\boldsymbol{\nu}}} \left[\sum_{k=0}^{N-1} \Big(\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) + p_h(h(\mathbf{x}_k, \boldsymbol{\xi})) + p_g(g(\mathbf{u}_k, \boldsymbol{\xi})) \Big) + p_N(\mathbf{x}_N) \right] \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t); \boldsymbol{\nu}) \mathrm{d}t, \quad \mathbf{u}_k = \pi_{\mathbf{W}}(\mathbf{x}_k; \boldsymbol{\xi}, \boldsymbol{\nu}).
\end{aligned}$$

- Assumes known differentiable dynamics and objectives for policy optimization.
- This can be considered as discretize-then-sample-and-learn approach.

Model, data-driven or learning-based control: comparison

Method	**Online cost**	**Offline cost**	**Constraint-aware?**	**Handles unknown dynamics?**
Classical MPC	✗ high online cost	✓ zero offline cost	✓ industry standard	✗ needs recursive system ID*
Approx. MPC (supervised)	✓	✗ requires labelled dataset	✓ postprocessing & verification	✗ requires online correction
DPC (self-supervised)	✓	✓ low cost: no labelled data needed	✓ constraints embedded during training	✗ needs known model
Goal	✓	✓	✓	✓

Modeling the Dynamics Using Function Encoders

- FE (Ingebrand et al. 2024) learns a set of NODE basis functions $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_B\}$ that span a subspace $\hat{\mathcal{F}} = \text{span}\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_B\}$ supported by the data.
- Given some $\boldsymbol{\nu}$, the dynamics are approximated as $\mathbf{f} \approx \hat{\mathbf{f}} \in \hat{\mathcal{F}}$, which takes the form of a linear combination of the learned basis functions under time discretization,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t); \boldsymbol{\nu})\, \mathrm{d}t \approx \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \sum_{j=1}^{B} c_j(\boldsymbol{\nu})\, \mathbf{g}_j(\mathbf{x}(t), \mathbf{u}_k; \boldsymbol{\theta}_j)\, \mathrm{d}t$$

$$\approx \mathbf{x}_k + \sum_{j=1}^{B} c_j(\boldsymbol{\nu}) \int_{t_k}^{t_{k+1}} \mathbf{g}_j(\mathbf{x}(t), \mathbf{u}_k; \boldsymbol{\theta}_j)\, \mathrm{d}t \quad (1)$$

  where $\boldsymbol{\theta}_j$ are the network parameters of the basis network $\mathbf{g}_j$.
- Importantly, the NODE basis functions $\{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_B\}$ do not depend explicitly on $\boldsymbol{\nu}$; as such, the dynamics function is uniquely determined by the coefficients $\mathbf{c} \in \mathbb{R}^B$.

Offline Learning of FE Basis Functions for System Dynamics

- The coefficients $\mathbf{c}$ to some $\mathbf{f} \in \mathcal{F}$ can be computed in closed-form via the normal equation as $(\mathbf{G} + \lambda \mathbf{I})\mathbf{c} = \mathbf{F}$.
- Here, $\mathbf{G}_{ij} = \langle \mathbf{g}_i, \mathbf{g}_j \rangle$ and $\mathbf{F}_i = \langle \mathbf{f}, \mathbf{g}_i \rangle$ can both be easily computed using Monte Carlo integration from a small amount of input-output data collected online.

Offline Learning of FE Basis Functions for System Dynamics

- The coefficients $\mathbf{c}$ to some $\mathbf{f} \in \mathcal{F}$ can be computed in closed-form via the normal equation as $(\mathbf{G} + \lambda \mathbf{I})\mathbf{c} = \mathbf{F}$.
- Here, $\mathbf{G}_{ij} = \langle \mathbf{g}_i, \mathbf{g}_j \rangle$ and $\mathbf{F}_i = \langle \mathbf{f}, \mathbf{g}_i \rangle$ can both be easily computed using Monte Carlo integration from a small amount of input-output data collected online.

Algorithm 2 (using $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \cdot \mathbf{f}(\mathbf{x_k}, \mathbf{u_k}; \boldsymbol{\nu})$ for simplicity).

