MACHINE LEARNING
Engineering

Part of the
MACHINE LEARNING
Series

PAPER • OPEN ACCESS

Policy learning for real-time control of digital twins in additive manufacturing via differentiable predictive control using multi-step time-series deep neural networks

To cite this article: Seul Lee *et al* 2026 *Mach. Learn.: Eng.* **2** 025017

View the article online for updates and enhancements.

You may also like

- Methods of the theory of singularities in the phenomenology of phase transitions
E I Kut'in, V L Lorman and S V Pavlov
- Analytic auto-differentiable CDM cosmography
Konstantin Karchev
- DiffGLE: Differentiable coarse-grained dynamics using generalized Langevin equation
Jinu Jeong and Ishan Nadkarni

This content was downloaded from IP address 207.229.146.37 on 23/11/2026 at 13:49

MACHINE LEARNING
Engineering

Check for updates

PAPER

OPEN ACCESS

RECEIVED
15 May 2026

REVISED
13 July 2026

ACCEPTED FOR PUBLICATION
16 September 2026

PUBLISHED
23 November 2026

Original content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

Policy learning for real-time control of digital twins in additive manufacturing via differentiable predictive control using multi-step time-series deep neural networks

Seul Lee[1], **Yi-Ping Chen**[1], **Ying-Kuan Tsai**[2], **Vispi Karkaria**[1] and **Wei Chen**[1,*]

[1] Department of Mechanical Engineering, Northwestern University, 2145 Sheridan Rd., Evanston, IL 60208, United States of America
[2] Ingram School of Engineering, Texas State University, 601 University Dr, San Marcos, Texas 78666, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: weichen@northwestern.edu

Keywords: digital twin, policy learning, differentiable predictive control, real-time decision-making, data augmentation.

Abstract

Advanced manufacturing processes such as directed energy deposition (DED) additive manufacturing involve strongly coupled dynamics, process variability, and high-frequency decision-making, making real-time implementation of model predictive control (MPC) computationally challenging. This study develops a model-based offline policy learning framework based on differentiable predictive control (DPC) to enable real-time high-frequency control of manufacturing processes. The proposed framework integrates a one-shot multi-step predictor based on the Time-series Dense Encoder and a parameterized control policy within a fully differentiable structure, allowing the entire framework to be optimized end-to-end with respect to an MPC-inspired loss without requiring online optimization or supervised MPC data. In addition, we introduce a model rollout-based data augmentation strategy to address the limitations of open-loop control data, which often fails to capture constraint-aware behavior. Through iterative model rollouts during policy training, the proposed strategy progressively augments the training data with new trajectories, exposing the policy to constraint-aware behavior. The proposed framework is evaluated on a digital twin of DED, where the learned control policy tracks a reference melt pool temperature trajectory while satisfying melt pool depth constraints and maintaining smooth laser power regulation. Results demonstrate that the learned policy achieves tracking performance comparable to MPC while significantly reducing computational cost, with an inference time on the order of 10^{-4} s per control step. Furthermore, the proposed method improves constraint satisfaction compared to MPC while avoiding the excessive conservativeness of robust MPC. These results highlight the potential of DPC as a practical alternative for real-time control in advanced manufacturing systems that require high-frequency decision-making.

Nomenclature

$(\cdot)_f$	Trajectory segment over the future prediction horizon
$(\cdot)_p$	Trajectory segment over the past window
ξ	Policy model input
ℓ_{DPC}	DPC stage loss
$\hat{(\cdot)}$	Predicted quantity
$\hat{\mathbf{x}}_k$	Predicted system state at time step k
D	Known external-variable trajectory
d_k	Known external variable at time step k
R	Reference trajectory over the prediction horizon

© 2026 The Author(s). Published by IOP Publishing Ltd

r_k	Reference signal at time step k
U	Control input trajectory
u_k	Control input at time step k
X	System state trajectory
x_k	System state at time step k
$\mathcal{B}$	Mini-batch of samples used in model-rollout augmentation
$\mathcal{D}_u, \mathcal{D}_d, \mathcal{D}_x$	Generated datasets of control input sequences, known external-variable sequences, and system state sequences
$\mathcal{D}_{\text{aug}}$	Augmented dataset generated by model rollout
$\mathcal{D}_{\text{train}}$	Training dataset
ϕ	Parameters of the surrogate model
$\pi_\theta(\cdot)$	Neural policy model
θ	Parameters of the policy model
$\tilde{(\cdot)}$	Quantity generated by model rollout
$\underline{(\cdot)}$, $\overline{(\cdot)}$	Lower and upper bounds
Ξ_{ctrl}	Set of admissible policy model inputs
d_x, d_y	Nearest boundary distances from the current laser position at time step k along the x- and y-axis
$f(\cdot)$	Nonlinear system dynamics
$f_\phi(\cdot)$	Neural surrogate model
$f_\phi{}^N(\cdot)$	N-step recursive rollout of the surrogate dynamics model
K	Epoch interval between successive model-rollout augmentations
L_Q	Aggregate quantile loss over states and prediction horizon
L_q	Pointwise quantile loss
L_{DPC}	DPC training loss
L_{MPC}	MPC objective function
M	Number of samples used for model-rollout augmentation
N	Prediction horizon length
Q_r, Q_{du},	Weights for reference tracking error, control input variation,
Q_c, Q_u	state constraint violation, and input constraint violation
T	Total number of model-rollout augmentation iterations
w	Window length
z	Position of the laser nozzle along the z-axis
AM	Additive manufacturing
DED	Directed energy deposition
DPC	Differentiable predictive control
MPC	Model predictive control
TiDE	Time-series Dense Encoder

1. Introduction

Modern manufacturing systems are increasingly characterized by high levels of sensing, connectivity, and process variability, requiring control strategies that can operate reliably under dynamic conditions [1]. Digital Twin frameworks enable such functionality by integrating real-time data from physical systems with virtual models for monitoring, prediction, and decision-making [2–8]. In advanced manufacturing processes such as DED AM, process conditions directly govern melt pool dynamics and thermal history, which in turn determine microstructure evolution and ultimately material performance [9–11]. This strong process-structure coupling necessitates precise regulation of melt pool behavior. Thus, feedback

control has been shown to influence grain structure, improve tensile properties, and reduce microstructural variability [12–14], motivating the use of predictive, optimization, and control capabilities enabled by Digital Twin frameworks.

MPC has been widely recognized as a suitable control framework for complex manufacturing systems due to its ability to explicitly handle multivariable interactions, system constraints, and dynamic process behavior through iterative optimization [15, 16]. This is particularly critical in AM processes, where process variables-such as laser power, scan speed, and material feed rate-must be optimized under thermal and physical constraints to regulate transient thermal behavior and ensure consistent part quality. Moreover, the prediction-based decision-making structure of MPC aligns naturally with the Digital Twin paradigm, where a virtual model is used to simulate future system behavior and proactively guide decisions in real-time applications.

Despite these advantages, as shown in figure 1(a), the requirement to solve an optimization problem online at every time step introduces significant computational overhead, limiting the applicability of MPC in systems with high-dimensional control inputs or scenarios with high control frequency, such as AM [17, 18]. To address this limitation, prior work has explored explicit MPC, where the optimal control law is computed offline as a parametric function of the system state, along with a range of complexity-reduction techniques that aim to simplify this representation for efficient online evaluation [19–24]. While these approaches significantly reduce online computation, their scalability remains limited due to the growth in complexity with system dimension, prediction horizon, and constraints.

To overcome the limitations of explicit MPC, particularly its exponential growth in complexity and memory requirements, neural network (NN)-based approximate MPC approaches have been proposed, which replace the online optimization with a learned parametric mapping from system states to control actions [25–28], as shown in figure 1(b). Among these approaches, a common formulation is to approximate the MPC policy via supervised learning, where a NN is trained to mimic the control actions generated by an MPC controller using state-action trajectory data [29–33]. While such approaches enable efficient policy evaluation at deployment by replacing online optimization with fast inference, they shift much of the computational burden to the offline data-generation stage, since large state-action datasets must be obtained by repeatedly solving the original MPC problem. This requirement can become computationally expensive, or even intractable, for nonlinear, high-dimensional, or long-horizon MPC problems, and also presupposes access to a fully implemented MPC, which itself may be computationally prohibitive in such settings.

Different from leveraging supervised learning from collecting samples from MPC trajectories, DPC has recently emerged as a model-based policy learning framework that trains a neural control policy directly through a differentiable prediction model [34, 35]. Unlike conventional approximate MPC approaches that first solve MPC problems to generate training data, DPC learns the policy by embedding a differentiable system model directly into the training loop. Specifically, DPC formulates policy learning as an optimization problem over predicted system behavior across a finite horizon. By conditioning on the current state or state history, along with the desired reference trajectory and input/output constraints, the policy model first generates a candidate sequence of future control actions. These control actions are then applied to a pre-trained differentiable system model to predict the resulting system response. The predicted trajectory is subsequently evaluated using an MPC-inspired loss function that penalizes tracking error, constraint violations, and control effort. Because the system model remains fixed but differentiable, gradients of this loss can be propagated through the predicted system response to update only the policy model. As a result, the policy is optimized directly with respect to its closed-loop performance over the prediction horizon, without requiring state-action training data generated from an external MPC controller. In this setting, the supervised learning problem is effectively recast as a sampling problem, where training samples are generated by specifying initial conditions, reference trajectories, and constraint configurations.

Prior work on DPC has primarily focused on improving individual components of the framework. This includes the use of structured or Koopman-based dynamics models to improve computational efficiency and scalability [36], the incorporation of control barrier functions for safety guarantees [37], and the development of data-driven predictive safety filters to enforce safe operation [38]. Following this formulation, DPC has been applied to a range of domains, including building energy systems [34], power system optimization [36], robotics [38], and large-scale infrastructure control [39]. These studies demonstrate that DPC can provide a practical alternative to conventional MPC by replacing online optimization with efficient policy inference. However, applications of DPC in manufacturing systems remain limited, particularly in advanced manufacturing processes such as AM, where system dynamics are strongly coupled with process conditions and subject to variability [11].

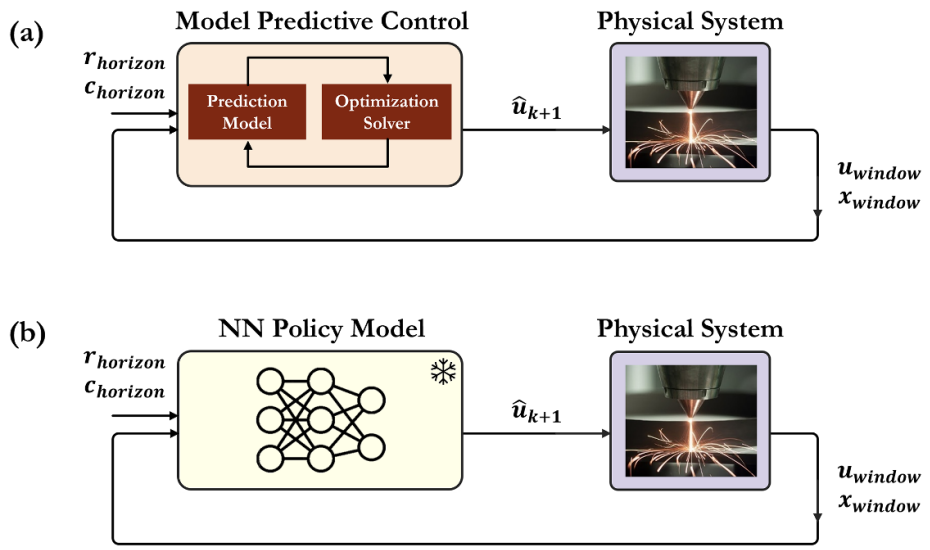


Figure 1. Comparison of MPC and NN-based approximate MPC. (a) MPC with online optimization. (b) Approximate MPC using a neural network (NN) policy. Here, x_{window} and u_{window} denote the state and control trajectories over the window, respectively, while r_{horizon} and c_{horizon} denote the reference trajectory and constraint bounds over the prediction horizon. The control action $\hat{u}_{k+1}$ is applied to the system.

In systems with strongly coupled and variable dynamics, the generation of training data becomes particularly important. While DPC enables flexible generation of training samples through arbitrarily specified objective trajectories, the sampling task is not a trivial problem because time-series trajectories must preserve temporal dependencies and cross-variable relationships constrained by the underlying system dynamics [40]. Poorly generated samples without realistic physical meaning can introduce critical issues: (1) The generated trajectories that fall outside the validity region of the learned system model may not benefit the training of the policy model, and (2) the generated trajectories and constraint sets that are inconsistent with the system's achievable behavior over the prediction horizon will discourage the learned policy from handling the constraints. For effective policy learning, training data must therefore be both representative of realistic operating conditions and consistent with the validity region of the learned system model. However, existing DPC works largely rely on heuristic or parameter-based sampling strategies for generating training data that only focus on the spatial domain but ignore the spatiotemporal coupling nature of the samples. To the best of our knowledge, a systematic approach for constructing physically consistent and representative objective trajectories remains underexplored.

Beyond sampling, it is critical for the learned system model to accurately and sufficiently represent system behavior, as model-based policy learning relies on predicted state trajectories over the prediction horizon. However, existing DPC approaches typically rely on one-step prediction models, where multi-step trajectories are generated by recursively applying the model over the prediction horizon [34–36]. This recursive formulation introduces error accumulation over the prediction horizon [41] and fails to capture the joint temporal dependencies of multi-step trajectories. This limitation is particularly critical in AM, where the system dynamics exhibit significant temporal dependency due to the influence of thermal accumulation and melt pool evolution across time. As a result, predicted trajectories that fail to capture temporal correlations across the horizon can limit the performance of the learned policy model, particularly in partially observable settings.

Research objective. To address the aforementioned challenges, this work presents a model-based policy learning framework using DPC for the Digital Twin of DED, as illustrated in figure 2. First, to address the challenge of capturing strongly coupled and temporally-dependent dynamics in DED processes, we incorporate a multi-step prediction model through the TiDE [42], enabling one-shot prediction over the full prediction horizon within a single forward pass. Second, we address training data construction by introducing a model-based data augmentation strategy that systematically generates trajectories consistent with system dynamics and feasible operating conditions. The proposed framework is evaluated on both an illustrative example and a case study in DED metal AM. The main contributions are summarized as follows:

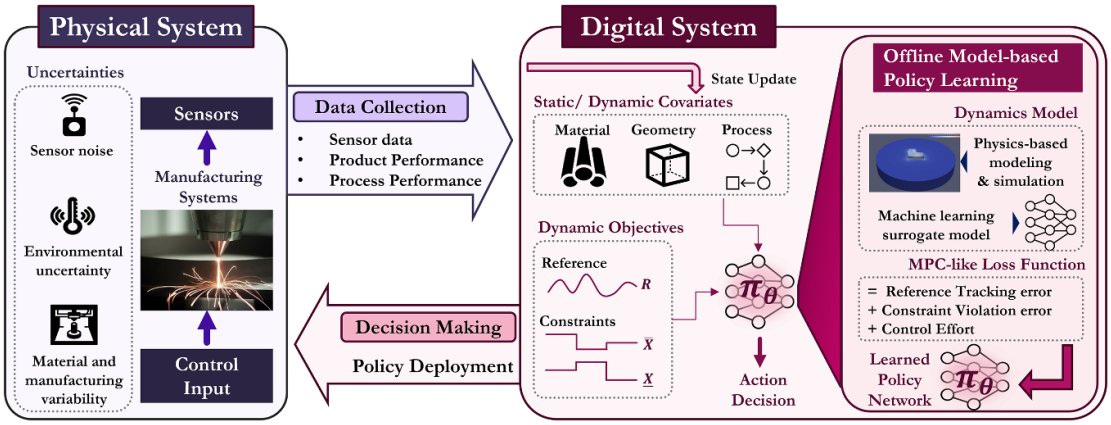


Figure 2. Overview of the proposed policy-based digital twin framework for manufacturing systems. From the physical system, data are collected and transferred to the digital system for state updates. In the digital system, both static and dynamic covariates together with dynamic objectives are provided as inputs to a policy network, which is trained offline using a learned surrogate model and an MPC-like loss function. The trained policy network is then used for real-time decision inference, and the resulting control decisions are subsequently applied back to the physical manufacturing system, enabling a bi-directional interaction within the digital twin framework. Here, π_θ denotes the policy model parameterized by θ, $\mathbf{R}$ denotes the reference trajectory over the prediction horizon, and $\underline{\mathbf{X}}$ and $\overline{\mathbf{X}}$ denote the lower and upper bounds of the system states, respectively.

- We extend the existing DPC framework by incorporating a multi-step-ahead prediction model, enabling one-shot prediction over the full prediction horizon.
- We propose a model rollout-based data augmentation strategy that constructs physically consistent training trajectories, addressing the limitations of heuristic objective sampling in existing DPC implementations.
- We demonstrate the applicability of the proposed framework to a metal AM control problem, highlighting its effectiveness in handling system variability in data-driven control settings.

