Learning What Matters: Task Tailored Dynamics Models through Differentiable MPC

Jan Węgrzynowski, Piotr Kicki, Grzegorz Czechmanowski, Krzysztof Walas

***Abstract*—In model-based control, dynamics models are typically trained by minimizing open-loop prediction errors uniformly across all states and time steps. However, due to finite model capacity, this approach often misallocates representational power, as not all prediction errors impact the downstream closed-loop task equally. In this paper, we propose a task-aware training methodology that weights multi-step prediction errors based on their analytical sensitivity approximation to the closed-loop task cost, extracted via differentiable MPC. Experimental results demonstrate that fine-tuning dynamics models with our sensitivity-weighted loss significantly improves closed-loop tracking performance compared to standard Mean Squared Error (MSE) or variance-based state standardization.**

***Index Terms*—Model Predictive Control, Dynamics Learning, Differentiable Optimization, System Identification**

I. Introduction

The deployment of optimization-based control, particularly Model Predictive Control (MPC), has seen significant success in highly dynamic robotic applications. Traditionally reliant on first-principles models, there is a growing trend to use high-capacity function approximators, such as neural networks, to capture complex, unmodeled phenomena like aerodynamic drag or complex friction interactions [1]–[4]. When these learned dynamics models are integrated into MPC frameworks, they can significantly enhance closed-loop performance [3], [5]–[7].

However, training these neural dynamics models presents a fundamental challenge. Conventionally, models are trained via regression to minimize the open-loop multi-step prediction error between the simulated trajectory and the ground truth [7], [8]. Standard loss functions, such as MSE, implicitly assume that an error in any state dimension at any future time step is equally detrimental. In practice, due to the limited computational resources available for real-time inference, the model's representational capacity is strictly bounded. Consequently, the model is forced to distribute its limited capacity evenly across all states. Yet, in the context of closed-loop control, this assumption rarely holds true: a small prediction error in a critical state (e.g., heading angle at high speeds) can severely degrade task performance, while a larger error in a less relevant state might be entirely compensated for by the controller's feedback loop.

The concept of aligning the dynamics model training objective with its downstream usage has been explored within the RL community. Termed "value-aware" model learning, these approaches adjust the model loss based on the gradient of the value function, encouraging the model to be accurate where it affects the policy's expected return [9]–[11]. Furthermore, techniques to shape the prediction loss to avoid irrelevant visual distractors have been successfully deployed in model-based RL [12]. Despite these advances in RL, weighting the training loss in the context of explicit optimization-based control, where the planner solves a constrained optimal control problem at every time step, remains largely unexplored.

In this work, we bridge this gap by learning dynamics models with a control-aware loss tailored for MPC. Rather than treating the controller as a black-box policy, we leverage the known, differentiable structure of the MPC optimization problem [13], [14]. By utilizing differentiable MPC, we can analytically compute the sensitivity of the planned control actions and ultimately, approximate the closed-loop task cost with respect to specific state prediction errors along the horizon. We utilize these sensitivities to construct a weighting tensor that explicitly penalizes prediction errors that would have a significant impact on the controller's performance.

The main contributions of this paper are:

1) a novel task-aligned training objective that utilizes differentiable MPC to extract sample-, state- and prediction length- dependent sensitivity weights, effectively teaching the model "what matters" for control, and
2) an empirical evaluation demonstrating that models trained with our sensitivity-weighted loss achieve superior closed-loop performance without requiring larger network architectures.

II. Method

In modern model-based control, it is common to use universal function approximators, such as neural networks, to learn the dynamics model which is then used in a predictive controller. Ultimately, the objective of learning a dynamics model is not merely to minimize open-loop prediction errors, but to maximize the closed-loop performance of the controller. Due to limited computational resources and the strict timing requirements of the control interval, the dynamics model's capacity is constrained. Conventional training methodologies typically evaluate model accuracy using uniformly weighted loss functions. However, this implicitly assumes that all state errors are equally detrimental to the task, which we believe is rarely true in complex robotic systems. To explicitly align model training with its downstream usage, we introduce a task-aware weighting formulation, i.e., we aim to make the

This work was supported by an internal PUT grant 0214/SBAD/0256 and DNMK: 0214/SBAD/0257. All authors are with the Institute of Robotics and Machine Intelligence, Poznan University of Technology, Poznan, Poland, and with IDEAS Research Institute, Warsaw, Poland jan.wegrzynowski@put.poznan.pl

dynamics model more accurate in states that are critical for controller performance, given the finite limit on model capacity.

Let $\hat{\mathbf{x}}_{t:t+t_p} \in \mathbb{R}^{t_p \times n}$ represent a predicted state trajectory of length t_p and state dimension n. This trajectory is rolled out from an initial true state x_t using a sequence of applied controls $\mathbf{u}_{t:t+t_p-1}$ and a parameterized dynamics model f_θ^{rk4} (utilizing Runge-Kutta 4 integration). Let $\mathbf{x}_{t:t+t_p}^{\text{true}}$ denote the corresponding ground-truth trajectory.