Input: set of datasets $\mathcal{D}$ collected from dynamics $\mathcal{F}$, learning rate α
Initialize basis functions $\mathbf{g}_1, \ldots, \mathbf{g}_B$ with trainable parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_B$
while not converged **do**
  for all $\mathcal{D}_l \in \mathcal{D}$ **do**
    reset loss $L = 0$
    for all $(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1}) \in \mathcal{D}_i$ **do**
      $\mathbf{c} \leftarrow (\mathbf{G} + \lambda \mathbf{I})^{-1}\mathbf{F}$
      $\hat{\mathbf{x}}_{k+1} \leftarrow \hat{\mathbf{x}}_{k+1}$ from (1)
      $L \leftarrow L + \|\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}\|_2^2$
    end for
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L$
  end for
end while
Output: trained basis functions $\mathbf{g}_1, \ldots, \mathbf{g}_B$

DPC with FE-NODE Dynamics

$$\min_{\mathbf{W}} \quad \mathbb{E}_{\mathbf{x}_0 \sim P_{\mathbf{x}_0},\, \boldsymbol{\xi} \sim P_{\boldsymbol{\xi}},\, \mathbf{c} \sim P_{\mathbf{c}}} \left[\sum_{k=0}^{N-1} \Big(\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) + p_h(h(\mathbf{x}_k, \boldsymbol{\xi})) + p_g(g(\mathbf{u}_k, \boldsymbol{\xi})) \Big) + p_N(\mathbf{x}_N) \right] \tag{2}$$

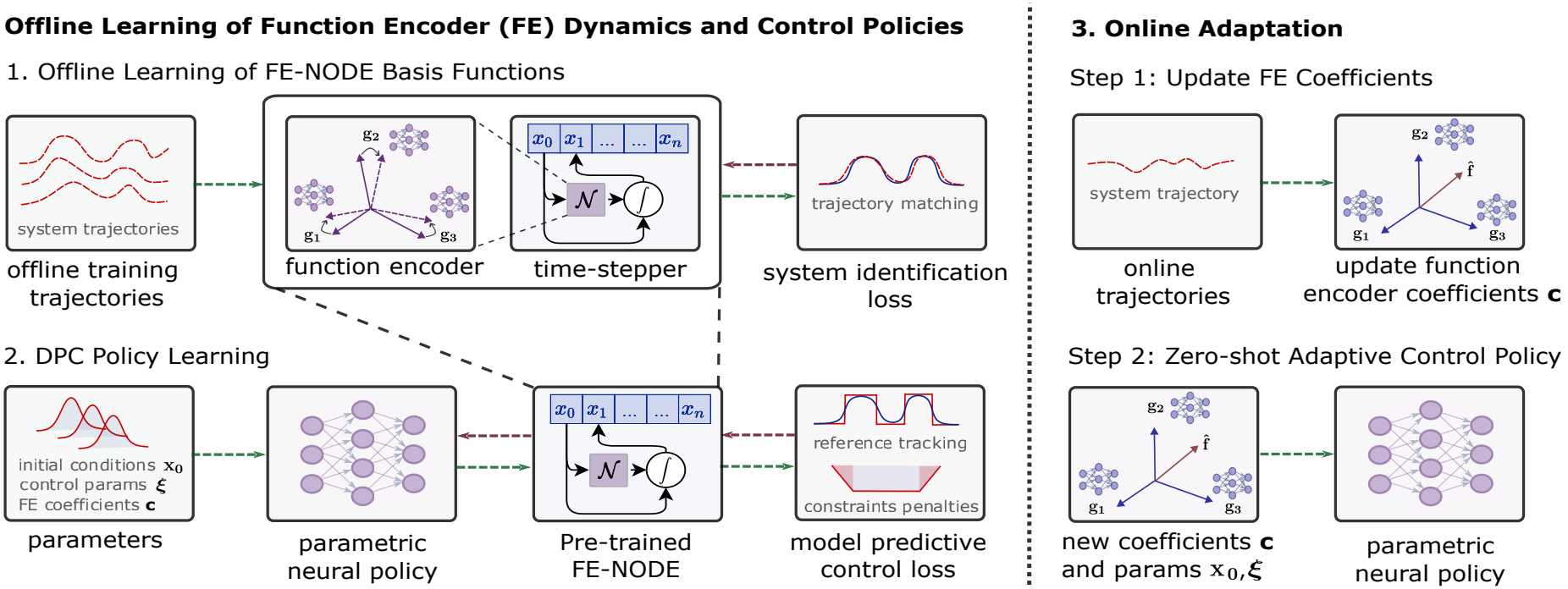
$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \sum_{j=1}^{B} c_j\, \mathbf{g}_j(\mathbf{x}(t), \mathbf{u}_k; \boldsymbol{\theta}_j)\, \mathrm{d}t, \tag{3}$$

$$\mathbf{u}_k = \pi_{\mathbf{W}}(\mathbf{x}_k; \boldsymbol{\xi}, \mathbf{c}). \tag{4}$$