The rest of this paper is organized as follows. Section 2 introduces the technical background on MPC, DPC, the TiDE architecture, and the thermal simulation model. Section 3 presents the proposed policy learning framework, including surrogate modeling, DPC-based policy learning, and the model rollout-based data augmentation strategy. Section 4 demonstrates the proposed framework using an illustrative linear system example, while section 5 evaluates its performance on a DED AM control problem. Finally, conclusions and future research directions are provided in section 6.

2. Technical background

2.1. MPC

MPC utilizes an explicit dynamic model of a system to predict its future behavior and compute optimal control actions over a finite prediction horizon. Once the optimal control sequence is obtained, the first control action is applied to the system, and the optimization is repeated at the next time step using updated state information, following a receding horizon strategy. The MPC problem at time step k can be written as

$$\min_{\mathbf{u}_{k:k+N-1}} \quad \sum_{i=1}^{N} \left(Q_r \|\hat{\mathbf{x}}_{k+i} - \mathbf{r}_{k+i}\|_2^2 + Q_{du} \|\mathbf{u}_{k+i} - \mathbf{u}_{k+i-1}\|_2^2 \right) \tag{1a}$$

$$\text{s.t.} \quad \hat{\mathbf{x}}_{k+i} = f(\mathbf{x}_{k+i-1}, \mathbf{u}_{k+i-1}), \quad i \in \mathbb{N}_1^N \tag{1b}$$

$$\mathbf{x}_{k+i} \in \mathcal{X} \tag{1c}$$

$$\mathbf{u}_{k+i} \in \mathcal{U} \tag{1d}$$

where N denotes the prediction horizon length, $\mathbf{x}_{k+i} \in \mathbb{R}^{n_x}$ denotes the system state, $\hat{\mathbf{x}}_{k+i} \in \mathbb{R}^{n_x}$ denotes the predicted state, $\mathbf{r}_{k+i} \in \mathbb{R}^{n_r}$ denotes the reference trajectory, and $\mathbf{u}_{k+i} \in \mathbb{R}^{n_u}$ denotes the control input, all at time step $k+i$, where n_x, n_r, and n_u denote the dimensions of the state, reference, and input vectors, respectively. The sets $\mathcal{X}$ and $\mathcal{U}$ denote the admissible state and input constraint sets, respectively.

The system dynamics in equation (1*b*) is described by the function $f(\cdot)$, which represents the state evolution of the system. The objective function in equation (1*a*) defines a cost over the prediction horizon which can be flexibly specified to reflect system-specific performance requirements; a common choice, particularly for tracking problems, is a quadratic cost that penalizes deviations of the predicted state from the reference trajectory and variations in control inputs. The constraints in equations (1*c*) and (1*d*) define admissible regions for states and inputs, respectively. The weighting coefficients Q_r and Q_{du} in (1*a*) determine the relative importance of reference tracking and control effort.

Depending on the system dynamics and problem structure, the computational method used to solve an MPC problem varies. For unconstrained linear systems with quadratic cost, the resulting finite-horizon optimal control problem is equivalent to the standard linear quadratic regulator and can be solved via Riccati equation [15]. When constraints are introduced, the MPC problem can be formulated as a convex quadratic program (QP), for which efficient solvers such as active-set, interior-point, or first-order methods are commonly used [43–47]. Nonlinear MPC generally leads to a nonconvex nonlinear program, which is typically solved using iterative gradient-based methods such as sequential QP or interior-point methods [48, 49]. For such cases, solving optimization problems online at every time step can be computationally demanding for real-time applications as the number of function evaluations required is often unbounded, particularly in high-dimensional systems, long-horizon settings, or at high decision frequencies.

2.2. DPC

In this work, we use DPC to replace the online optimization in MPC with a neural policy model, which is learned from a fully differentiable policy learning architecture. Figure 3 illustrates how the policy model is learned using a pre-trained NN-based system model, hereafter referred to as the surrogate model. The DPC problem is formulated, following and modifying the formulation in [35], as:

$$\min_{\theta} \sum_{i=1}^{N} \ell_{\mathrm{DPC}}\left(\hat{\mathbf{x}}_i, \mathbf{u}_i, \boldsymbol{\xi}_i\right) \tag{2a}$$

$$\text{s.t.} \quad \hat{\mathbf{x}}_i = f_\phi\left(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}\right), \quad i \in \mathbb{N}_1^N \tag{2b}$$

$$\mathbf{u}_i = \pi_\theta\left(\boldsymbol{\xi}_i\right) \tag{2c}$$

$$\mathbf{x}_i \in \mathcal{X} \tag{2d}$$

$$\mathbf{u}_i \in \mathcal{U} \tag{2e}$$

$$\boldsymbol{\xi}_i \in \Xi^{\mathrm{ctrl}}. \tag{2f}$$

Here, $\pi_\theta(\cdot)$ denotes the policy model parameterized by θ, and $f_\phi(\cdot)$ denotes the one-step neural surrogate dynamics model parameterized by ϕ. The policy input $\boldsymbol{\xi}_i \in \Xi^{\mathrm{ctrl}}$ represents the decision-relevant information provided to the controller at prediction step i. As illustrated in figure 3, this information may include past state trajectories, future reference trajectories, and state constraints over the prediction horizon; we therefore omit the prediction-step subscript and define the policy input compactly as

$$\boldsymbol{\xi} = \begin{bmatrix} \mathbf{X}_{\mathrm{p}}^\top & \mathbf{R}^\top & \underline{\mathbf{X}}^\top & \overline{\mathbf{X}}^\top \end{bmatrix}, \tag{3}$$

where

$$\begin{aligned} \mathbf{X}_{\mathrm{p}} &= [\mathbf{x}_{1-w}, \ldots, \mathbf{x}_0] \in \mathbb{R}^{n_x \times w}, \quad \mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_N] \in \mathbb{R}^{n_r \times N}, \\ \underline{\mathbf{X}} &= [\underline{\mathbf{x}}_1, \ldots, \underline{\mathbf{x}}_N] \in \mathbb{R}^{n_x \times N}, \quad \overline{\mathbf{X}} = [\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_N] \in \mathbb{R}^{n_x \times N}. \end{aligned} \tag{4}$$

Here, $\mathbf{X}_{\mathrm{p}}$ denotes the past state trajectory segment over the window length w, $\mathbf{R}$ denotes the reference trajectory over the prediction horizon, and $\underline{\mathbf{X}}$ and $\overline{\mathbf{X}}$ denote the lower and upper state bounds, respectively. These components correspond to the policy inputs illustrated in figure 3.

Given the aggregated policy input $\boldsymbol{\xi}$, the policy model generates a future control sequence, which is subsequently propagated through the surrogate dynamics model to predict future system trajectories.

$$\mathbf{U}_{\mathrm{f}} = \pi_\theta\left(\boldsymbol{\xi}\right) \tag{5a}$$

$$\hat{\mathbf{X}}_{\mathrm{f}} = f_\phi^N\left(\mathbf{X}_{\mathrm{p}}, \mathbf{U}_{\mathrm{f}}\right) \tag{5b}$$

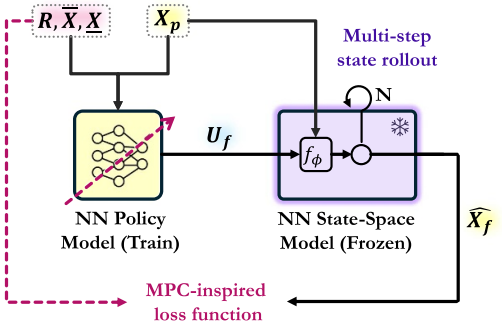


Figure 3. Illustration of the standard Differentiable Predictive Control (DPC) architecture (adapted from [35]). The policy model receives the policy input consisting of the reference trajectory $\mathbf{R}$, the lower and upper state bounds $\underline{\mathbf{X}}$ and $\overline{\mathbf{X}}$, and the past state trajectory segment $\mathbf{X}_\mathrm{p}$. The policy network then generates the future control input trajectory $\mathbf{U}_\mathrm{f}$, which is provided to the neural surrogate dynamics model f_ϕ. Using recursive rollout over the prediction horizon, the surrogate model predicts the future system-state trajectory $\hat{\mathbf{X}}_\mathrm{f}$. The predicted trajectory is subsequently evaluated using an MPC-inspired loss function based on reference tracking and constraint satisfaction objectives, and the resulting loss is backpropagated through both the surrogate model and the policy network to optimize the policy parameters.

where

$$\mathbf{U}_\mathrm{f} = [\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}] \in \mathbb{R}^{n_u \times N}, \quad \hat{\mathbf{X}}_\mathrm{f} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N] \in \mathbb{R}^{n_x \times N}. \tag{6}$$

Here, $\mathbf{U}_\mathrm{f}$ denotes the future control trajectory generated by the policy model, and $\hat{\mathbf{X}}_\mathrm{f}$ denotes the predicted future state trajectory. The function $f_\phi^N(\cdot)$ denotes the recursive application of $f_\phi(\cdot)$ defined in (2*b*) over the prediction horizon. As illustrated in the right-hand block of figure 3, the surrogate model performs a multi-step rollout by iteratively propagating the predicted states using the generated control sequence. The predicted trajectories are subsequently evaluated using an MPC-inspired loss function, defined as follows:

$$\begin{aligned}\mathcal{L}_\mathrm{DPC}(\theta) = \frac{1}{nN}\sum_{j=1}^{n}\sum_{i=1}^{N}\Big(& Q_r\|\mathbf{r}_i^j - \hat{\mathbf{x}}_i^j\|_2^2 + Q_{du}\|\mathbf{u}_i^j - \mathbf{u}_{i-1}^j\|_2^2 \\ & + Q_c\|p\left(\underline{\mathbf{x}}_i^j, \hat{\mathbf{x}}_i^j\right)\|_2^2 + Q_c\|p\left(\hat{\mathbf{x}}_i^j, \overline{\mathbf{x}}_i^j\right)\|_2^2 + Q_u\|p\left(\underline{\mathbf{u}}, \mathbf{u}_{i-1}^j\right)\|_2^2 + Q_u\|p\left(\mathbf{u}_{i-1}^j, \overline{\mathbf{u}}\right)\|_2^2\Big)\end{aligned} \tag{7}$$

where i indexes the prediction horizon steps, j indexes the sampled trajectories, and n denotes the batch size. The weighting coefficients Q_r, Q_{du}, Q_c, and Q_u balance the relative contributions of the corresponding loss terms: the reference tracking error, control effort, state constraint violation, and input constraint violation terms, respectively. Here, $\underline{\mathbf{u}}$ and $\overline{\mathbf{u}}$ denote the lower and upper bounds on the control inputs, respectively, which are fixed parameters specified externally. The function $p(\mathbf{a}, \mathbf{b}) = \max(\mathbf{0}, \mathbf{a} - \mathbf{b})$ defines the soft constraint violation. This formulation closely follows the structure of the MPC problem, where the tracking and control smoothing terms correspond to the cost, and the penalty terms approximate the output and input constraints. As illustrated by the dashed feedback path in figure 3, the resulting loss is backpropagated through both the surrogate dynamics model and the policy network, enabling end-to-end policy optimization. As a result, the policy model learns a mapping from control-relevant features to control sequences, enabling efficient inference while retaining the ability to handle nonlinear dynamics, time-varying references, and inequality constraints, thereby replacing the need for online optimization in traditional MPC.

2.3. TiDE

In this work, we employ the TiDE, an MLP-based architecture for one-shot long-horizon forecasting [42], as the surrogate model for the discrete-time dynamical system. The selection of a suitable time-series deep neural network for DPC is guided by two primary considerations.

First, the model must effectively capture long-horizon spatio-temporal dependencies required for multi-step prediction of dynamical systems. This requirement is particularly critical in the considered AM setting, where system behavior is strongly history-dependent and influenced by spatially distributed exogenous variables. In such cases, the full Markovian state is either not observable through available sensing or too high-dimensional to be used for real-time control. A deployable controller must instead

operate on a reduced, observable state representation, under which the dynamics become effectively non-Markovian and the system is only partially observed. Consequently, future states cannot be accurately inferred from the reduced state alone. Prediction instead depends on a window of past states and inputs, together with past and future exogenous information, which compensates for the partial observability. To address this, we adopt a direct input–output modeling formulation, in which all decision-relevant information is aggregated into a single input representation. Therefore, the chosen architecture must be capable of modeling complex temporal dependencies over extended horizons.

Second, the inference speed should be sufficiently fast. Although DPC performs policy learning offline and is therefore less sensitive to inference latency, faster inference directly reduces training time and enables scalability to larger datasets. TiDE satisfies these requirements through its architecture, which jointly processes past targets, covariates, and control inputs over the entire time window via parallel dense layers. Residual connections enable stable learning of deep temporal mappings, allowing the model to capture complex input–output dependencies over the prediction horizon. In addition, the use of fully connected layers enables parallel processing of the entire input sequence, resulting in linear computational complexity with respect to the sequence length and faster inference compared to recurrent or attention-based models. On long-term forecasting benchmarks, TiDE achieves accuracy comparable to or better than Transformer-based baselines with 5× faster inference and more than 10× faster training [42]. A further advantage of TiDE is its full differentiability. All components support automatic differentiation via backpropagation, implemented using `PyTorch`, enabling end-to-end differentiability required for gradient-based policy optimization within the DPC framework.

As illustrated in figure 4, TiDE takes as input the past target sequence, along with past and known future covariates, and directly predicts the state trajectory over the prediction horizon. In this work, the term 'covariates' is used to collectively refer to both known external variables and control inputs. The TiDE formulation is given by

$$\hat{\mathbf{X}}_{f(k)} = \mathrm{TiDE}_{\phi}\left(\mathbf{X}_{p(k)}, \mathbf{D}_{p(k)}, \mathbf{D}_{f(k)}, \mathbf{U}_{p(k)}, \mathbf{U}_{f(k)}\right), \tag{8}$$

where

$$\begin{aligned} \mathbf{X}_{p(k)} &= [\mathbf{x}_{k-w+1}, \ldots, \mathbf{x}_k] \in \mathbb{R}^{n_x \times w}, \quad \hat{\mathbf{X}}_{f(k)} = [\hat{\mathbf{x}}_{k+1}, \ldots, \hat{\mathbf{x}}_{k+N}] \in \mathbb{R}^{n_x \times N}, \\ \mathbf{D}_{p(k)} &= [\mathbf{d}_{k-w}, \ldots, \mathbf{d}_{k-1}] \in \mathbb{R}^{n_d \times w}, \quad \mathbf{D}_{f(k)} = [\mathbf{d}_k, \ldots, \mathbf{d}_{k+N-1}] \in \mathbb{R}^{n_d \times N}, \\ \mathbf{U}_{p(k)} &= [\mathbf{u}_{k-w}, \ldots, \mathbf{u}_{k-1}] \in \mathbb{R}^{n_u \times w}, \quad \mathbf{U}_{f(k)} = [\mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1}] \in \mathbb{R}^{n_u \times N}. \end{aligned} \tag{9}$$

Here, $\mathbf{X}_{p(k)}$, $\mathbf{D}_{p(k)}$, and $\mathbf{U}_{p(k)}$ represent past observations relative to time step k over a window, while $\mathbf{D}_{f(k)}$ and $\mathbf{U}_{f(k)}$ denote known future inputs relative to time step k over the prediction horizon. The variable $\mathbf{d} \in \mathbb{R}^{n_d}$ denotes known external variables, which are factors that influence the system but are not directly controllable, such as geometry information that can be prescribed in AM applications or operating conditions. The model output $\hat{\mathbf{X}}_{f(k)}$ corresponds to the predicted future state trajectory. When the time index (k) is omitted, the variables are to be interpreted as individual samples re-indexed with a local time origin $(k = 0)$, retaining only the relative temporal structure within each segment; this interpretation applies whenever index-free notation is used hereafter.

This formulation aligns with the structure of dynamical systems where future states depend not only on historical states but also on external inputs and control actions. As illustrated on the right side of figure 4, the TiDE architecture separates target variables and covariates, explicitly incorporating both system evolution and exogenous influences into the prediction process. This design enables the model to capture complex temporal interactions while maintaining computational efficiency, making it well-suited for multi-step prediction in dynamical systems.

2.4. Thermal simulation model

This work uses an explicit finite element analysis (FEA) code for part-scale thermomechanical simulation of the DED process, named GAMMA, developed by Liao *et al* [51] the code is implemented in Python and further accelerated using CuPy 9.0.0 [52].