Algorithm 3
Offline learning of parametric neural policies via DPC

Input: pre-trained NODE basis functions $\mathbf{g}_1, \ldots, \mathbf{g}_B$, learning rate β
Initialize policy network $\pi_{\mathbf{W}}$ with parameters $\mathbf{W}$
while not converged **do**
  for all $\mathbf{x}_0 \sim P_{\mathbf{x}_0}$, $\boldsymbol{\xi} \sim P_{\boldsymbol{\xi}}$, $\mathbf{c} \sim P_{\mathbf{c}}$ **do**
    loss $L \leftarrow 0$
    for $k = 0, 1, N-1$ **do**
      $\mathbf{u}_k \leftarrow \pi_{\mathbf{W}}(\mathbf{x}_k; \boldsymbol{\xi}, \mathbf{c})$
      $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_{k+1}$ from (3)
    end for
    $L \leftarrow L$ from (2)
    $\mathbf{W} \leftarrow \mathbf{W} - \beta \nabla_{\mathbf{W}} L$
  end for
end while
Output: trained policy network $\pi_{\mathbf{W}}$

DPC with FE-NODE Dynamics and Neural Policies



Example 1: Stabilizing a Van der Pol Oscillator

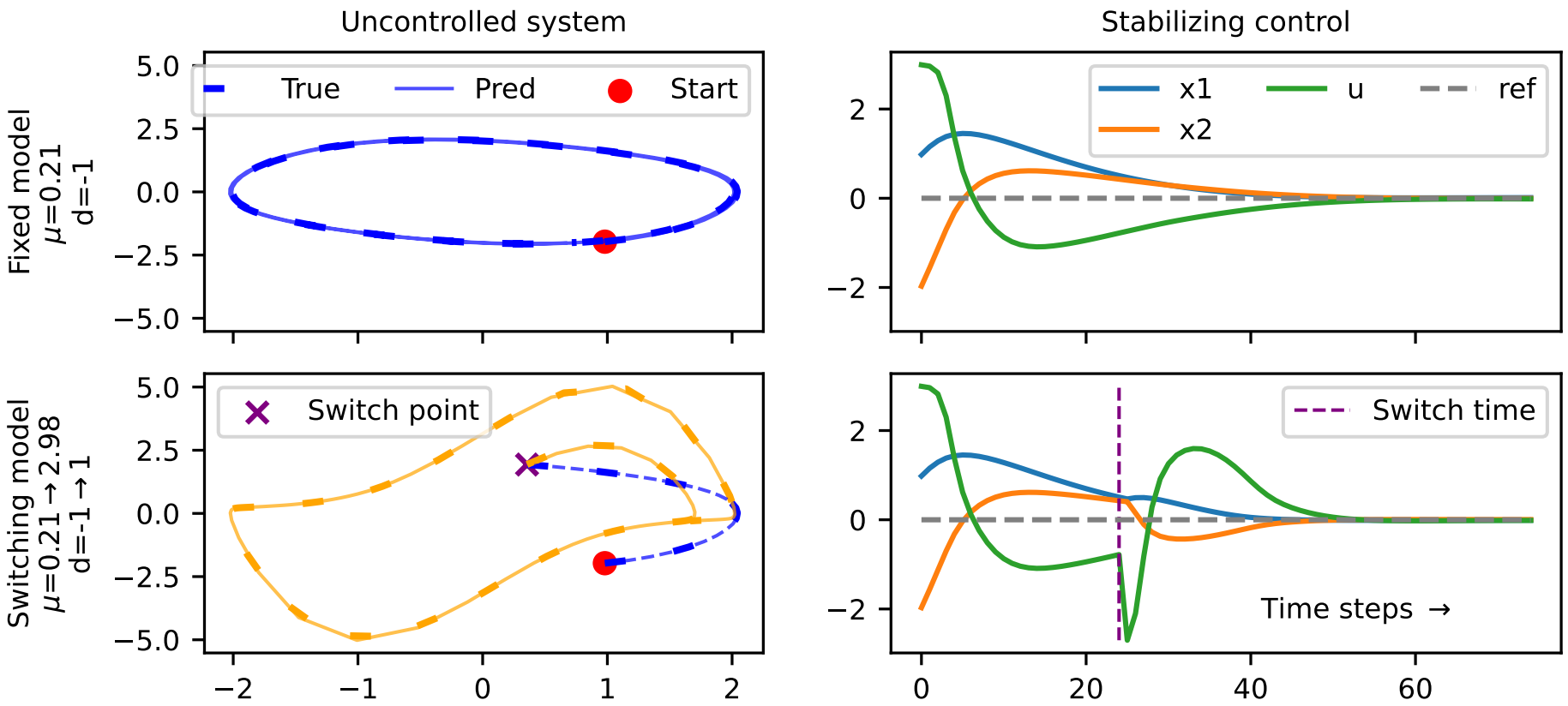
$$\begin{cases} \dot{x}_1 &= d \cdot x_2, \\ \dot{x}_2 &= \mu(1 - x_1^2)x_2 - x_1 + u, \end{cases}$$

where $[x_1, x_2]^\top \in [-2, 2] \times [-5, 5]$, $u \in [-3.0, 3.0]$, and parameters $\mu \sim \mathcal{U}[0.1, 3.0]$ and $d \in \{-1, 1\}$ determine the dynamics. The objective is, $p_N(\mathbf{x}_N) = \|\mathbf{x}_N\|^2$ and $\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) = \|\mathbf{u}_k\|^2$.

Example 1: Stabilizing a Van der Pol Oscillator

$$\begin{cases} \dot{x}_1 &= d \cdot x_2, \\ \dot{x}_2 &= \mu(1 - x_1^2)x_2 - x_1 + u, \end{cases}$$

where $[x_1, x_2]^\top \in [-2, 2] \times [-5, 5]$, $u \in [-3.0, 3.0]$, and parameters $\mu \sim \mathcal{U}[0.1, 3.0]$ and $d \in \{-1, 1\}$ determine the dynamics. The objective is, $p_N(\mathbf{x}_N) = \|\mathbf{x}_N\|^2$ and $\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) = \|\mathbf{u}_k\|^2$.



Example 2: Reference Tracking of a Two-tank System

$$\begin{cases} \dot{x}_1 &= d_1(1-v)p - d_2\sqrt{x_1}, \\ \dot{x}_2 &= d_1 vp + d_2\sqrt{x_1} - d_2\sqrt{x_2}. \end{cases}$$

where $[x_1, x_2]^\top \in [0,1]^2$, control inputs $[p, v]^\top \in [0,1]^2$, $d_1 \sim \mathcal{U}[0.06,\ 0.1]$ and $d_2 \sim \mathcal{U}[0.01,\ 0.06]$. The objective is $p_N(\mathbf{x}_N) = \|\mathbf{x}_N - \mathbf{x}_{\text{ref}}(\boldsymbol{\xi})\|^2$ and $\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) = \|\mathbf{x}_k - \mathbf{x}_{\text{ref}}(\boldsymbol{\xi})\|^2 + \|\mathbf{u}_k\|^2$.

Example 2: Reference Tracking of a Two-tank System

$$\begin{cases} \dot{x}_1 &= d_1(1-v)p - d_2\sqrt{x_1}, \\ \dot{x}_2 &= d_1 vp + d_2\sqrt{x_1} - d_2\sqrt{x_2}. \end{cases}$$

where $[x_1, x_2]^\top \in [0,1]^2$, control inputs $[p, v]^\top \in [0,1]^2$, $d_1 \sim \mathcal{U}[0.06, 0.1]$ and $d_2 \sim \mathcal{U}[0.01, 0.06]$. The objective is $p_N(\mathbf{x}_N) = \|\mathbf{x}_N - \mathbf{x}_{\text{ref}}(\boldsymbol{\xi})\|^2$ and $\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) = \|\mathbf{x}_k - \mathbf{x}_{\text{ref}}(\boldsymbol{\xi})\|^2 + \|\mathbf{u}_k\|^2$.