The heat transfer process is modeled based on Fourier's law of heat conduction, expressed as the following governing equation:

$$\rho C_{\mathrm{p}}(T)\frac{\partial T}{\partial t} = k(T)\nabla^2 T + q_{\mathrm{laser}} + q_{\mathrm{conv}} + q_{\mathrm{rad}}, \tag{10}$$

where T denotes the temperature (K), ρ denotes the material density (g mm^{-3}), $C_{\mathrm{p}}(T)$ denotes the temperature-dependent specific heat capacity (J g^{-1} K^{-1}), and $k(T)$ denotes the thermal conductivity

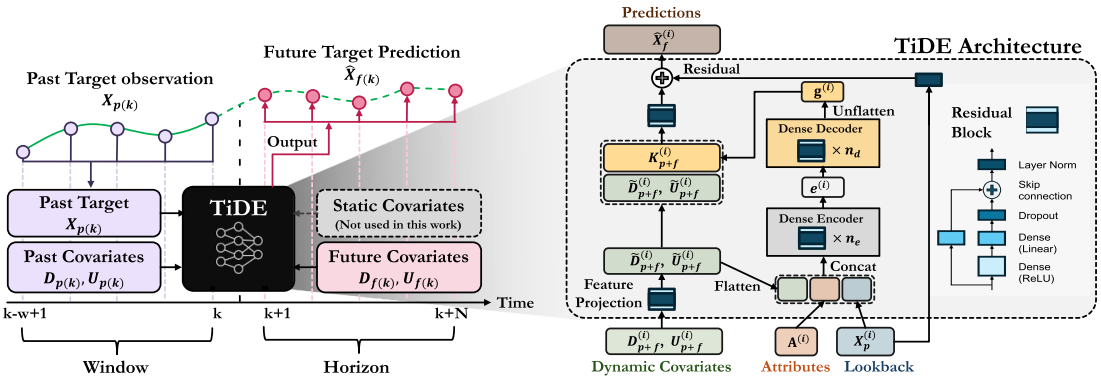


Figure 4. Data structure of the input and output of TiDE (left, adapted from [50]) and network architecture of TiDE (right, adapted from [42]). The TiDE model receives the past target trajectory segment $\mathbf{X}_{p(k)}$ together with the past and future known external-variable trajectory segments $\mathbf{D}_{p(k)}$ and $\mathbf{D}_{f(k)}$, and the past and future control input trajectory segments $\mathbf{U}_{p(k)}$ and $\mathbf{U}_{f(k)}$, defined relative to the current time step k. Based on these inputs, the model predicts the future target trajectory $\hat{\mathbf{X}}_{f(k)}$ over the prediction horizon.

(W mm^{-1} K^{-1}). The laser heat input q_{laser} is modeled as a surface heat flux with a Gaussian spatial distribution applied to the top surface, reflecting the top-down irradiation of the laser beam. In addition to the laser input, heat losses due to convection and radiation are considered through the terms q_{conv} and q_{rad}, which are imposed as boundary conditions on all exposed surfaces except the bottom substrate. Convective heat transfer depends on the temperature difference between the part and the ambient environment, while radiative heat loss accounts for thermal emission effects based on temperature-dependent radiation. The bottom surface of the substrate is treated with a fixed-temperature boundary condition to maintain ambient conditions. Material deposition is carried out using the inactive element method, where each element is assigned a predefined activation time during the preprocessing stage according to the specified toolpath. During simulation, elements are sequentially activated and incorporated into the mesh once the distance between the laser center and the element centroid falls below the beam diameter r_{beam}. The details of the FEA model setup can be found in [14, 51].

As GAMMA models only heat conduction within melt pool regions in simulations, it tends to overestimate melt pool temperatures compared to sensor measurements. To compensate for this, one approach is to calibrate the material's artificial thermal conductivity using experimental data when an element's temperature surpasses the liquidus threshold [14]. For simplicity, in this work, the extracted melt pool temperature is multiplied by 0.5 to approximate experimentally observed temperature levels, thereby compensating for the overestimation of melt pool temperature. In future work, this treatment will be further enhanced through model calibration using experimental data to address simulation-experiment discrepancies in machine learning surrogate models for real-system implementation.

3. Method

This work proposes a policy learning workflow built upon the DPC framework [35], augmented with a multi-step prediction model for long-horizon forecasting and a structured data generation strategy for improving training efficiency and feasibility, as illustrated in figure 5. Specifically, we first construct a NN-based surrogate model based on the TiDE architecture for one-shot multi-step prediction (section 3.1). We then integrate the learned surrogate model into the DPC framework to train a policy model (section 3.2). Finally, we introduce a model-based data augmentation strategy to guide the generation of high-quality training trajectories (section 3.3).

3.1. Surrogate modeling with TiDE

In this work, we adopt a quantile-based learning approach to capture both the nominal system behavior and the variability induced by stochastic effects. This approach estimates conditional quantiles of

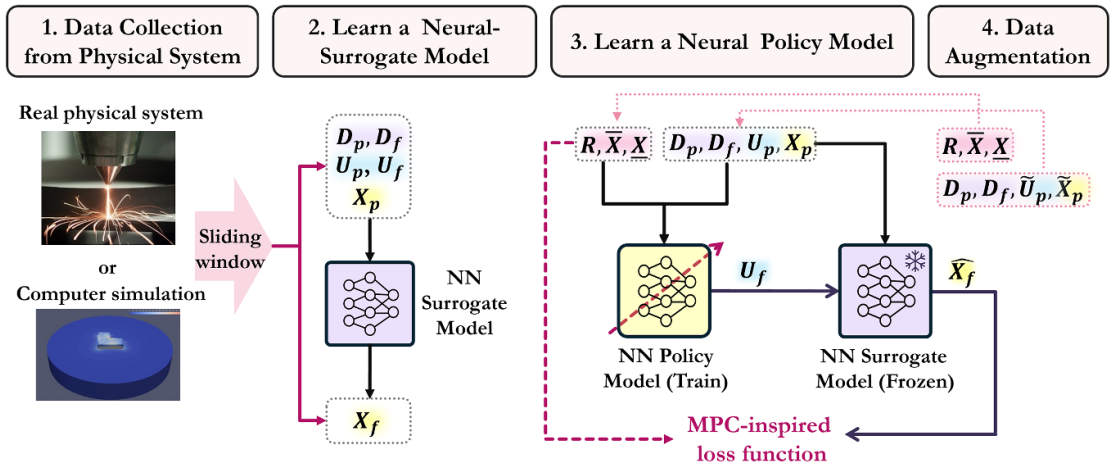


Figure 5. Proposed model-based policy training architecture for the DED control problem. First, collected data are transformed into trajectory datasets using a sliding-window approach. The past target trajectory segment $\mathbf{X}_\mathrm{p}$, past and future known external-variable trajectory segments $\mathbf{D}_\mathrm{p}$ and $\mathbf{D}_\mathrm{f}$, and past and future control input trajectory segments $\mathbf{U}_\mathrm{p}$ and $\mathbf{U}_\mathrm{f}$ are then used to train the neural surrogate dynamics model to predict the future target trajectory $\mathbf{X}_\mathrm{f}$. The learned surrogate model is subsequently frozen and used for neural policy training. The policy model receives the reference trajectory $\mathbf{R}$, lower and upper state bounds $\underline{\mathbf{X}}$ and $\overline{\mathbf{X}}$, together with $\mathbf{D}_\mathrm{p}$, $\mathbf{D}_\mathrm{f}$, $\mathbf{U}_\mathrm{p}$, and $\mathbf{X}_\mathrm{p}$, and generates the future control trajectory $\mathbf{U}_\mathrm{f}$. The frozen surrogate model predicts the future system-state trajectory $\hat{\mathbf{X}}_\mathrm{f}$ over the prediction horizon, which is evaluated using the MPC-inspired loss function for policy optimization. Finally, rollout-generated trajectory samples are iteratively added to the training dataset for data augmentation during policy training.

the output distribution rather than its mean, following the procedure proposed in [53], where quantile-based surrogate models were developed for robust MPC (RMPC). For a given quantile level q, the pointwise quantile loss is defined as

$$L_q\left(x_t, \hat{x}_t\right) = \begin{cases} q\left(x_t - \hat{x}_t\right), & \text{if } x_t \geqslant \hat{x}_t, \\ (1-q)\left(\hat{x}_t - x_t\right), & \text{if } x_t < \hat{x}_t. \end{cases} \tag{11}$$

where the loss assigns asymmetric penalties to overestimation and underestimation, with weights determined by the quantile level q.

We next illustrate the overall surrogate model training procedure, including data generation, sample construction, and model training. In this setting, the sequence of external variables $\mathcal{D}_d$ is given *a priori* and contains known external information. The control input sequence $\mathcal{D}_u$ is generated according to the user's chosen input profile or control policy. By applying $\mathcal{D}_u$ to the system in an open-loop manner under the given conditions $\mathcal{D}_d$, the corresponding output sequence $\mathcal{D}_x$ is obtained either from simulation or from the physical system. Accordingly, we define the sequences as

$$\mathcal{D}_u = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\}, \quad \mathcal{D}_d = \{\mathbf{d}_0, \ldots, \mathbf{d}_{n-1}\}, \quad \mathcal{D}_x = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}. \tag{12}$$

The time-series data are divided into segments using a sliding window of length $w+N$, where the segment index $l \in \mathbb{N}_0^{n-(w+N)}$; for each segment, the past and future components are defined as

$$\begin{aligned} \mathbf{X}_\mathrm{p}^{(l)} &= [\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+w}], & \mathbf{X}_\mathrm{f}^{(l)} &= [\mathbf{x}_{l+w+1}, \ldots, \mathbf{x}_{l+w+N}], \\ \mathbf{D}_\mathrm{p}^{(l)} &= [\mathbf{d}_l, \ldots, \mathbf{d}_{l+w-1}], & \mathbf{D}_\mathrm{f}^{(l)} &= [\mathbf{d}_{l+w}, \ldots, \mathbf{d}_{l+w+N-1}], \\ \mathbf{U}_\mathrm{p}^{(l)} &= [\mathbf{u}_l, \ldots, \mathbf{u}_{l+w-1}], & \mathbf{U}_\mathrm{f}^{(l)} &= [\mathbf{u}_{l+w}, \ldots, \mathbf{u}_{l+w+N-1}]. \end{aligned} \tag{13}$$

With the dataset constructed, the resulting samples are randomly partitioned into training and validation sets. Based on the quantile loss defined in equation (11), the TiDE model is then trained in a supervised manner as follows:

$$\min_{\phi} \quad \frac{1}{(n-N-w+1)} \sum_{l=0}^{n-(w+N)} L_Q\left(\mathbf{X}_\mathrm{f}^{(l)}, \hat{\mathbf{X}}_\mathrm{f}^{(l)}\right) \tag{14a}$$

$$\text{s.t.} \quad \hat{\mathbf{X}}_\mathrm{f}^{(l)} = \mathrm{TiDE}_\phi\left(\mathbf{X}_\mathrm{p}^{(l)}, \mathbf{D}_\mathrm{p}^{(l)}, \mathbf{D}_\mathrm{f}^{(l)}, \mathbf{U}_\mathrm{p}^{(l)}, \mathbf{U}_\mathrm{f}^{(l)}\right), \quad l \in \mathbb{N}_0^{n-(w+N)} \tag{14b}$$

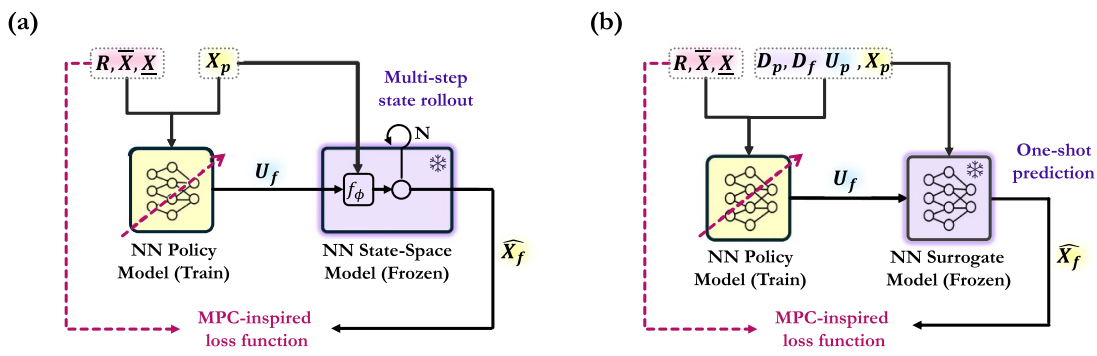


Figure 6. Comparison of the DPC framework. (a) Standard DPC with a recursive state-space model (adapted from [35]). (b) Proposed DPC with a TiDE-based surrogate model enabling one-shot multi-step prediction. The notation for f_ϕ, $\mathbf{R}$, $\underline{\mathbf{X}}$, $\overline{\mathbf{X}}$, $\mathbf{X}_\mathrm{p}$, $\mathbf{U}_\mathrm{f}$, and $\hat{\mathbf{X}}_\mathrm{f}$ follows figure 3. The proposed framework in (b) additionally incorporates the past and future known external-variable trajectory segments $\mathbf{D}_\mathrm{p}$ and $\mathbf{D}_\mathrm{f}$, together with the past control input trajectory segment $\mathbf{U}_\mathrm{p}$, into the policy and surrogate-model inputs.

$$L_Q\left(\mathbf{X}_\mathrm{f}^{(l)}, \hat{\mathbf{X}}_\mathrm{f}^{(l)}\right) = \frac{1}{n_x N}\sum_{j=1}^{n_x}\sum_{i=1}^{N} L_q\left(x_{j,i}^{(l)}, \hat{x}_{j,i}^{(l)}\right) \tag{14c}$$

where $\hat{\mathbf{X}}_\mathrm{f}^{(l)} = [\hat{\mathbf{x}}_1^{(l)}, \ldots, \hat{\mathbf{x}}_N^{(l)}]$, with $\hat{\mathbf{x}}_i^{(l)} \in \mathbb{R}^{n_x}$ denoting the predicted state vector at the ith step of the lth segment. The element $x_{j,i}^{(l)}$ represents the jth state component at the ith prediction step of the lth segment. The loss is computed element-wise over both the state dimension and the prediction horizon, and then averaged over all segments, state dimensions, and time steps.

3.2. Policy learning with DPC

In this section, we describe how the standard DPC framework introduced in section 2.2 is adapted to incorporate a multi-step prediction model as the surrogate and additional decision-relevant variables, by replacing the recursive prediction model in equation (2*b*) with a TiDE-based predictor, as illustrated in figure 6(b).

Using the surrogate model described in section 3.1, the DPC problem is formulated as:

$$\min_{\theta} \sum_{i=1}^{N} \ell_{\mathrm{DPC}}\left(\hat{\mathbf{x}}_i, \mathbf{u}_i, \boldsymbol{\xi}\right) \tag{15a}$$

$$\text{s.t.} \quad \hat{\mathbf{X}}_\mathrm{f} = \mathrm{TiDE}_\phi\left(\mathbf{X}_\mathrm{p}, \mathbf{D}_\mathrm{p}, \mathbf{D}_\mathrm{f}, \mathbf{U}_\mathrm{p}, \mathbf{U}_\mathrm{f}\right) \tag{15b}$$

$$\mathbf{U}_\mathrm{f} = \pi_\theta\left(\boldsymbol{\xi}\right) \tag{15c}$$

$$\mathbf{x}_i \in \mathcal{X}, \quad i \in \mathbb{N}_1^N \tag{15d}$$

$$\mathbf{u}_i \in \mathcal{U} \tag{15e}$$

$$\boldsymbol{\xi} \in \Xi^{\mathrm{ctrl}} \tag{15f}$$

where the policy input $\boldsymbol{\xi}$ is defined as

$$\boldsymbol{\xi} = \begin{bmatrix} \mathbf{X}_\mathrm{p}^\top & \mathbf{D}_\mathrm{p}^\top & \mathbf{D}_\mathrm{f}^\top & \mathbf{U}_\mathrm{p}^\top & \mathbf{R}^\top & \underline{\mathbf{X}}^\top & \overline{\mathbf{X}}^\top \end{bmatrix}, \tag{16}$$

which extends the standard policy input representation in equation (3) by incorporating past known external variables $\mathbf{D}_\mathrm{p}$, future known external variables $\mathbf{D}_\mathrm{f}$, and past control inputs $\mathbf{U}_\mathrm{p}$. Rather than propagating the system state recursively, the TiDE model in equation (15*b*) directly maps past states and inputs, together with future external variables and control inputs, to the predicted state sequence over the horizon, enabling one-shot multi-step prediction of $\hat{\mathbf{X}}_\mathrm{f}$.

The same MPC-inspired loss in equation (7), following [35], is used to optimize the policy parameters via backpropagation through the differentiable closed-loop system. Overall, by incorporating the TiDE-based surrogate model and the augmented input representation, the DPC framework is extended to handle long prediction horizons and strong temporal dependencies.