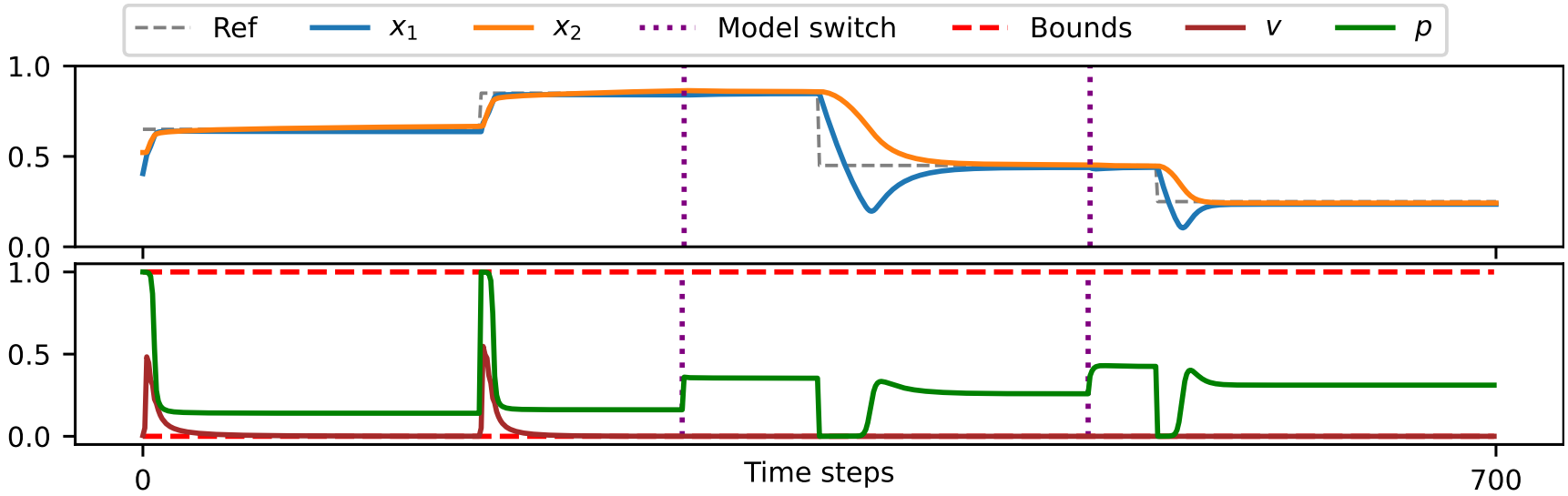


Figure: Two-tank system reference tracking under multiple system switches using FE-DPC.

Example 3: Reference Tracking of a Glycolytic Oscillator (GO)

$$
\begin{cases}
\dot{x}_1 &= J_0 - \frac{k_1 x_1 x_6}{1+(x_6/K_1)^4} + u, \text{(no control in below figure)} \\
\dot{x}_2 &= 2\frac{k_1 x_1 x_6}{1+(x_6/K_1)^4} - k_2 x_2 (N - x_5) - k_6 x_2 x_5, \\
\dot{x}_3 &= k_2 x_2 (N - x_5) - k_3 x_3 (A - x_6), \\
\dot{x}_4 &= k_3 x_3 (A - x_6) - k_4 x_4 x_5 - \kappa (x_4 - x_7), \\
\dot{x}_5 &= k_2 x_2 (N - x_5) - k_4 x_4 x_5 - k_6 x_2 x_5, \\
\dot{x}_6 &= -2\frac{k_1 x_1 x_6}{1+(x_6/K_1)^4} + 2k_3 x_3 (A - x_6) - k_5 x_6, \\
\dot{x}_7 &= \psi \kappa (x_4 - x_7) - k x_7.
\end{cases}
$$

where $k_1 \in \{80, 90, 100\}$ and $K_1 \in \{0.5, 0.75\}$.

Example 3: Reference Tracking of a Glycolytic Oscillator (GO)

$$
\begin{cases}
\dot{x}_1 &= J_0 - \frac{k_1 x_1 x_6}{1+(x_6/K_1)^4} + u, \text{(no control in below figure)} \\
\dot{x}_2 &= 2\frac{k_1 x_1 x_6}{1+(x_6/K_1)^4} - k_2 x_2 (N - x_5) - k_6 x_2 x_5, \\
\dot{x}_3 &= k_2 x_2 (N - x_5) - k_3 x_3 (A - x_6), \\
\dot{x}_4 &= k_3 x_3 (A - x_6) - k_4 x_4 x_5 - \kappa (x_4 - x_7), \\
\dot{x}_5 &= k_2 x_2 (N - x_5) - k_4 x_4 x_5 - k_6 x_2 x_5, \\
\dot{x}_6 &= -2\frac{k_1 x_1 x_6}{1+(x_6/K_1)^4} + 2k_3 x_3 (A - x_6) - k_5 x_6, \\
\dot{x}_7 &= \psi\kappa (x_4 - x_7) - k x_7.
\end{cases}
$$

where $k_1 \in \{80, 90, 100\}$ and $K_1 \in \{0.5, 0.75\}$.