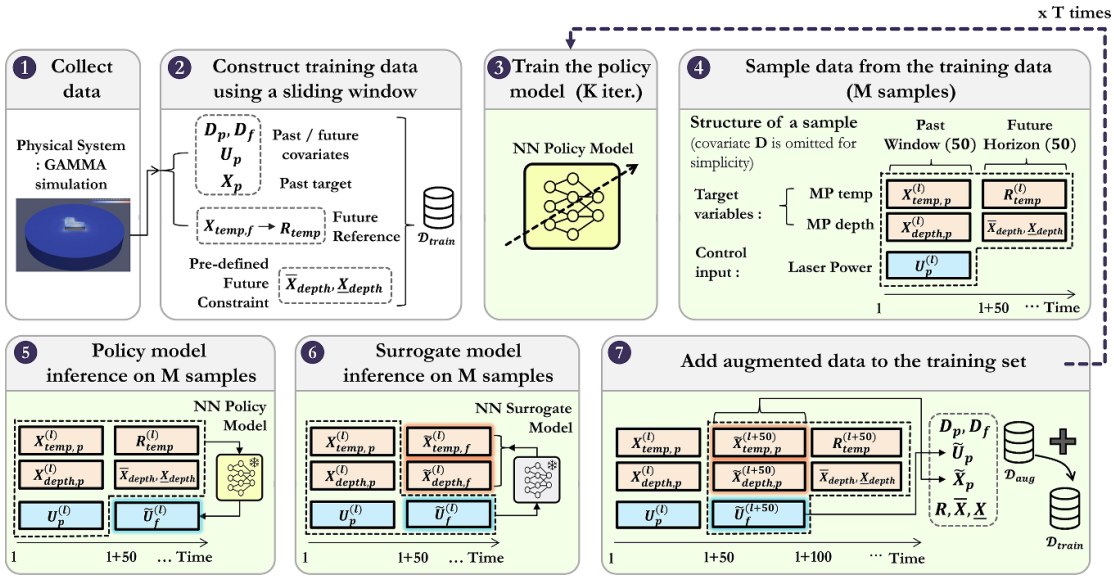


Figure 7. Data augmentation pipeline illustrated using the DED control problem. Here, the subscripts p and f denote past and future trajectory segments, respectively. The collected data are sliced into trajectory segments, where the known external-variable trajectories $\mathbf{D}_{\mathrm{p/f}}$ and the control/state trajectories $\mathbf{U}_{\mathrm{p}}$ and $\mathbf{X}_{\mathrm{p}}$ are used as policy inputs. The future melt pool temperature trajectory $\mathbf{X}_{\mathrm{temp,f}}$ defines the reference trajectory $\mathbf{R}$, while $\underline{\mathbf{X}}_{\mathrm{depth}}$ and $\overline{\mathbf{X}}_{\mathrm{depth}}$ define the melt pool depth constraints (Steps 1–2). The policy model is then updated for K iterations following the procedure described in section 3.2 (Step 3). During data augmentation, a mini-batch of M samples is drawn from the training dataset, where $\mathbf{X}^{(l)}_{\mathrm{temp,p/f}}$ and $\mathbf{X}^{(l)}_{\mathrm{depth,p/f}}$ denote the melt pool temperature and depth trajectories of the lth sample, and $\mathbf{U}^{(l)}_{\mathrm{p}}$ denotes the corresponding past laser power trajectory (Step 4). The policy model generates the rollout-based future control input trajectory $\tilde{\mathbf{U}}^{(l)}_{\mathrm{f}}$ (Step 5), and the surrogate model predicts the future melt pool temperature and depth trajectories $\tilde{\mathbf{X}}^{(l)}_{\mathrm{temp,f}}$ and $\tilde{\mathbf{X}}^{(l)}_{\mathrm{depth,f}}$ (Step 6). Finally, the prediction horizon is shifted forward to construct the augmented dataset $\mathcal{D}_{\mathrm{aug}}$, which is merged into the training dataset $\mathcal{D}_{\mathrm{train}}$ (Step 7). The data augmentation process is repeated for T iterations.

3.3. Data augmentation using model-rollout

Since constructing physically consistent and representative training trajectories is critical for effective policy learning in DPC, yet generating highly spatiotemporally coupled time-series data remains challenging, we propose a model rollout-based data augmentation strategy. Inspired by model-based offline reinforcement learning [54–56], the key idea is to use the learned surrogate model to roll out the current policy model and generate additional trajectories, thereby exposing the policy model to dynamically consistent and constraint-aware behaviors.

The feasibility of generated objective trajectories depends on the underlying system dynamics. In systems with fast dynamics, a wide range of target trajectories can be achieved within a finite prediction horizon, and randomly generated objectives may still provide feasible and informative training signals. In contrast, for systems with slow dynamics, such trajectories are often unrealistic, as the system may be unable to recover from constraint violations or track references within the given horizon. As a result, infeasible objectives can induce overly aggressive or inconsistent control actions during training, leading to degraded policy model performance. This issue is further exacerbated by distribution discrepancy between the training data used to learn the surrogate model and the trajectories induced by the policy model during optimization. When the policy model attempts to satisfy unrealistic objectives within a single horizon, it may generate control inputs that drive the system into regions not well covered by the training data, increasing the risk of model exploitation, where the policy model leverages inaccuracies of the surrogate model to achieve artificially low predicted costs. In such systems, the generation of reference and constraint trajectories should account for gradual regulation dynamics to promote smooth, constraint-aware behavior.

Figure 7 illustrates the proposed data generation process. Initially, the training dataset is constructed from trajectories generated under open-loop control using a sliding-window approach (Steps 1–2 in figure 7), reusing the same data generation and segmentation procedure as in the TiDE training described in section 3.1. In this process, past observations define the look-back window, while future segments define the target trajectories for the policy model. Following the sliding-window segmentation

introduced for TiDE training, the policy input for the lth segment is constructed as

$$\boldsymbol{\xi}^{(l)} = \left[\mathbf{X}_{\mathrm{p}}^{(l)\top} \quad \mathbf{D}_{\mathrm{p}}^{(l)\top} \quad \mathbf{D}_{\mathrm{f}}^{(l)\top} \quad \mathbf{U}_{\mathrm{p}}^{(l)\top} \quad \mathbf{R}^{(l)\top} \quad \underline{\mathbf{X}}^{\top} \quad \overline{\mathbf{X}}^{\top}\right], \tag{17}$$

where the components $\mathbf{X}_{\mathrm{p}}^{(l)}$, $\mathbf{D}_{\mathrm{p}}^{(l)}$, $\mathbf{D}_{\mathrm{f}}^{(l)}$, and $\mathbf{U}_{\mathrm{p}}^{(l)}$ follow the slicing scheme defined in equation (13), and are directly reused to construct the policy input. The reference trajectory $\mathbf{R}^{(l)}$ is constructed from the corresponding future state segment $\mathbf{X}_{\mathrm{f}}^{(l)}$ obtained under open-loop simulation. As a result, the reference inherits dynamically feasible and physically consistent evolution patterns, providing realistic target trajectories for the policy model. The constraint bounds $\underline{\mathbf{X}}$ and $\overline{\mathbf{X}}$ are externally specified based on the control requirements and are not derived from the time-series data.

Using this dataset as the initial training set, the policy model is trained through an iterative procedure that alternates between policy model updates (Step 3 in figure 7) and model-based data augmentation (Steps 4–7 in figure 7). At each data augmentation stage, a mini-batch $\mathcal{B} = \{\mathbf{s}^{(l)}\}_{l=1}^{M}$ of M samples is drawn from the training dataset (Step 4 in figure 7). Each sample $\mathbf{s}^{(l)}$ corresponds to a sliced segment of the time-series trajectories and defines a policy input vector given by

$$\mathbf{s}^{(l)} = \left[\mathbf{X}_{\mathrm{p}}^{(l)\top} \quad \mathbf{D}_{\mathrm{p}}^{(l)\top} \quad \mathbf{D}_{\mathrm{f}}^{(l)\top} \quad \mathbf{U}_{\mathrm{p}}^{(l)\top} \quad \mathbf{R}^{(l)\top} \quad \underline{\mathbf{X}}^{\top} \quad \overline{\mathbf{X}}^{\top}\right]. \tag{18}$$

For each sample $\mathbf{s}^{(l)} \in \mathcal{B}$, the policy model generates a sequence of future control inputs over the prediction horizon (Step 5 in figure 7), as defined in equation (19a). These inputs are then applied to the surrogate model, together with the corresponding past observations and covariates, to predict the future trajectory over the horizon (Step 6 in figure 7), as given in equation (19b).

$$\tilde{\mathbf{U}}_{\mathrm{f}}^{(l)} = \pi_{\theta}\left(\mathbf{s}^{(l)}\right), \tag{19a}$$

$$\tilde{\mathbf{X}}_{\mathrm{f}}^{(l)} = \mathrm{TiDE}_{\phi}\left(\mathbf{X}_{\mathrm{p}}^{(l)}, \mathbf{D}_{\mathrm{p}}^{(l)}, \mathbf{D}_{\mathrm{f}}^{(l)}, \mathbf{U}_{\mathrm{p}}^{(l)}, \tilde{\mathbf{U}}_{\mathrm{f}}^{(l)}\right), \tag{19b}$$

where $\tilde{\mathbf{U}}_{\mathrm{f}}^{(l)}$ denotes the sequence of future control inputs generated by the policy model, and $\tilde{\mathbf{X}}_{\mathrm{f}}^{(l)}$ denotes the corresponding future state trajectory obtained by propagating $\tilde{\mathbf{U}}_{\mathrm{f}}^{(l)}$ through the surrogate model. The tilde notation $(\tilde{\cdot})$ indicates quantities generated via model-based rollout, distinguishing them from the original data collected from the physical system or its simulator.

Using the predicted trajectories, a new sample is generated by shifting the time window forward by the prediction horizon (Step 7 in figure 7), where the predicted future trajectories are reinterpreted as the past components of the next sample.

$$\tilde{\mathbf{X}}_{\mathrm{p}}^{(l+N)} = \tilde{\mathbf{X}}_{\mathrm{f}}^{(l)}, \quad \tilde{\mathbf{U}}_{\mathrm{p}}^{(l+N)} = \tilde{\mathbf{U}}_{\mathrm{f}}^{(l)}. \tag{20}$$

Here, we assume $w = N$ so that $\tilde{\mathbf{X}}_{\mathrm{f}}^{(l)}$ directly serves as the past window of the next sample. If $w < N$, the last w steps of $\tilde{\mathbf{X}}_{\mathrm{f}}^{(l)}$ can be used as the past window. If $w > N$, the past window can be formed by concatenating the last $w - N$ steps of $\mathbf{X}_{\mathrm{p}}^{(l)}$ with $\tilde{\mathbf{X}}_{\mathrm{f}}^{(l)}$. The resulting augmented sample is then constructed as

$$\tilde{\mathbf{s}}^{(l+N)} = \left[\tilde{\mathbf{X}}_{\mathrm{p}}^{(l+N)\top} \quad \mathbf{D}_{\mathrm{p}}^{(l+N)\top} \quad \mathbf{D}_{\mathrm{f}}^{(l+N)\top} \quad \tilde{\mathbf{U}}_{\mathrm{p}}^{(l+N)\top} \quad \mathbf{R}^{(l+N)\top} \quad \underline{\mathbf{X}}^{\top} \quad \overline{\mathbf{X}}^{\top}\right]. \tag{21}$$

These samples collectively form an augmented dataset, which is appended to the existing training dataset as follows:

$$\mathcal{D}_{\mathrm{aug}} = \left\{\tilde{\mathbf{s}}^{(l+N)} \mid \mathbf{s}^{(l)} \in \mathcal{B}\right\}, \tag{22a}$$

$$\mathcal{D}_{\mathrm{train}} \leftarrow \mathcal{D}_{\mathrm{train}} \cup \mathcal{D}_{\mathrm{aug}}, \tag{22b}$$

where $\mathcal{D}_{\mathrm{aug}}$ denotes the set of augmented samples generated via model rollout, and $\mathcal{D}_{\mathrm{train}}$ represents the training dataset, which is iteratively updated by incorporating the augmented data. The procedure then returns to Step 3 in figure 7, where the policy model is updated using both real and augmented data, and the rollout procedure is repeated under the updated policy model. Following prior work in model-based reinforcement learning that limits model rollout steps to mitigate model bias and compounding errors [55, 56], the number of rollout iterations T is kept small. Overall, the proposed approach constructs an augmented training dataset via iterative model-based rollout and dataset update, which is used for policy model training in DPC.

4. Illustrative example

We demonstrate the proof of concept of leveraging multi-step-ahead predictor as the surrogate for DPC on a discrete-time linear system. In this case study, since the states are very responsive to the control actions, randomly generated objective trajectories are sufficient for effective policy training, without requiring additional data augmentation.

4.1. System description

We consider the following discrete-time linear system, adopted from the RMPC case study in [53]:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.2 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} (u_k + \varepsilon_k), \tag{23a}$$

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix}, \quad \varepsilon_k \sim \mathcal{N}\left(0, 0.1^2\right), \tag{23b}$$

where $\mathbf{x}_k = [x_{1,k}, x_{2,k}]^\top \in \mathbb{R}^2$ denotes the system state vector, with $x_{1,k}$ and $x_{2,k}$ representing the two state variables. The control input is $u_k \in \mathbb{R}$, and the disturbance is $\varepsilon_k \in \mathbb{R}$. The reference trajectory $r_k \in \mathbb{R}$ is defined for the tracking variable $x_{1,k}$, while constraints are imposed on $x_{2,k}$ through lower and upper bounds $\underline{c}_k$ and $\bar{c}_k$, respectively. The corresponding MPC problem is formulated as:

$$\min_{u_{k:k+N-1}} \quad \mathcal{L}_{\text{MPC}} = \sum_{i=1}^{N} \left(Q_r \|\hat{x}_{1,k+i} - r_{k+i}\|_2^2 + Q_{du} \|u_{k+i} - u_{k+i-1}\|_2^2 \right) \tag{24a}$$

$$\text{s.t.} \quad \begin{bmatrix} \hat{x}_{1,k+1:k+N} \\ \hat{x}_{2,k+1:k+N} \end{bmatrix} = \text{TiDE}_\phi \left(x_{1,k-w+1:k}, x_{2,k-w+1:k}, u_{k-w:k+N-1} \right) \tag{24b}$$

$$\underline{c}_{k+i} \leqslant \hat{x}_{2,k+i} \leqslant \bar{c}_{k+i}, \quad i \in \mathbb{N}_1^N \tag{24c}$$

where the objective in equation (24*a*) captures the tracking of x_1 and variations in control inputs, the predicted trajectories $\hat{x}_{1,k}$ and $\hat{x}_{2,k}$ are generated using the surrogate model in equation (24*b*), and the constraints in equation (24*c*) enforce bounds on x_2 over the prediction horizon.

4.2. Data generation and surrogate modeling

The training set is generated by simulating the stochastic linear system described in equation (23). Starting from the initial condition $\mathbf{x}_0 = [0, 0]^\top$, the system is propagated for $m = 3 \times 10^5$ time steps, with inputs independently sampled from a uniform distribution as $u_k \sim \mathcal{U}(-5, 5)$ (figure 8(a)). The resulting time series is normalized to the range $[-1, 1]$ using min–max normalization and subsequently segmented using a sliding-window approach to construct input–output pairs, as defined in table 1, for training the dynamics model in equation (24*b*), following the procedure described in section 3.1.

The surrogate model follows the TiDE architecture described in section 2.3. The encoder and decoder each consist of a single residual block with a hidden size of 128, the decoder output dimension is set to 16, and the temporal decoder uses a hidden size of 32. Covariates are projected to a feature dimension of 4, with layer normalization and a dropout rate of 0.2.

The model is trained using the Adam optimizer [57] with a learning rate of 10^{-3} and weight decay of 2×10^{-3}, along with a step learning rate scheduler that decays the learning rate by a factor of 0.9 every 10 epochs. Training is performed for 1000 epochs with a batch size of 64, and early stopping with a patience of 50 epochs is used to select the best model based on validation performance. A median regression loss is employed, corresponding to minimizing the absolute prediction error.

On the validation set, the model achieves MAPE values of 4.89% and 4.75% for the two state variables, respectively, with corresponding RRMSE values of 0.0378 and 0.0379, indicating consistent predictive accuracy across both states. For additional details on the model architecture, training procedure, and validation protocol, we refer the reader to [53].

4.3. Policy learning

For policy model training, the same simulation trajectories used for surrogate model learning are reused to construct the past observations and inputs, whereas the future reference trajectories and constraint bounds are randomly generated. The upper bounds are uniformly sampled from the interval $[2, 5]$, and the lower bounds are defined symmetrically in the interval $[-5, -2]$. The reference trajectories are also uniformly sampled from $[-5, 5]$ (figure 8(b)).

The control policy is defined as

$$u_{k:k+N-1} = \pi_\theta \left(x_{1,k-w+1:k}, x_{2,k-w+1:k}, u_{k-w:k-1}, r_{k+1:k+N}, c_{k+1:k+N}, \bar{c}_{k+1:k+N} \right), \tag{25}$$

where the policy map π_θ is represented by a fully connected NN with two hidden layers of size 128. The inputs follow the structure defined in table 1 where past observations and inputs are concatenated and combined with future reference and constraint trajectories, and the network outputs a control sequence over the prediction horizon.

The policy model is trained within the DPC framework described in section 2.2, where the predicted control sequence is propagated through the learned surrogate model to obtain future state trajectories for loss evaluation in (7). The weighting factors are set to $Q_r = 1.0$, $Q_{du} = 0.1$, and $Q_c = 3.0$, prioritizing constraint satisfaction over tracking performance, while assigning a relatively small weight to the control effort term due to the randomly generated reference and constraint trajectories. The input constraint weight is set to $Q_u = 0$, as input constraints are not considered in this example.