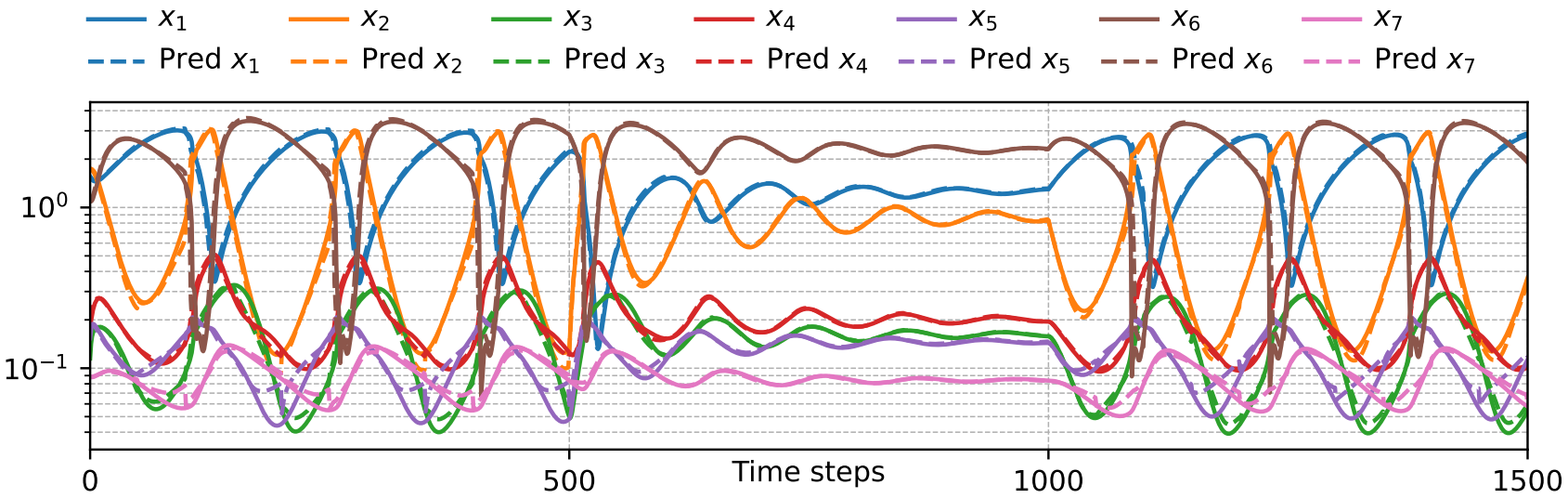


Figure: Parameters switch every 500 step, predictions calibrated against true states every 50 step.

Example 3: Reference Tracking of a Glycolytic Oscillator (G0)

Control objective is defined by $p_N(\mathbf{x}_N) = \|\mathbf{x}_{1,N} - \mathbf{x}_{1,\text{ref}}(\boldsymbol{\xi})\|^2$ and $\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) = \|\mathbf{x}_{1,k} - \mathbf{x}_{1,\text{ref}}(\boldsymbol{\xi})\|^2 + \|\mathbf{u}_k\|^2$.

Example 3: Reference Tracking of a Glycolytic Oscillator (G0)

Control objective is defined by $p_N(\mathbf{x}_N) = \|\mathbf{x}_{1,N} - \mathbf{x}_{1,\text{ref}}(\boldsymbol{\xi})\|^2$ and $\ell(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\xi}) = \|\mathbf{x}_{1,k} - \mathbf{x}_{1,\text{ref}}(\boldsymbol{\xi})\|^2 + \|\mathbf{u}_k\|^2$.