The model is trained for 500 epochs using the Adam optimizer [57] with a learning rate of 10^{-4} and regularization penalty of 10^{-4}, along with a step learning rate scheduler with a decay factor of 0.90 applied every 10 epochs. Training is performed with a batch size of 256, and early stopping with a patience of 50 epochs is used based on validation performance.

4.4. Real-time control performance

To evaluate the performance of the system controlled by the learned policy, we construct an arbitrary reference trajectory composed of step, smooth (sigmoid), and sinusoidal segments, together with time-varying constraint bounds, as shown in figure 8(e). This setup enables evaluation under both abrupt and smooth reference changes, as well as active and inactive constraint regimes.

As shown in figure 8(e), the system accurately tracks the reference when constraints are inactive and follows the constraint boundaries once they become active, demonstrating the desired control behavior. The corresponding control input in figure 8(f) is smooth and continuous throughout the entire trajectory, while it only exhibits peaks during rapid reference transitions, indicating a balance between tracking performance and control effort. Despite disturbances in the system dynamics, the policy maintains the state within the prescribed constraint bounds, particularly around time steps 270, 400, and 500 where constraints are active. This result shows that the policy model can effectively capture the underlying noise characteristics present in the training data. The overall control performance is comparable to that of the baseline MPC controller (figures 8(g) and (h)), while achieving a significantly reduced computation time, with an average of less than 1.0×10^{-4} seconds per step compared to 0.2380 seconds for MPC, evaluated on an AMD EPYC 7413 CPU (48 cores) processor (figures 8(c) and (d)).

Overall, this illustrative example demonstrates how leveraging the multi-step predictive surrogate using TiDE within the DPC framework enables control performance comparable to MPC, while achieving significantly lower computational cost. We next consider a more complex nonlinear system to further evaluate its generalization and robustness under realistic operating conditions.

5. Engineering example: DED control problem

In this section, we implement the proposed DPC framework on a digital twin of the DED AM process. The DED process is subject to uncertainty arising from material variability and changing environmental conditions, which makes it challenging to maintain consistent material properties and avoid defects. This motivates the use of proactive control strategies such as MPC [11]. However, as reported in [53], the computational cost of solving MPC in real time can be prohibitive for DED processes, which motivates the use of DPC to enable faster, real-time implementation. Building on the MPC-based melt pool regulation previously demonstrated for this process [50], this case study examines whether the learned DPC policy can replace the online optimization of MPC while retaining comparable tracking and constraint-handling performance at substantially reduced computational cost.

In this study, the digital twin focuses on the melt pool region at each time step rather than the entire build geometry, as it is where process variations directly influence quality-related outcomes such as porosity, microstructure, and residual stress. In practice, high-fidelity simulation at part-scale for AM is computationally expensive and increases data dimensionality, while it does not benefit the real-time regulation of laser power as it only affects the melt pool-scale instead of the part-scale. Therefore, the virtual counterpart of the digital twin for DED only focuses on modeling the melt pool behaviors, which provides a direct and efficient description of the system while preserving the information needed for

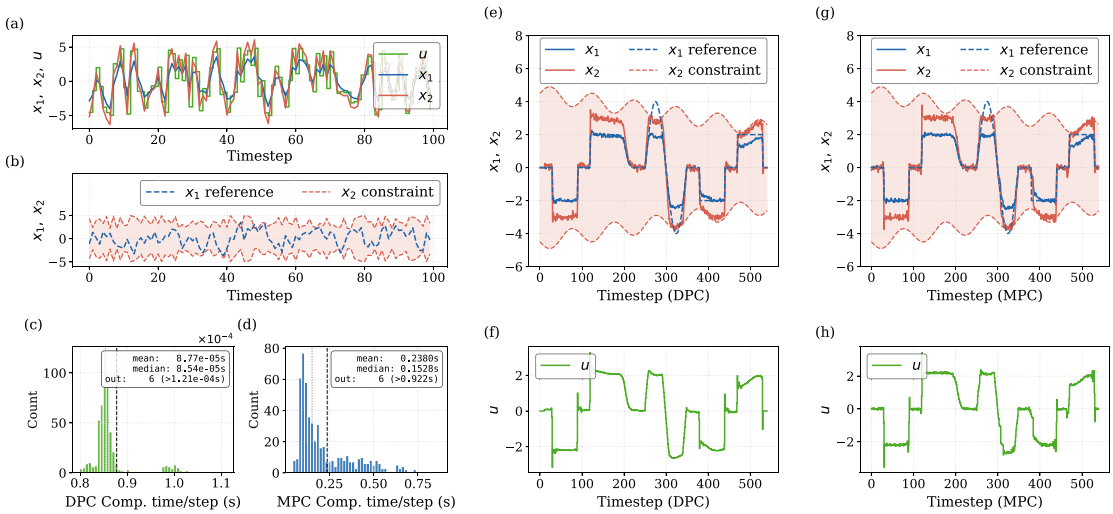


Figure 8. Overview of the generated training data and closed-loop performance comparison between DPC and MPC. A total of 3×10^5 time steps are generated to construct the dataset, of which 100 samples are shown for illustration. (a) Simulated data: state trajectories x_1 and x_2 together with the control input u over a representative window of 100 time steps. (b) Generated objective data: corresponding reference trajectory for x_1 and constraint bounds for x_2 used during data generation. (c) Histogram of computation time per step for DPC (note that the time is scaled by 10^{-4} (s). (d) Histogram of computation time per step for MPC. (e) and (f) Closed-loop state trajectories and control input under DPC. (g) and (h) Closed-loop state trajectories and control input under MPC.

Table 1. Input–output structure for the illustrative linear example with history window $w = 10$ and prediction horizon $N = 10$.

Illustrative Linear Example						
	TiDE			Policy		
	Input		Output	Input		Output
symbol	past $[k-w+1:k]$	future $[k+1:k+N]$	future $[k+1:k+N]$	past $[k-w+1:k]$	future $[k+1:k+N]$	future $[k+1:k+N]$
x_1	•		•	•	• (reference)	
x_1	•		•	•	• (constraints)	
u	•	•		•		•

process optimization [50]. This choice also aligns with the available sources of in-situ sensing data in DED systems, such as co-axial infrared thermography and vision-based sensors [58], from which melt pool features can be extracted.

The proposed digital twin of the DED process comprises four key modules: (a) the physical system, which in practice corresponds to the real DED process but is represented in this work by a high-fidelity FEA simulation; (b) the virtual system, where a policy model is deployed for control, supported by a TiDE-based surrogate model that is not directly used during operation but can be leveraged for policy updates or other tasks when needed; (c) the physical-to-virtual interface, where measurements from the physical system are provided as state observations; and (d) the virtual-to-physical interface, where the learned policy generates control inputs for real-time execution.

At each time step, the current melt pool state, characterized by temperature and depth, is measured from the physical system and provided to the virtual system as feedback. Based on this state observation, known dynamic covariates (e.g. geometric context), and specified future objectives, the policy model generates control actions, which are then applied to the physical system. This receding-horizon control formulation, in which the policy is re-evaluated at every step using updated measurements, enables responsive adjustment to varying operating conditions and supports real-time optimal control in the DED system.

5.1. System description

The physical system is represented by an explicit FEA model, developed by Liao *et al* [51], which performs part-scale thermomechanical simulation of the DED process. The model provides melt pool-related quantities, including temperature and geometry.

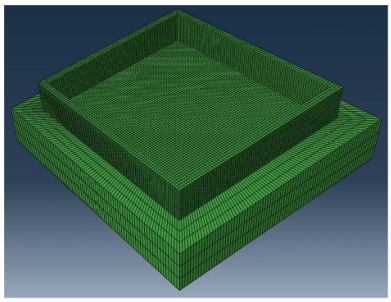


Figure 9. Single-track square.

The target built geometry is a single-track square thin-wall structure with a length of 40 mm on each side, a track width of 1.5 mm, 10 deposition layers, and a layer height of 0.75 mm, as shown in figure 9. The deposited material is 316 L stainless steel on an AISI 1018 substrate. For the simulation, the domain is discretized with an element size of 0.375 mm, resulting in a total of 40 540 elements. The substrate height is set to 10 mm, and the scanning speed is fixed at 7 mm s^{-1}. This setup follows the methodology presented in [50].

The control objective is to regulate melt pool behavior during the printing process. In this framework, the primary requirement is to maintain the melt pool depth within predefined bounds, as it is directly related to defect formation through the dilution ratio [10]. For a layer height of 0.75 mm, the melt pool depth is constrained within the range of 0.075–0.225 mm, corresponding to 10%–30% dilution. In addition, the melt pool temperature is required to track a predefined reference trajectory. In practical manufacturing settings, such trajectories may be obtained from offline process optimization to maximize heat treatment time for precipitation hardening [59].

Notably, the generated temperature reference is designed to improve material properties but does not explicitly enforce constraints on melt pool depth. Similar situations arise in many engineering systems, where target trajectories are defined based on specific criteria while additional constraints must be satisfied. As a result, tracking the reference alone may violate the desired dilution range and lead to defects such as porosity. To address this, the control objective must account for both requirements by prioritizing melt pool depth constraints when conflicts arise with temperature tracking. Following the above control requirements, the corresponding MPC problem is formulated as:

$$\min_{\mathbf{u}_{k:k+N-1}} \quad \mathcal{L}_{\mathrm{MPC}} = \sum_{i=1}^{N} \left(Q_r \| \hat{x}_{\mathrm{temp},k+i} - r_{\mathrm{temp},k+i} \|_2^2 + Q_{du} \| u_{k+i} - u_{k+i-1} \|_2^2 \right) \tag{26a}$$

$$\text{s.t.} \quad \begin{bmatrix} \hat{x}_{\mathrm{temp},k+1:k+N} \\ \hat{x}_{\mathrm{depth},k+1:k+N} \end{bmatrix} = \mathrm{TiDE}_{\phi} \Big(x_{\mathrm{temp},k-w+1:k}, x_{\mathrm{depth},k-w+1:k}, d_{x,k-w:k+N-1}, d_{y,k-w:k+N-1}, z_{k-w:k+N-1}, u_{k-w:k+N-1} \Big) \tag{26b}$$

$$0.075\,\mathrm{mm} \leqslant \hat{x}_{\mathrm{depth},k+i} \leqslant 0.225\,\mathrm{mm}, \quad i \in \mathbb{N}_1^N \tag{26c}$$

$$504\,\mathrm{W} \leqslant u_{k+i} \leqslant 732\,\mathrm{W} \tag{26d}$$

where $x_{\mathrm{temp},k}$ and $x_{\mathrm{depth},k}$ denote the melt pool temperature and melt pool depth at time step k, respectively, and u_k represents the laser power input. The reference signal $r_{\mathrm{temp},k}$ corresponds to the desired temperature trajectory. The spatial covariates $d_{x,k}$ and $d_{y,k}$ denote the nearest boundary distances from the current laser position at time step k along the x- and y-axes, respectively, while z_k represents the position of the laser nozzle along the z-axis in the build direction at time step k, corresponding to the external variables $\mathbf{d}$ in section 2.3. These variables form a multivariate time series that captures geometry-related information. The predicted states $\hat{x}_{\mathrm{temp},k+i}$ and $\hat{x}_{\mathrm{depth},k+i}$ are obtained from the surrogate model in (26*b*). The objective in (26*a*) captures temperature tracking and variations in control inputs, and the constraints in (26*c*) and (26*d*) enforce bounds on the melt pool depth and control input, respectively.

5.2. Data generation and surrogate modeling

The dataset generation in the DED example follows Chen *et al* [50]. To generate a diverse set of operating conditions, a variety of laser power profiles are designed using the approach proposed by Karkaria

et al [59], where each laser power trajectory is parameterized in a 10-dimensional space using a Fourier-series-based representation, and Latin hypercube sampling is employed to efficiently explore the design space. Based on this approach, a total of 99 simulations with varying laser power profiles are conducted, among which 98 are used for training and validation, while the remaining one is reserved for evaluation.

From each simulation, melt pool temperature and melt pool depth are extracted at every time step, together with the corresponding control input and geometry-related variables. Specifically, the dataset consists of melt pool temperature $x_{\text{temp},k}$, melt pool depth $x_{\text{depth},k}$, laser power input u_k, laser z-location z_k, and distances to the nearest geometry boundaries along the x- and y-axes, $d_{x,k}$ and $d_{y,k}$. To construct the training dataset, all simulation trajectories are concatenated and normalized using min–max normalization to $[-1, 1]$. Each laser power profile consists of approximately 6230 time steps, resulting in a total of $m = 610\,417$ time-series segments for 98 profiles. The dataset is then split into training and validation sets and segmented using a sliding window approach following the input–output structure in table 2 and the procedure described in section 3.1. The details of the setup of DED, feature extraction, and data processing can be found in [50].

The surrogate model follows the TiDE architecture described in section 2.3 to capture the coupled evolution of melt pool temperature and depth. The encoder and decoder each consist of a single residual block with a hidden size of 128, while the decoder produces a 16-dimensional representation that is further processed by a temporal decoder with a hidden dimension of 32. Temporal projections with a width of 4 are applied to both past and future covariates, along with layer normalization and a dropout rate of 0.2.

The model is trained using the Adam optimizer [57] with a learning rate of 10^{-3} and a weight decay of 10^{-3}, and a step learning rate scheduler that decays the learning rate by a factor of 0.95 every 2 epochs. Training is carried out for 500 epochs with a batch size of 128, where the best model is selected based on validation loss.

On the validation set, the model achieves MAPE values of 1.24% and 4.25% for temperature and depth, respectively, with corresponding RRMSE values of 0.0515 and 0.0449, indicating accurate prediction of temperature dynamics and reasonable performance for melt pool depth despite its stronger nonlinearity. For additional details on the model architecture, training procedure, and validation protocol, we refer the reader to [53].

5.3. Policy learning

To construct realistic past observations for policy training, we reuse the state and input trajectories generated during the system identification phase. However, in contrast to the illustrative linear example, where both reference and constraint sequences are randomly generated, the DED setting adopts a different approach. Specifically, the upper and lower constraint bounds are fixed based on the system description and are applied uniformly across the prediction horizon, leaving the generation of reference trajectories as the only design choice. In this work, instead of generating reference trajectories randomly or using heuristic approaches, we construct them by leveraging the future segments of the melt pool temperature obtained through the sliding window segmentation, as illustrated in Step 2 of figure 7, to achieve realistic reference signals that reflect feasible system behavior and improve the fidelity of the reference tracking objective.

To construct the initial training dataset, simulation trajectories are normalized and split into training and validation sets and segmented following the input–output structure in table 2 using a sliding-window approach. The resulting policy model can be written as

$$\begin{aligned} u_{k:k+N-1} = \pi_\theta\Big(&x_{\text{temp},k-w+1:k}, x_{\text{depth},k-w+1:k}, z_{k-w:k+N-1}, d_{x,k-w:k+N-1}, d_{y,k-w:k+N-1}, u_{k-w:k-1}, \\ &r_{\text{temp},k+1:k+N}, \underline{x}_{\text{depth},k+1:k+N}, \overline{x}_{\text{depth},k+1:k+N},\Big), \end{aligned} \tag{27}$$

where the policy map π_θ is represented by a fully connected NN with two hidden layers of size 1024.

The policy model is trained for 500 epochs using the AdamW optimizer [60] with a learning rate of 10^{-4} and weight decay of 10^{-4}, with early stopping applied based on validation performance using a patience of 100 epochs. During training, additional data are generated through periodic rollout of the current policy using the learned dynamics model, following the procedure to be described in section 5.4, where new samples are collected every 50 epochs and appended to the training dataset, resulting in a progressively expanding dataset over the course of training. Training follows the DPC framework described in section 2.2, where the predicted control sequence is propagated through the surrogate

Table 2. Input–output structure of the DED control problem with history window $w = 50$ and prediction horizon $N = 50$.

DED Control Problem							
		TiDE			Policy		
		Input		Output	Input		Output
Description	Symbol	past $[k-w+1:k]$	future $[k+1:k+N]$	future $[k+1:k+N]$	past $[k-w+1:k]$	future $[k+1:k+N]$	future $[k+1:k+N]$
mp-temp	x_{temp}	•		•	•	• (reference)	
mp-depth	x_{depth}	•		•	•	• (constraints)	
u	u	•	•		•		•
z-loc	z	•	•		•	•	
x-dist	d_x	•	•		•	•	
y-dist	d_y	•	•		•	•	

model to evaluate the loss defined in equation (7). The loss consists of terms to penalize reference tracking error, control effort, and constraint violation with weighting factors $Q_r = 1.0$, $Q_{du} = 1.0$, $Q_c = 10.0$, and $Q_u = 10.0$.

Following the observation in [50], constraint enforcement is applied conditionally depending on the operating regime, such that during the first three layers the lower-bound depth constraint is excluded from the loss, as the melt pool depth remains below the target dilution range due to substrate boundary conditions. To maintain a balanced loss scale in this regime, the tracking term is upweighted to 10, and the control effort term is increased proportionally. In addition, during layer transition phases, where satisfying the lower-bound constraint is inherently difficult, the lower-bound violation term is also deactivated for stable training, while outside of these conditions both upper and lower bound constraints are enforced.

5.4. Data augmentation

We employ a data augmentation strategy to address the difficulty of recovering constraint satisfaction within a short prediction horizon when the system state deviates significantly from the constraint bounds. Figure 10(a) shows training data generated from open-loop control trajectories, which contain a large portion of states that significantly violate the constraint bounds. In contrast, figure 10(b) presents the behavior of an MPC implementation, where the system tracks the reference when constraints are inactive and remains near the constraint boundary once they become active while minimizing tracking error. In particular, the melt pool temperature closely follows the reference in the early stage (e.g. before approximately time step 1800) and subsequently stays near the constraint boundary, where the constraints become active, resulting in a trade-off between constraint satisfaction and tracking performance.

Under this setting, training the policy using only sliding-window samples from open-loop trajectories encourages constraint satisfaction within a single prediction horizon. This often leads to aggressive control actions aimed at rapid feasibility recovery. Such behavior is observed in figure 11 during the initial phase (time steps 50–100), where the policy trained solely on open-loop data exhibits abrupt control inputs to mitigate constraint violations.

To address this issue, we incorporate data augmentation following the procedure in section 3.3. Specifically, every $K = 50$ epochs, $M = 102\,400$ samples (approximately 20% of the initial training dataset size of 488 254 samples) are generated and added to the training dataset. A sensitivity study on the augmentation frequency K and size M, supporting this choice, is provided in appendix A. Each column in figure 11 corresponds to the rollout generated by the policy $\pi^{(i)}$ at augmentation stage i, with the first few rollout steps shown for clarity. The melt pool temperature and depth trajectories are obtained from rollouts of the surrogate dynamics model. Since samples are drawn randomly at each stage, observing a continuous trajectory from a single rollout is unlikely in practice; a continuous rollout is shown here for illustration.

In figure 11, at augmentation stage 1, the rollout generated by the initial policy $\pi^{(0)}$ appears to satisfy the constraint, but does so through abrupt changes in laser power. Such control action trajectories lie outside the data distribution captured by the dynamics model, and the corresponding surrogate model predictions are therefore unreliable. As the rollout and policy update steps are repeated, the generated trajectories progressively move toward the constraint boundary while exhibiting smoother control actions. This indicates that the policy incrementally learns constraint-aware behavior consistent with the control objective. Consequently, the augmented dataset becomes more realistic and physically consistent, enabling constraint satisfaction without relying on abrupt corrective actions. However, excessive rollout

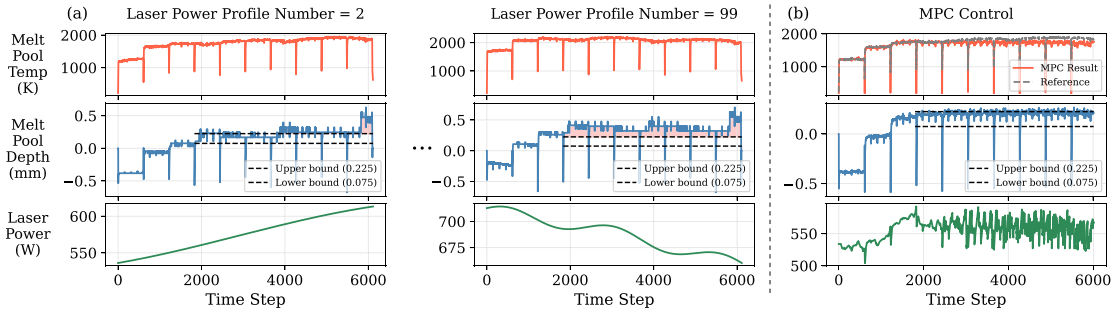


Figure 10. Comparison of system behavior under open-loop and MPC. In both plots, the top, middle, and bottom plots represent melt pool temperature, melt pool depth, and laser power input, respectively, and constraint-violating regions are highlighted. (a) System behavior under open-loop control using two representative laser power profiles (Nos. 2 and 99). (b) MPC result under one selected operating condition (laser power profile No. 1, with constraint bounds on melt pool depth).

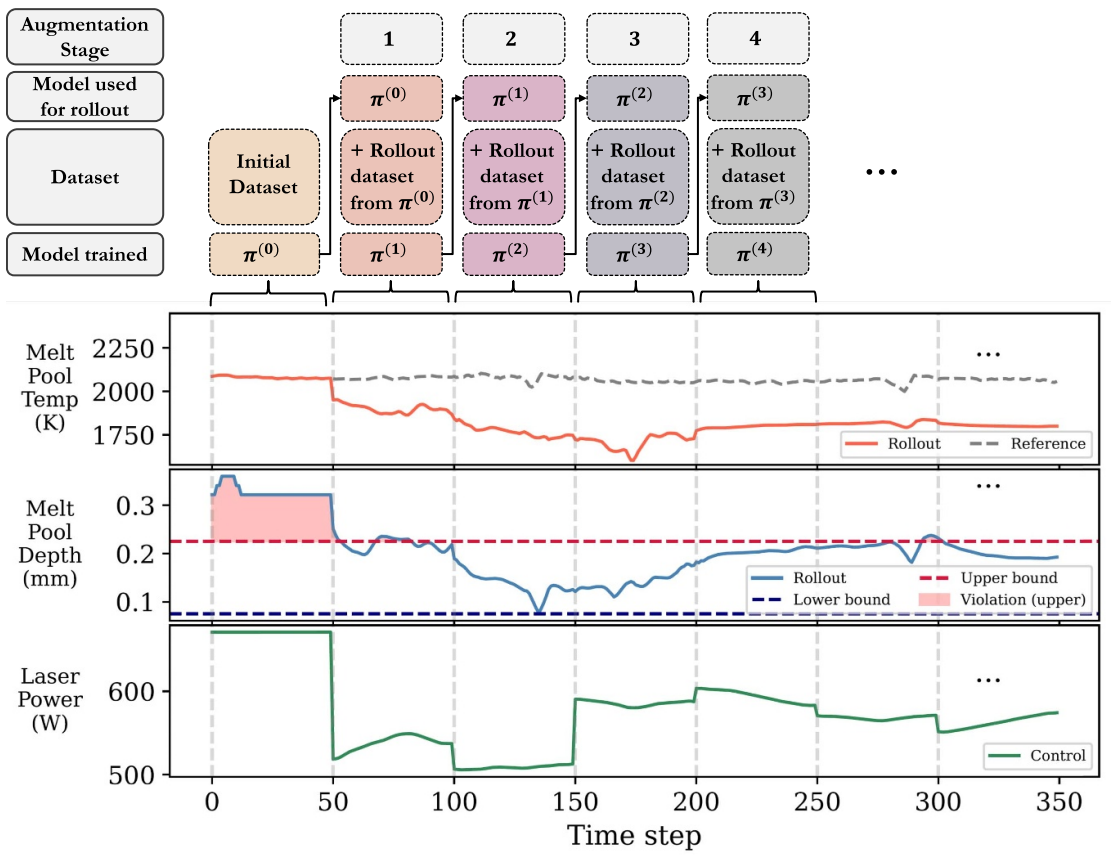


Figure 11. Illustration of the iterative data augmentation and policy update process. At augmentation stage *i*, the previously trained policy is used to generate rollout data, which is aggregated with the existing dataset to train the next policy. The bottom plots show the corresponding rollout trajectories: melt pool temperature (top), melt pool depth with constraint bounds (middle), and control input (bottom). Constraint violations observed in early rollouts are progressively reduced as the policy is updated through successive iterations.

iterations may accumulate model errors and degrade data quality; therefore, the rollout horizon must be carefully chosen to balance policy improvement and model reliability, as discussed in [55].

5.5. Real-time closed-loop control performance

The performance of the proposed policy learning framework in the DED setting is shown in figure 12, where the DPC policy with and without data augmentation is compared against MPC and RMPC from [50, 53]. In particular, figure 12(b) corresponds to the DPC policy trained with three data augmentation iterations. While MPC employs a nominal prediction model without explicitly accounting for

uncertainty, RMPC incorporates quantile-based uncertainty estimation to achieve robust constraint satisfaction.

Control performance. Before comparing the control performance of different methods, we first define the ideal control behavior in the considered DED process. Each layer corresponds to approximately 600 time steps, and transitions between layers can be identified by sharp drops in melt pool temperature and depth. In the first three layers, the melt pool depth constraint is inactive. Therefore, the ideal behavior in this regime is to closely track the melt pool temperature reference while maintaining moderate control effort, avoiding excessive peaks in the input. From the fourth layer, the depth constraint becomes intermittently active. However, since constraint activation remains sparse, temperature reference tracking continues to be the primary objective, similar to the earlier layers. From the fifth layer onward, the constraint becomes active over most of the time horizon. In this regime, the control objective shifts to maintaining constraint satisfaction while preserving tracking performance as much as possible. The ideal behavior is to operate as close as possible to the constraint boundary without violating it, effectively balancing constraint satisfaction and reference tracking.

During the first four layers (time steps $\in [0, 2440]$), better performance corresponds to error distributions concentrated near zero tracking error, consistent with the trends observed in figures 12(e)–(g). This is further supported by the quantitative results in table 3. MPC achieves the best tracking performance (mean error of 20.25 K), followed by RMPC (23.31 K). The baseline DPC without data augmentation exhibits degraded performance; however, its performance improves progressively with rollout-based augmentation, reaching a level comparable to MPC after three data augmentation iterations (26.23 K).

We now turn to the later layers (time steps $\in [2440, 6050]$), where the melt pool depth constraint becomes active. Under this regime, the key performance difference lies in how effectively each method balances constraint satisfaction and tracking performance. Constraint satisfaction behavior is compared in figures 12(h)–(j). The proposed data-augmented DPC policy (figure 12(h)) operates close to the constraint boundary while exhibiting substantially fewer violations than MPC (figure 12(i)), without being overly conservative despite not explicitly modeling uncertainty. MPC exhibits frequent constraint violations due to its nominal formulation, whereas RMPC (figure 12(j)) significantly reduces violations by adopting conservative control behavior. This suggests that although DPC does not explicitly model uncertainty as in RMPC, it adapts to it through the data distribution, enabling the policy to learn constraint-aware behavior from data generated under stochastic system dynamics. In contrast, RMPC tends to be overly conservative, showing oscillatory behavior near the constraint boundary and sacrificing reference tracking performance. The proposed DPC achieves closer constraint-boundary tracking while maintaining improved tracking performance.

This behavior is further quantified in table 3. RMPC achieves a low violation rate (0.3%) by incorporating uncertainty bounds, but at the cost of increased tracking error (165.54 K). In contrast, MPC achieves the lowest tracking error (120.10 K) but exhibits a high violation rate (9.06%), indicating insufficient robustness. The proposed DPC policy $\pi^{(3)}$ achieves a balanced trade-off between tracking performance and constraint satisfaction, maintaining tracking performance close to MPC (130.19 K) while incurring only a moderate violation rate (3.14%).

Overall, the DPC policy $\pi^{(3)}$ achieves a balanced trade-off between tracking performance and constraint satisfaction, maintaining tracking performance close to that of MPC while significantly reducing constraint violations. Although it exhibits a higher violation rate than RMPC, it operates closer to the constraint boundary, resulting in improved overall reference tracking performance. These results highlight that the proposed rollout-based data augmentation enables DPC to effectively bridge the gap between the efficiency of MPC and the robustness of RMPC. Furthermore, by limiting the rollout depth, the method mitigates the accumulation of model errors while maximizing the benefit of augmented data, resulting in stable and physically consistent real-time control performance.

Robustness to observation noise and input disturbances. To assess the robustness of the learned policy beyond the nominal setting, we conduct additional closed-loop evaluations under operational disturbances that may arise during deployment. We consider two sources depending on where they enter the control loop. The first is noise applied to the observation, which corrupts the state measurements provided to the policy, such as sensor noise. The second is disturbance affecting the actuation, which perturbs the control input applied to the system, such as variations in laser power. Accordingly, four test cases are considered. In the nominal case (`base`), no additional perturbation is applied. In the sensor-noise case (`noise`), zero-mean Gaussian noise is added to the melt pool states observed from the physical system, with standard deviations of 20 K for melt pool temperature and 0.005 mm for melt pool depth. In the laser-power disturbance case (`dist`), zero-mean Gaussian noise with a standard

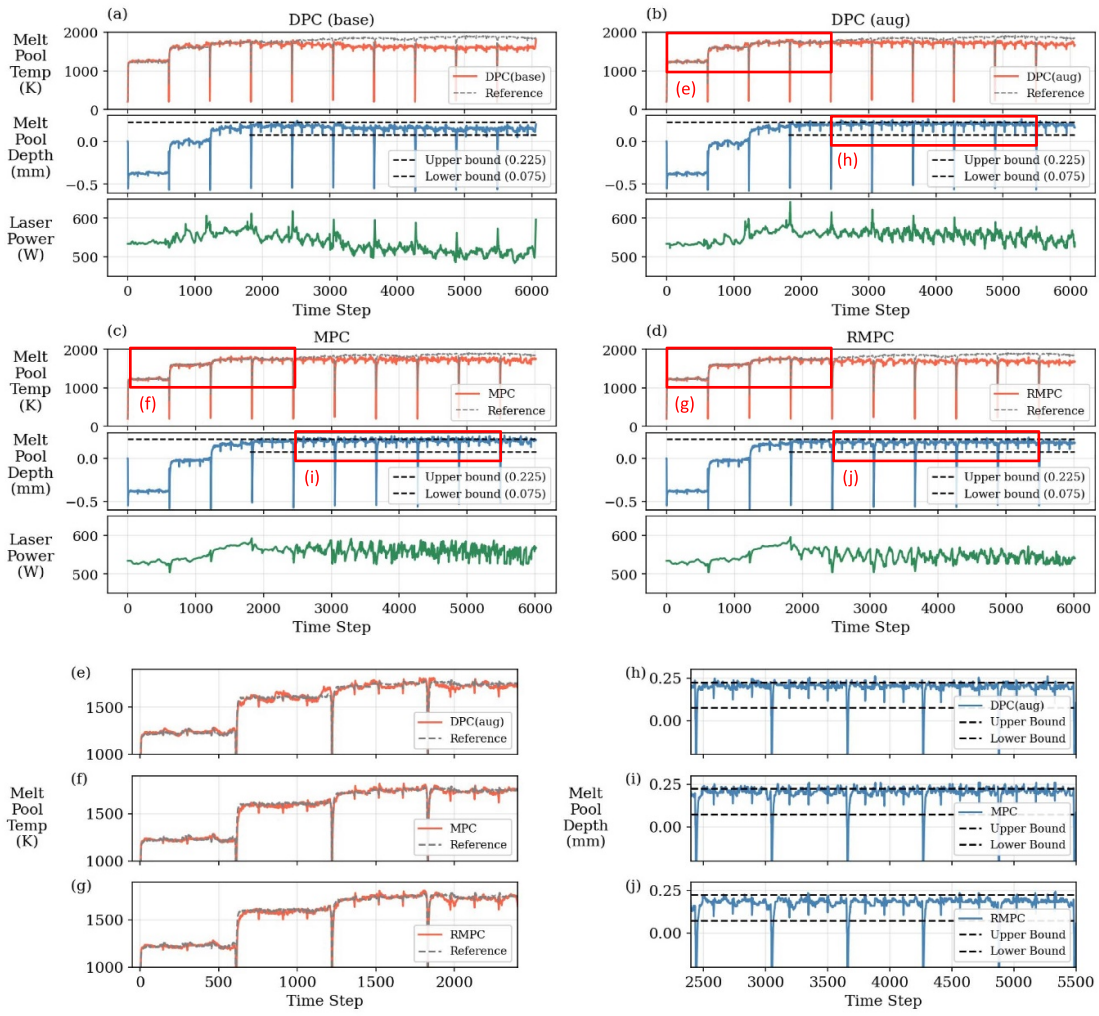


Figure 12. Comparison of control performance between DPC (base), DPC (augmented), MPC, and RMPC. (a)–(d) Full trajectories of melt pool temperature, melt pool depth, and laser power for (a) DPC (base), (b) DPC (augmented), (c) MPC, and (d) RMPC. (e)–(g) Zoomed-in melt pool temperature responses with corresponding reference for (e) DPC (augmented), (f) MPC, and (g) RMPC. (h)–(j) Zoomed-in melt pool depth with constraint bounds for (h) DPC (augmented), (i) MPC, and (j) RMPC.

Table 3. Performance comparison of DPC, MPC, and RMPC.