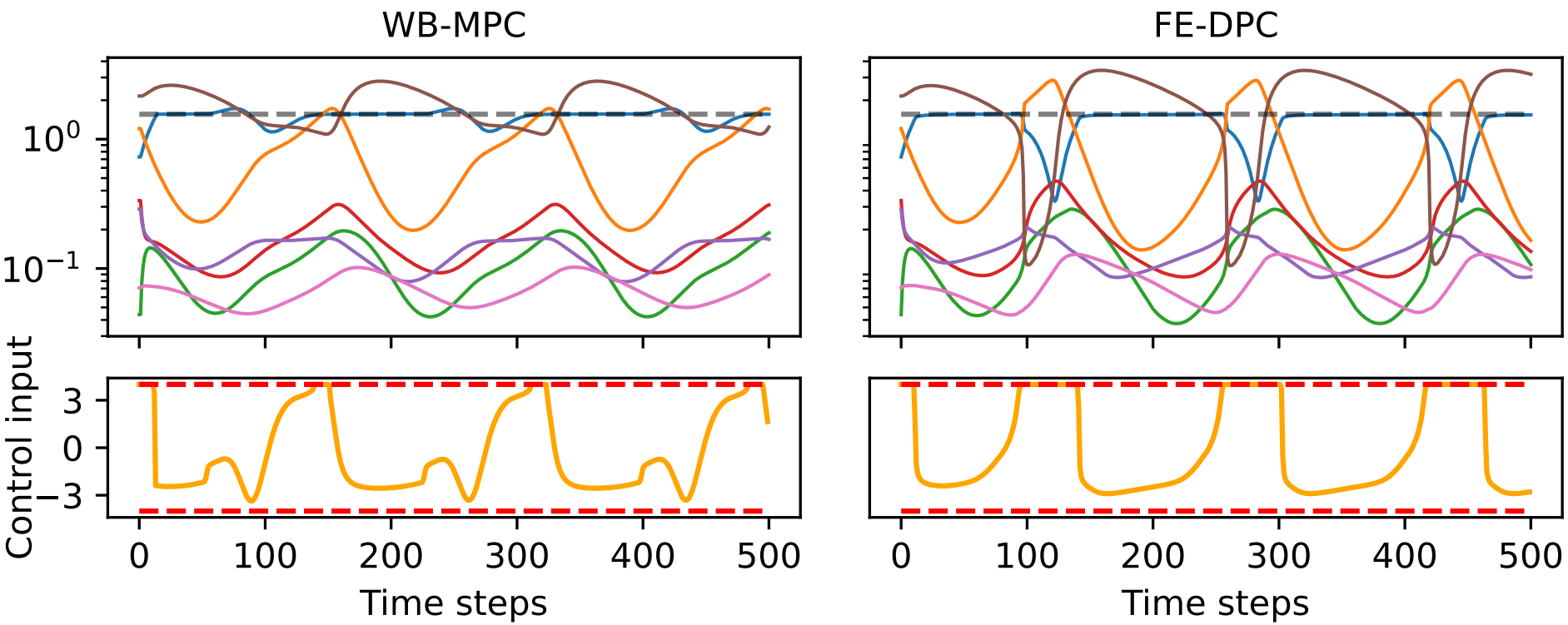


Figure: WB-MPC (left) and FE-DPC (right) based reference tracking of x_1 (blue) state.

Example 4: Controlling a Quadrotor

$$
\begin{cases}
\dot{p}_n = \cos\theta\cos\psi\, u \\
\quad + (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)\, v \\
\quad + (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\, w, \\
\dot{p}_e = \cos\theta\sin\psi\, u \\
\quad + (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi)\, v \\
\quad + (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\, w, \\
\dot{h} = \sin\theta\, u - \sin\phi\cos\theta\, v - \cos\phi\cos\theta\, w, \\
\dot{u} = rv - qw - g\sin\theta, \\
\dot{v} = pw - ru + g\cos\theta\sin\phi, \\
\dot{w} = qu - pv + g\cos\theta\cos\phi - (F/m), \\
\tau_\psi = 0.,
\end{cases}
\qquad
\begin{aligned}
F &= mg - 10(h - \alpha_1) + 3w, \\
\tau_\phi &= -(\phi - \alpha_2) - p, \\
\tau_\theta &= -(\theta - \alpha_3) - q, \\
\dot{\phi} &= p + \sin\phi\tan\theta\, q + \cos\phi\tan\theta\, r, \\
\dot{\theta} &= \cos\phi\, q - \sin\phi\, r, \\
\dot{\psi} &= (\sin\phi/\cos\theta)\, q + (\cos\phi/\cos\theta)\, r, \\
\dot{p} &= ((J_y - J_z)/J_x)\, qr + (1/J_x)\, \tau_\phi, \\
\dot{q} &= ((J_z - J_x)/J_y)\, pr + (1/J_y)\, \tau_\theta, \\
\dot{r} &= ((J_x - J_y)/J_z)\, pq + (1/J_z)\, \tau_\psi,
\end{aligned}
$$

$$
[\alpha_1, \alpha_2, \alpha_3]^\top \in [\boldsymbol{\alpha}_{\min},\, \boldsymbol{\alpha}_{\max}] = \begin{bmatrix} 0, & -0.524, & -0.524 \\ 1.5, & 0.524, & 0.524 \end{bmatrix}^\top \subset \mathbb{R}^3,
$$

$$
m \sim \mathcal{U}[1.2,\, 1.6], \quad J_x = J_y \sim \mathcal{U}[0.050,\, 0.058], \quad J_z \sim \mathcal{U}[0.090,\, 0.110].
$$

Objective: maintain an altitude of 0.4m with zero linear and angular velocities.

Example 4: Controlling a Quadrotor

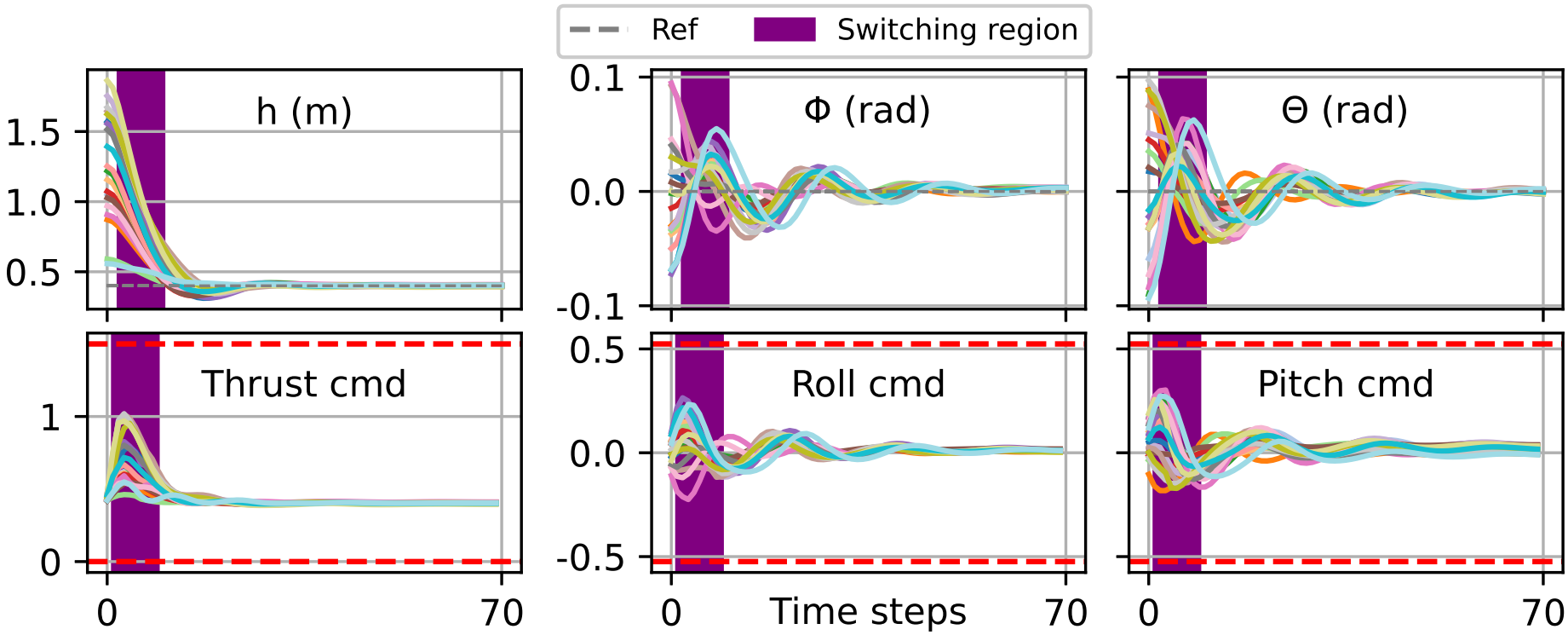


Figure: 20 quadrotor models with distinct dynamics parameterization are randomly initialized within state bounds. Each experiences a random dynamics switch between 2–20 s.

Inference Times

The main trade-off

Algorithm	Metric	Van der Pol	Two Tank	GO (7D)	Quad (12D)
FE-DPC	Error (MSE)	0.0027	0.0085	0.1803	0.0220
	Time (s)	0.53	1.13	5.89	1.93
WB-MPC*	Error (MSE)	0.0027	0.0042	0.0323	0.0242
	Time (s)	1.21	6.75	136.07	155.85

Table: Comparison of error (MSE) and computation time (s) for each benchmark.

* Assumed no plant-model mismatch.

Conclusion

We introduced FE-DPC, a framework for zero-shot adaptive predictive control:

- FE-NODE to identify unknown dynamics online from a few data points.
- DPC to learn a parametric policy offline conditioned on identified dynamics.
- Instantly adapts to unknown dynamics without retraining.
- Examples shown for high-dimensional (12D) and stiff (7D) systems.
- Achieves accuracy competitive with MPC at fraction of computational cost.

Conclusion

We introduced FE-DPC, a framework for zero-shot adaptive predictive control:

- FE-NODE to identify unknown dynamics online from a few data points.
- DPC to learn a parametric policy offline conditioned on identified dynamics.
- Instantly adapts to unknown dynamics without retraining.
- Examples shown for high-dimensional (12D) and stiff (7D) systems.
- Achieves accuracy competitive with MPC at fraction of computational cost.

Some remarks and future work:

- Complimentary to model-based approaches, potential benefits by combining e.g. parameter estimation, preconditioning, uncertain systems etc.
- Closed-loop guarantees.

References I

Drgoňa, J. et al. (2024). "Learning constrained parametric differentiable predictive control policies with guarantees". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54.6, pp. 3596–3607.

Ingebrand, T. et al. (2024). "Zero-shot transfer of neural ODEs". In: *Advances in Neural Information Processing Systems* 37, pp. 67604–67626.

Iqbal, H. et al. (2025). "Zero-Shot Function Encoder-Based Differentiable Predictive Control". In: *arXiv preprint arXiv:2511.05757*.

Low, S. A. et al. (2025). "Function Spaces Without Kernels: Learning Compact Hilbert Space Representations". In: *arXiv preprint arXiv:2509.20605*.