	DED Control Performance							
	MP Temp. Tracking Error $\lvert x_{\text{temp}} - r_{\text{temp}} \rvert$ (K) time steps $\in [0, 2440]$		MP Temp. Tracking Error $\lvert x_{\text{temp}} - r_{\text{temp}} \rvert$ (K) time steps $\in [2440, 6050]$		MP Depth Violation $\max\left(0,\ x_{\text{depth}} - \bar{x}_{\text{depth}}\right)$ (mm)			
Method	Mean	Std	Mean	Std	Mean	Std	Count	Rate
DPC ($\pi^{(0)}$)	44.21	43.96	227.13	83.31	0.0073	0.0040	16	0.0026
DPC ($\pi^{(1)}$)	68.24	69.97	134.16	62.52	0.0080	0.0066	138	0.0228
DPC ($\pi^{(2)}$)	29.95	43.42	128.26	62.12	0.0096	0.0075	223	0.0369
DPC ($\pi^{(3)}$)	26.23	42.68	130.19	62.33	0.0080	0.0068	190	0.0314
DPC ($\pi^{(4)}$)	29.10	43.57	130.30	63.47	0.0081	0.0067	201	0.0332
DPC ($\pi^{(5)}$)	35.01	44.07	125.89	62.55	0.0081	0.0063	221	0.0365
MPC	20.25	40.52	120.10	60.15	0.0099	0.0082	561	0.0906
RMPC	23.31	43.57	165.54	57.34	0.0088	0.0054	19	0.0031

deviation of 10 W is added to the control input before it is applied to the physical system. The combined case (`both`) includes both perturbations. The same four cases are applied to MPC and RMPC for comparison.

Figure 13 summarizes the closed-loop performance under the four cases, and the corresponding detailed trajectories are provided in appendix B. The learned DPC policy maintains closed-loop

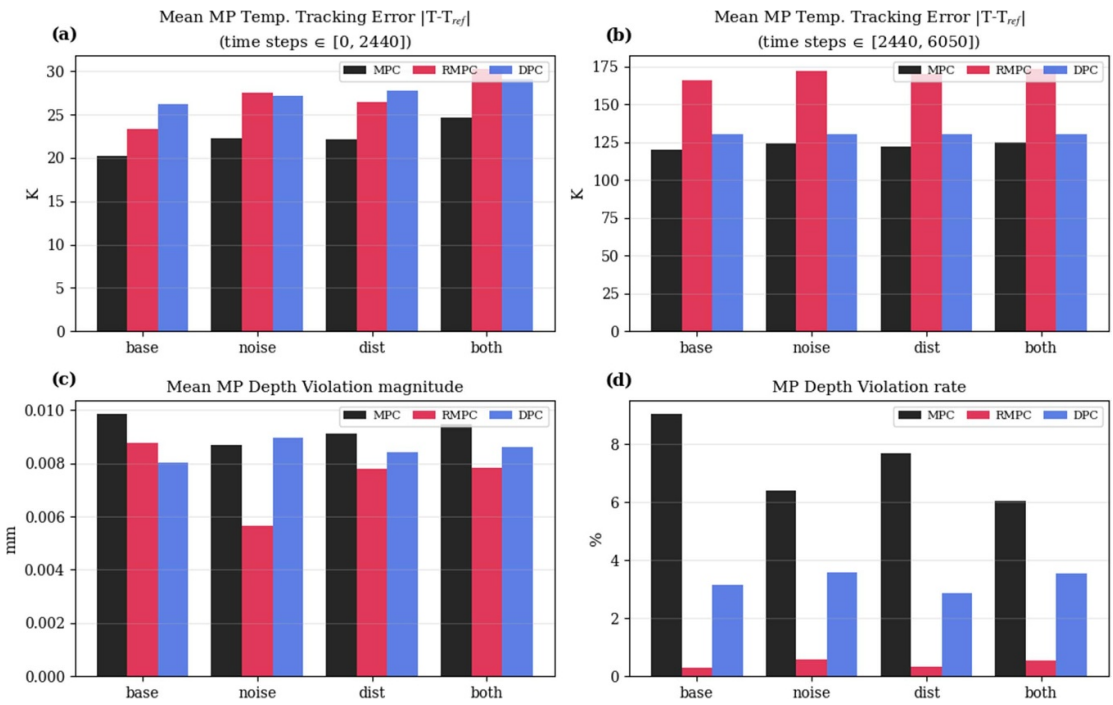


Figure 13. Statistical comparison of control performance between DPC, MPC, and RMPC under the four robustness test cases: nominal condition (`base`), sensor noise (`noise`), laser-power disturbance (`dist`), and combined sensor noise and laser-power disturbance (`both`). Panels (a) and (b) show the mean melt pool temperature tracking error in time steps $[0, 2440]$, where the depth constraint is mostly inactive, and $[2440, 6050]$, where the depth constraint becomes active, respectively. Panel (c) shows the mean depth over-bound magnitude, and panel (d) shows the depth violation rate.

behavior similar to the nominal case under all perturbation settings, without significant degradation in either tracking or constraint-handling performance. The mean temperature tracking error in time steps $[0, 2440]$ changes only moderately, from 26.23 K in the nominal case to 27.19 K under sensor noise, 27.71 K under laser-power disturbance, and 29.03 K when both perturbations are present. In time steps $[2440, 6050]$, where the depth constraint is active over most of the horizon, the tracking error remains nearly unchanged across all cases, ranging from 130.1–130.5 K. The depth violation rate also remains within a narrow range, from 2.88%–3.57%, indicating that the learned policy maintains similar constraint-handling behavior even when noisy observations and input disturbances are introduced.

Compared with MPC and RMPC, DPC preserves the same qualitative trade-off observed in the nominal study. MPC achieves slightly lower tracking error but shows substantially higher depth violation rates across all cases, ranging from 6.05%–9.06%. RMPC achieves the lowest violation rates, below 0.57%, but at the cost of much larger tracking error in the constraint-active region, approximately 165.5–173.5 K, reflecting its conservative behavior. In contrast, DPC maintains tracking performance close to MPC in the constraint-active region while substantially reducing violations relative to MPC, without becoming as conservative as RMPC. These results indicate that the learned policy is reasonably robust to observation noise and actuation-side disturbances within the tested perturbation levels.

We note that these robustness results are empirical and should not be interpreted as formal robustness guarantees. Unlike MPC and RMPC, which re-solve an optimization problem using the current state at each time step, the learned DPC policy is a fixed parameterized feedback law and therefore is not certified beyond its training distribution [18]. Consequently, although the receding-horizon feedback structure is retained, its online adaptation to out-of-distribution conditions is more limited than that of MPC. Addressing this limitation for broader deployment would require additional certification or correction mechanisms, such as control barrier function–based corrections [37] or predictive safety filters [38], which we leave for future work.

Computational time. The computational time per control step for DPC and RMPC is shown in figure 14. All experiments were conducted on an AMD EPYC 7413 CPU (48 cores) processor. DPC achieves consistently low inference times on the order of 10^{-4} seconds (mean 6.09×10^{-4} s), whereas RMPC requires substantially higher computation time, with a mean of 0.1634 s and a maximum exceeding 1.2 s, along with significant variability.

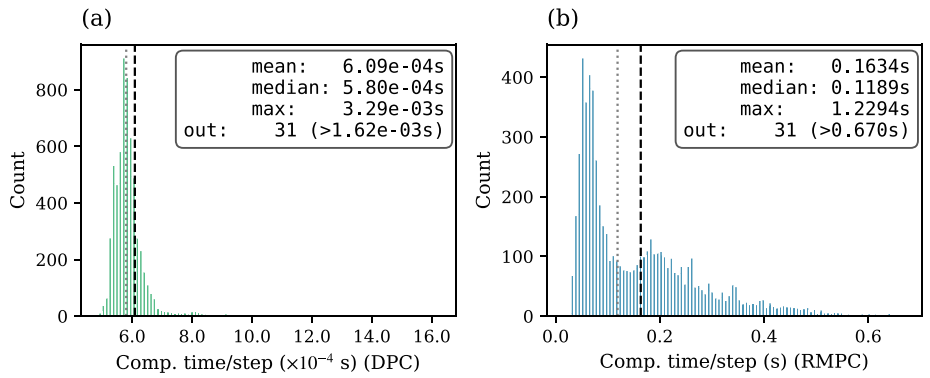


Figure 14. Histogram of computation time per control step for (a) DPC and (b) RMPC, evaluated on an AMD EPYC 7413 CPU (48 cores). DPC achieves a mean inference time of 6.09×10^{-4} s, whereas RMPC requires a mean of 0.1634 s with significant variability.

In DED systems, laser power can be actuated at frequencies on the order of several hundred Hz, while MPC-based methods in this setting typically operate below 10 Hz due to computational limitations, preventing full utilization of the available actuation bandwidth. In contrast, the near-constant inference time of DPC enables high-frequency control without sacrificing stability. Furthermore, while the computational cost of MPC-based methods increases with problem complexity, the inference cost of DPC, although dependent on the size of the NN, remains effectively constant and significantly lower than that of MPC.

Overall, these results demonstrate that the proposed DPC framework provides a substantial computational advantage, making it well-suited for high-frequency, real-time control and edge computing scenarios where conventional MPC approaches may be impractical.

6. Conclusion

In this work, we presented a model-based policy learning framework for real-time control of high-frequency manufacturing processes based on the DPC framework, with DED AM serving as a representative application example. To address key limitations of the existing DPC framework, we introduced two main extensions: (i) a TiDE-based multi-step dynamics model that enables direct long-horizon prediction without recursive rollouts, and (ii) a structured data generation strategy based on model rollouts that produces physically consistent and constraint-aware training trajectories.

The proposed framework was validated on an engineering case study in DED, where the learned policy model achieved control performance comparable to MPC and RMPC while reducing computation time for real-time inference by over two orders of magnitude compared to RMPC. Notably, the approach leverages system trajectories generated under open-loop control to preserve realistic reference behavior, while augmenting them through model-based rollouts, enabling the policy to capture the trade-off between tracking and constraint satisfaction and to learn gradual, constraint-aware regulation dynamics. Despite these advantages, several limitations remain. First, control performance depends on the design of the MPC-inspired loss function, particularly the choice of weighting coefficients. In applications such as AM, where process dynamics vary across layers and transient regimes, careful tuning of these weights is critical. Second, constraint satisfaction remains a fundamental limitation of the learned policy, which lacks formal guarantees. Surrogate-model errors may propagate to the policy during training, although the quantile bounds learned by the surrogate could be incorporated to provide probabilistic constraint satisfaction under model uncertainty [53]. Beyond model uncertainty, the learned policy is not certified for conditions outside the training distribution, as discussed in section 5.5. Extending the framework with formal certification or online correction mechanisms, such as control barrier function–based corrections [37] or predictive safety filters [38], remains an important direction for future work. Finally, the framework assumes a fixed dynamics model during policy learning, which may limit its applicability under non-stationary operating conditions. In practical environments, concept drift may arise due to changes in system dynamics or operating conditions, requiring updates to both the dynamics model and the policy model. Since policy learning relies on the learned dynamics model, effective strategies for drift detection, update scheduling, and joint or sequential adaptation of the dynamics and policy

models remain important directions for future work. Relatedly, the present study is evaluated within a simulation-based digital twin environment. Physical implementation of the learned policy, together with model calibration against additional experimental data, is therefore a necessary next step toward assessing transferability to real DED systems.

Acknowledgments

We are grateful for the grant support from the National Science Foundation's Engineering Research Center for Hybrid Autonomous Manufacturing: Moving from Evolution to Revolution (ERC-HAMMER), under the Award Number EEC-2133630 and the NSF MADE-PUBLIC Future Manufacturing Research Grant Program under Award Number CMMI-2037026. Seul Lee acknowledges the Samsung Electronics Doctoral Scholarship, and Yi-Ping Chen acknowledges the Taiwan-Northwestern Doctoral Scholarship.

Data availability statement

The data cannot be made publicly available upon publication because they are not available in a format that is sufficiently accessible or reusable by other researchers. The data that support the findings of this study are available upon reasonable request from the authors.

Conflict of interest

The authors declare no competing interests.

Appendix A. Sensitivity analysis of data augmentation parameters

To examine the effect of the augmentation frequency K and the augmentation size M, we conduct a sensitivity study over a grid of $K \in \{10, 50, 100\}$ epochs and $M \in \{51\ 200,\ 102\ 400,\ 204\ 800\}$ samples, resulting in nine settings in total. For each setting, the best checkpoint across augmentation stages is selected based on a combined ranking over the four performance metrics used in table 3, namely the mean melt pool temperature tracking error in time steps $[0, 2440]$ and $[2440, 6050]$, the mean depth over-bound magnitude, and the depth violation rate. The selected checkpoints are compared against the MPC and RMPC baselines in figure A.15.

Across the tested settings, the qualitative trade-off relative to MPC and RMPC is largely preserved. Specifically, the temperature tracking error in time steps $[0, 2440]$ is comparable to or worse than that of MPC and RMPC, while the tracking error in time steps $[2440, 6050]$ is substantially lower than that of RMPC and slightly higher than that of MPC. The mean depth over-bound magnitude is comparable to or lower than that of RMPC, except for the case of $K = 100$ and $M = 102\ 400$ in figure A.15(c), which exceeds the MPC baseline, and the depth violation rate is higher than that of RMPC but lower than that of MPC across all settings.

At the same time, larger augmentation intervals tend to produce less favorable results. In particular, the cases with $K = 100$ show relatively larger tracking error in figure A.15(a), and the worst-performing settings in the depth-constraint metrics in figures A.15(c) and (d) also correspond to $K = 100$. This may be because augmentation is introduced only after the policy has already been trained for many epochs on the initial open-loop dataset, reducing the effect of newly added rollout samples. Another observation is that the selected best augmentation stage tends to decrease as K increases, with stages 5–8 for $K = 10$, stages 2–5 for $K = 50$, and stages 1–3 for $K = 100$. This trend suggests that when the augmentation interval is small, more augmentation stages may be needed to provide sufficient total training iterations, even though model error in the augmented data may accumulate across stages. Among the tested settings, we adopt $K = 50$ and $M = 102\ 400$, and this setting provides balanced overall performance across the four metrics. We note that the augmentation frequency and size should be tuned according to the task, model fidelity, and available data.

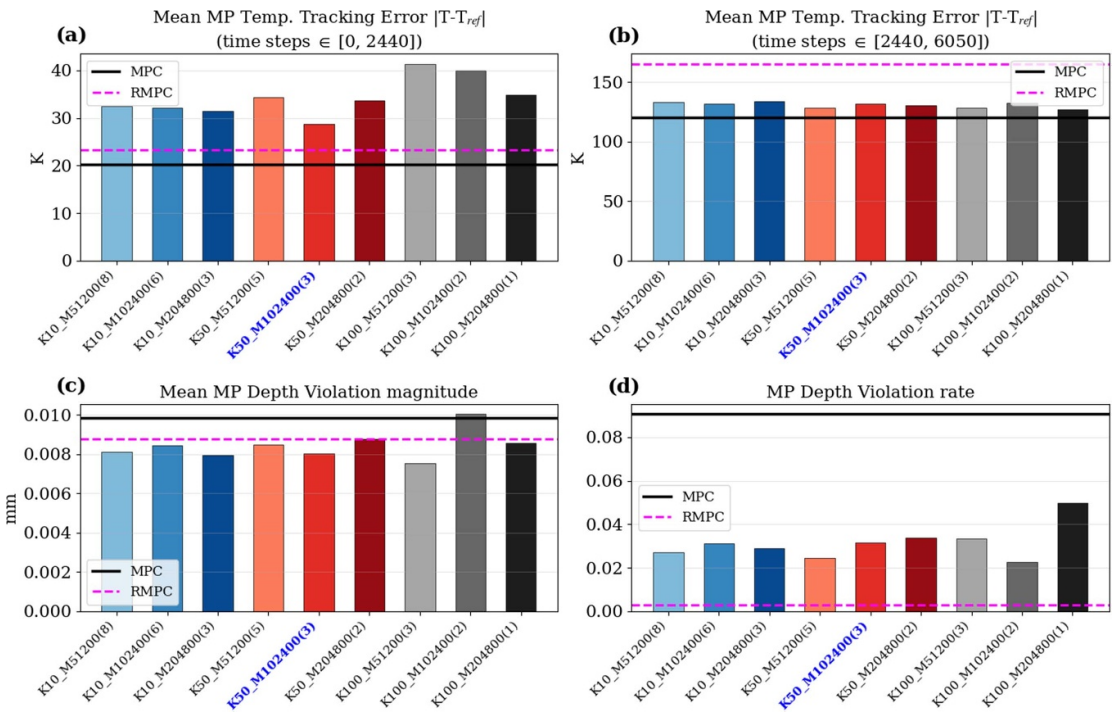


Figure A.15. Sensitivity analysis of the rollout-based augmentation frequency K and augmentation size M. Each bar represents the best checkpoint selected for a given (K, M) setting based on the combined ranking over four performance metrics, where the number in parentheses denotes the augmentation stage at which the selected checkpoint is obtained. Panels (a) and (b) show the mean melt pool temperature tracking error in time steps $[0, 2440]$ and $[2440, 6050]$, respectively. Panels (c) and (d) show the mean depth over-bound magnitude and the depth violation rate, respectively. The solid and dashed horizontal lines indicate the MPC and RMPC baselines.

Appendix B. Closed-loop trajectories under perturbations

Figure B.16 shows the detailed closed-loop trajectories of melt pool temperature and depth for DPC, MPC, and RMPC under the four robustness test cases described in the robustness evaluation of section 5.5. Consistent with the statistical results in figure 13, all three controllers maintain closed-loop behavior similar to their respective nominal cases, and DPC preserves the same qualitative trade-off relative to MPC and RMPC under the tested perturbations.

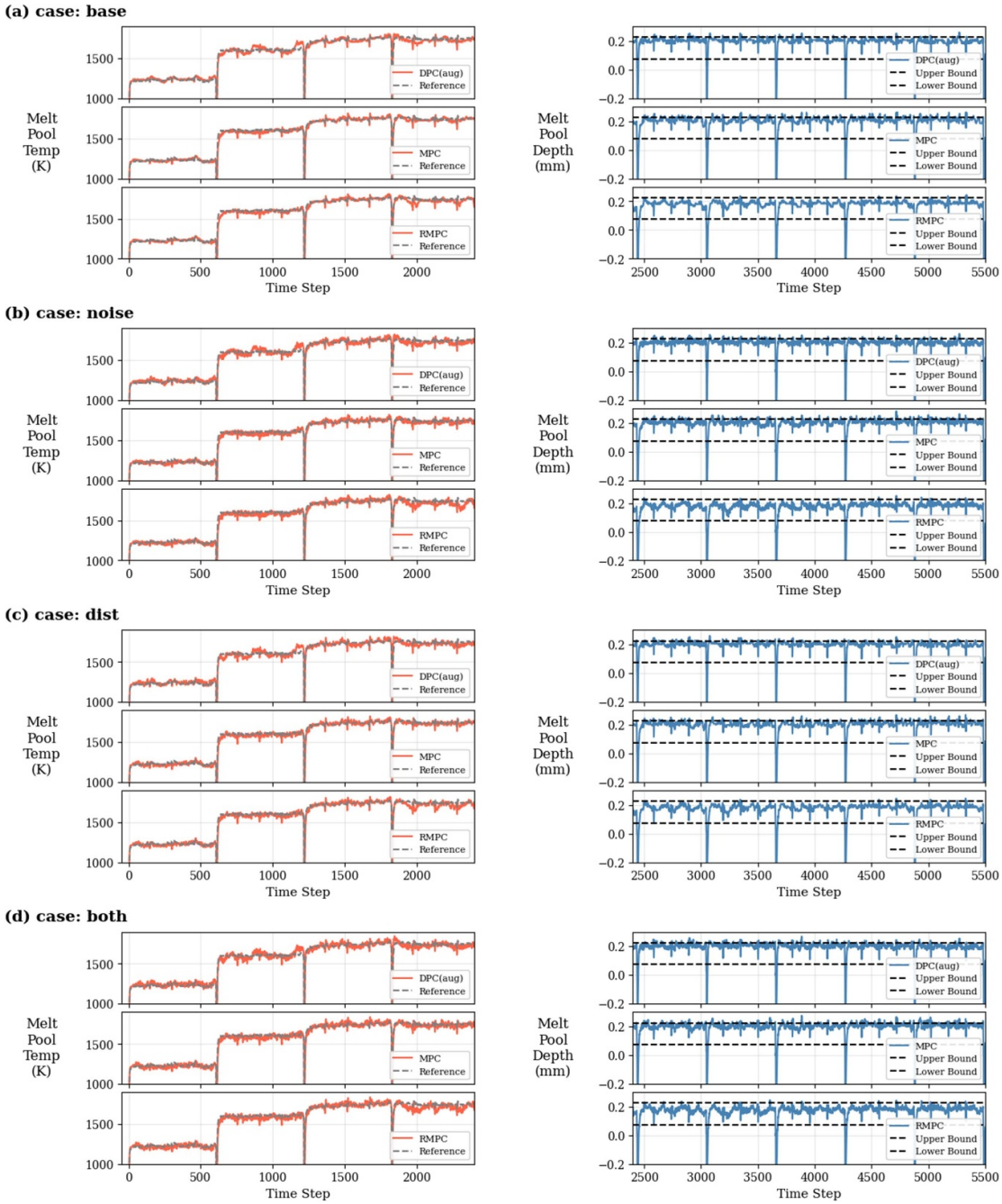


Figure B.16. Comparison of control performance between DPC, MPC, and RMPC under the four robustness test cases: nominal condition (`base`), sensor noise (`noise`), laser-power disturbance (`dist`), and combined sensor noise and laser-power disturbance (`both`). For each case, the left column shows melt pool temperature tracking, and the right column shows melt pool depth with the prescribed bounds.

ORCID iDs

Seul Lee 0009-0006-1820-5129
Yi-Ping Chen 0000-0002-9231-0860
Ying-Kuan Tsai 0009-0005-8853-2216
Vispi Karkaria 0000-0003-3733-2415
Wei Chen 0000-0002-4653-7124

References

[1] Lee J *et al* 2026 2026 roadmap on artificial intelligence and machine learning for smart manufacturing *Mach. Learn.: Eng.* **2**
[2] Glaessgen E and Stargel D 2012 The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conf. 20th AIAA/ASME/AHS Adaptive Structures Conf. 14th AIAA* (American Institute of Aeronautics and Astronautics) (https://doi.org/10.2514/6.2012-1818)
[3] Kritzinger W, Karner M, Traar G, Henjes J and Sihn W 2018 Digital Twin in manufacturing: a categorical literature review and classification *IFAC-PapersOnLine* **51** 1016–22
[4] Lu Y, Liu C, Wang K I-K, Huang H and Xu X 2020 Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues *Robot. Comput.-Integr. Manuf.* **61** 101837
[5] Karkaria V, Tsai Y-K, Chen Y-P and Chen W 2025 An optimization-centric review on integrating artificial intelligence and digital twin technologies in manufacturing *Eng. Optim.* **57** 161–207
[6] Van Beek A, Nevile Karkaria V and Chen W 2023 Digital twins for the designs of systems: a perspective *Struct. Multidiscip. Optim.* **66** 49
[7] Tsai Y-K, Karkaria V, Chen Y-P and Chen W 2025 Multi-generation control co-design for digital twin-enabled systems with deep reinforcement learning *Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conf.* vol 89220 (American Society of Mechanical Engineers) p V03AT03A006
[8] Chen Y-P, Suarez D, Tsai Y-K, Karkaria V, Hu G, Chen Z, Guo P, Cao J and Chen W 2026 Adaptive digital twin of sheet metal forming via proper orthogonal decomposition-based Koopman operator with model predictive control *J. Manuf. Syst.* **86** 762–73
[9] DebRoy T, Wei H L, Zuback J S, Mukherjee T, Elmer J W, Milewski J O, Beese A M, Wilson-Heid A, De A and Zhang W 2018 Additive manufacturing of metallic components – Process, structure and properties *Prog. Mater. Sci.* **92** 112–224
[10] Dass A and Moridi A 2019 State of the art in directed energy deposition: from additive manufacturing to materials design *Coatings* **9** 418
[11] Gunasegaram D, Barnard A, Matthews M, Jared B, Andreaco A, Bartsch K and Murphy A 2024 Machine learning-assisted in-situ adaptive strategies for the control of defects and anomalies in metal additive manufacturing *Addit. Manuf.* **81** 104013
[12] Su Y, Wang Z, Xu X, Luo K and Lu J 2022 Effect of closed-loop controlled melt pool width on microstructure and tensile property for Fe-Ni-Cr alloy in directed energy deposition *J. Manuf. Process.* **82** 708–21
[13] Smoqi Z, Bevans B D, Gaikwad A, Craig J, Abul-Haj A, Roeder B, Macy B, Shield J E and Rao P 2022 Closed-loop control of melt-pool temperature in directed energy deposition *Mater. Des.* **215** 110508
[14] Liao S, Webster S, Huang D, Council R, Ehmann K and Cao J 2022 Simulation-guided variable laser power design for melt pool depth control in directed energy deposition *Addit. Manuf.* **56** 102912
[15] Rawlings J B, Mayne D Q and Diehl M 2017 *Model Predictive Control: Theory, Computation and Design* 2nd edn (Nob Hill Publishing)
[16] Qin S J and Badgwell T A 2003 A survey of industrial model predictive control technology *Control Eng. Pract.* **11** 733–64
[17] Wang Y and Boyd S 2010 Fast model predictive control using online optimization *IEEE Trans. Control Syst. Technol.* **18** 267–78
[18] Gonzalez C, Asadi H, Kooijman L and Lim C P 2024 Neural networks for fast optimisation in model predictive control: a review (arXiv:2309.02668 [eess])
[19] Bemporad A, Borrelli F and Morari M 2002 Model predictive control based on linear programming - the explicit solution *IEEE Trans. Autom. Control* **47** 1974–85
[20] Alessio A and Bemporad A 2009 A Survey on Explicit Model Predictive Control *Nonlinear Model Predictive Control: Towards New Challenging Applications*, Ed L Magni, D M Raimondo and F Allgöwer (Springer) pp 345–69
[21] Jones C N and Morari M 2010 Polytopic approximation of explicit model predictive controllers *IEEE Trans. Autom. Control* **55** 2542–53
[22] Johansen T A 2004 Approximate explicit receding horizon control of constrained nonlinear systems *Automatica* **40** 293–300
[23] Tsai Y-K and Malak R J 2022 Design of approximate explicit model predictive controller using parametric optimization *J. Mech. Des.* **144** 124501
[24] Tsai Y-K and Malak Jr R J 2023 Control co-design with approximate explicit model predictive controllers *Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conf.* vol 87301 (American Society of Mechanical Engineers) p V03AT03A001
[25] Karg B and Lucia S 2020 Efficient representation and approximation of model predictive control laws via deep learning *IEEE Trans. Cybernetics* **50** 3866–78
[26] Drgoňa J, Picard D, Kvasnica M and Helsen L 2018 Approximate model predictive building control via machine learning *Appl. Energy* **218** 199–216
[27] Lucia S, Navarro D, Karg B, Sarnago H and Lucía Ó 2021 Deep learning-based model predictive control for resonant power converters *IEEE Trans. Ind. Inform.* **17** 409–20
[28] Tsai Y-K, Chen Y-P, Karkaria V and Chen W 2026 Reinforcement learning-based control co-design of digital twin-enabled full-vehicle active suspension systems *Struct. Multidiscip. Optim.* **69** 108
[29] Åkesson B M and Toivonen H T 2006 A neural network model predictive controller *J. Process Control* **16** 937–46
[30] Karg B and Lucia S 2021 Approximate moving horizon estimation and robust nonlinear model predictive control via deep learning *Comput. Chem. Eng.* **148** 107266
[31] Kiš K and Klaučo M 2019 Neural network based explicit MPC for chemical reactor control (arXiv:1912.04684 [cs])
[32] Löhr Y, Klaučo M, Fikar M and Mönnigmann M 2020 Machine learning assisted solutions of mixed integer MPC on embedded platforms *IFAC-PapersOnLine* **53** 5195–200
[33] Chan K J, Paulson J A and Mesbah A 2021 Deep learning-based approximate nonlinear model predictive control with offset-free tracking for embedded applications *2021 American Control Conf. (ACC)* (IEEE) pp 3475–81
[34] Drgoňa J, Tuor A, Skomski E, Vasisht S and Vrabie D 2021 Deep learning explicit differentiable predictive control laws for buildings *IFAC-PapersOnLine* **54** 14–19
[35] Drgoňa J, Kiš K, Tuor A, Vrabie D and Klaučo M 2022 Differentiable predictive control: deep learning alternative to explicit model predictive control for unknown nonlinear systems *J. Process Control* **116** 80–92
[36] King E, Drgona J, Tuor A, Abhyankar S, Bakker C, Bhattacharya A and Vrabie D 2022 Koopman-based differentiable predictive control for the dynamics-aware economic dispatch problem *2022 American Control Conf. (ACC)* (IEEE) pp 2194–201
[37] Cortez W S, Drgona J, Tuor A, Halappanavar M and Vrabie D 2022 Differentiable predictive control with safety guarantees: a control barrier function approach (arXiv:2208.02319 [eess])

[38] Viljoen J, Cortez W S, Drgona J, East S, Tomizuka M and Vrabie D 2024 Differentiable predictive control for robotics: a data-driven predictive safety filter approach (arXiv:2409.13817 [cs])
[39] Tumu R, Cortez W S, Drgoňa J, Vrabie D L and Glavaski S 2024 Differentiable predictive control for large-scale urban road networks (arXiv:2406.10433 [eess])
[40] Yoon J, Jarrett D and van der Schaar M 2019 Time-series generative adversarial networks *Neural Information Processing Systems* vol 32 (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper_files/paper/2019/hash/c9efe5f26cd17ba6216bbe2a7d26d490-Abstract.html)
[41] Somalwar A, Lee B D, Pappas G J and Matni N 2026 Statistical efficiency of single- and multi-step models for forecasting and control (arXiv:2603.23465 [eess])
[42] Das A, Kong W, Leach A, Mathur S, Sen R and Yu R 2024 Long-term forecasting with TiDE: time-series dense encoder, (arXiv:2304.08424 [stat])
[43] Ferreau H J, Bock H G and Diehl M 2008 An online active set strategy to overcome the limitations of explicit MPC *Int. J. Robust Nonlinear Control* **18** 816–30
[44] Stellato B, Banjac G, Goulart P, Bemporad A and Boyd S 2020 OSQP: an operator splitting solver for quadratic programs *Math. Program. Comput.* **12** 637–72
[45] Tsai Y-K and Malak R J 2023 Robust control co-design using tube-based model predictive control *2023 American Control Conf. (ACC)* pp 769–75
[46] Tsai Y-K and Malak Jr R J 2025 Control co-design with performance-robustness trade-off using tube-based stochastic model predictive control *ASME Lett. Dyn. Syst. Control* **5** 030903
[47] Tsai Y-K and Malak R J 2026 Parametric multi-objective optimization for simultaneous and nested control co-design formulations with tube-based model predictive controllers *J. Mech. Des.* 148 111701
[48] Diehl M, Ferreau H J and Haverbeke N 2009 Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation *Nonlinear Model Predictive Control: Towards New Challenging Applications*, ed L Magni, D M Raimondo and F Allgöwer (Springer) pp 391–417
[49] Grüne L and Pannek J 2017 Nonlinear Model Predictive Control *Nonlinear Model Predictive Control: Theory and Algorithms*, ed L Grüne and J Pannek (Springer) pp 45–69
[50] Chen Y-P, Karkaria V, Tsai Y-K, Rolark F, Quispe D, Gao R X, Cao J and Chen W 2025 Real-time decision-making for digital twin in additive manufacturing with model predictive control using time-series deep neural networks *J. Manuf. Syst.* **80** 412–24
[51] Liao S, Golgoon A, Mozaffar M and Cao J 2023 Efficient GPU-accelerated thermomechanical solver for residual stress prediction in additive manufacturing *Comput. Mech.* **71** 879–93
[52] Okuta R, Unno Y, Nishino D, Hido S and Loomis C 2017 Cupy: A numpy-compatible library for nvidia gpu calculations *Proc. Workshop on Machine Learning Systems (LearningSys) in The Thirty-First Annual Conf. on Neural Information Processing Systems (NIPS)* (available at: http://learningsys.org/nips17/assets/papers/paper_16.pdf)
[53] Chen Y-P, Tsai Y-K, Karkaria V and Chen W 2026 Uncertainty-aware digital twins: robust model predictive control using time-series deep quantile learning *J. Mech. Des.* **148** 021702
[54] Levine S, Kumar A, Tucker G and Fu J 2020 Offline reinforcement learning: tutorial, review, and perspectives on open problems (arXiv:2005.01643 [cs])
[55] Janner M, Fu J, Zhang M and Levine S 2019 When to trust your model: model-based policy optimization *Neural Information Processing Systems* vol 32 (Curran Associates, Inc.) (available at: https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html)
[56] Yu T, Thomas G, Yu L, Ermon S, Zou J Y, Levine S, Finn C and Ma T 2020 MOPO: model-based offline policy optimization *Neural Information Processing Systems* vol 33 (Curran Associates, Inc.) pp 14129–42 (available at: https://proceedings.neurips.cc/paper/2020/hash/a322852ce0df73e204b7e67cbbef0d0a-Abstract.html)
[57] Kingma D P and Ba J 2017 Adam: a method for stochastic optimization (arXiv:1412.6980 [cs])
[58] Ansari M J, Roccisano A, Arcondoulis E J G, Schulz C, Schläfer T and Hall C 2025 Advancements in in-situ monitoring technologies for detecting process-induced defects in the directed energy deposition process: a comprehensive review *Materials* **18** 4304
[59] Karkaria V, Goeckner A, Zha R, Chen J, Zhang J, Zhu Q, Cao J, Gao R X and Chen W 2024 Towards a digital twin framework in additive manufacturing: Machine learning and bayesian optimization for time series process optimization *J. Manuf. Syst.* **75** 322–32
[60] Loshchilov I and Hutter F 2019 Decoupled weight decay regularization (arXiv:1711.05101 [cs])