We formulate a sensitivity-weighted objective function for training the dynamics model as follows:

$$\mathcal{L}_{\text{sens}}(\theta) = \left\| S_{t:t+t_p} \odot \left(\hat{\mathbf{x}}_{t:t+t_p} - \mathbf{x}_{t:t+t_p}^{\text{true}} \right) \right\|_2^2 \tag{1}$$

where $S_{t:t+t_p} \in \mathbb{R}^{t_p \times n}$ is a prediction horizon length- and state-dependent weighting tensor, and $\odot$ denotes the element-wise (Hadamard) product.

The core problem formulation in this context lies in the optimal selection of S. Ideally, S should be designed such that the resulting training loss is proportional to the closed-loop performance of the controller operating with the learned model. Standard practices often default to naive selections for S, such as an all-ones tensor (standard MSE) or inverse variance scaling (state standardization).

Rather than using generic baselines, we propose structuring S to weight prediction errors across each **sample**, each **state**, and each **time step** of the prediction. This allows the model to focus its limited capacity on dynamic couplings that are significant for the control objective, while reducing the penalty for inaccuracies in states that are irrelevant to downstream performance.

III. Sensitivity of the Task Objective to Prediction Errors

To formalize the control-aware form of the weighting tensor $\mathbf{S}$ introduced in Section II, we must quantify how open-loop prediction errors impact the closed-loop performance of the system. We define the closed-loop performance via the total episodic task cost $J = \sum_{\tau=0}^{T} c(x_\tau, u_\tau)$, where $c(x, u)$ is the instantaneous cost.

Let $\mathbf{e}_{t:t+t_p} = \hat{\mathbf{x}}_{t:t+t_p} - \mathbf{x}_{t:t+t_p}^{\text{true}}$ denote the multi-step open-loop prediction error over the horizon at timestep t. Our objective is to derive the gradient of the total cost J with respect to this error trajectory for each sample. This is evaluated over trajectory tuples drawn from a dataset $\mathcal{D}$ consisting of ground-truth state and action sequences, denoted as $\mathcal{D} = \left\{ \left(\mathbf{x}_{i:i+t_p}^{\text{true}}, \mathbf{u}_{i:i+t_p} \right) \right\}_{i=1}^{N}$.

It is worth keeping in mind the open-loop nature of the model rollouts during training and the closed-loop nature of the controller during deployment. The controller optimizes a planned trajectory but only applies the immediate action u_t. At the subsequent timestep $t+1$, the system transitions to a new state based on the true dynamics. Subsequently the controller computes a fresh prediction horizon, around which a new set of errors $\mathbf{e}_{t+1:t+1+t_p}$ can occur. Consequently, the open-loop prediction error $\mathbf{e}_{t:t+t_p}$ influences overall task cost J entirely through its perturbation of the immediate action u_t.

Leveraging this insight, let us consider the total derivative of the task cost with respect to the prediction errors at time t to $t + t_p$:

$$\frac{\partial J}{\partial \mathbf{e}_{t:t+t_p}} = \frac{\partial c_t}{\partial \mathbf{e}_{t:t+t_p}} + \sum_{\tau=t+1}^{T^{task}} \frac{\partial c_\tau}{\partial \mathbf{e}_{t:t+t_p}}, \tag{2}$$

where c_t represents the step cost at time t that depends on state x_t and control u_t, and then by applying the chain rule, we derive the following expression

$$\frac{\partial J}{\partial \mathbf{e}_{t:t+t_p}} = \underbrace{\left[\frac{\partial c_t}{\partial u_t} + \sum_{\tau=t+1}^{T^{\text{task}}} \left(\frac{\partial c_\tau}{\partial x_\tau} \frac{\mathrm{d}x_\tau}{\mathrm{d}u_t} + \frac{\partial c_\tau}{\partial u_\tau} \frac{\mathrm{d}u_\tau}{\mathrm{d}u_t} \right) \right]}_{\substack{\text{Task Sensitivity} \\ \text{to Action } (\partial J / \partial u_t)}} \underbrace{\frac{\partial u_t}{\partial \mathbf{e}_{t:t+t_p}}}_{\substack{\text{Controller Sens.} \\ \text{to Pred. Error}}}. \tag{3}$$

This factorization decouples the problem into two distinct components: the sensitivity of the task objective to the applied action $\frac{\partial J}{\partial u_t}$, and the sensitivity of the controller to the underlying model inaccuracies $\frac{\partial u_t}{\partial \mathbf{e}_{t:t+t_p}}$. This establishes the theoretical ideal for our training weights S from Section II:

$$\mathbf{S}_{t:t+t_p} \propto \frac{\partial J}{\partial \mathbf{e}_{t:t+t_p}} = \frac{\partial J}{\partial u_t} \frac{\partial u_t}{\partial \mathbf{e}_{t:t+t_p}} \tag{4}$$

While this formulation provides a target for aligning model training with closed-loop performance, computing the exact analytical value of $\frac{\partial J}{\partial u_t}$ is intractable in practice. Specifically, computing the downstream task sensitivity requires evaluating the state-action Jacobians over time $\frac{\mathrm{d}x_\tau}{\mathrm{d}u_t}$. These terms are governed by the *true* underlying system dynamics, which are unknown and precisely what the parameterized model f_θ^{rk4} is attempting to learn. Therefore, we must rely on approximations of $\mathbf{S}_{t:t+t_p}$ to bridge this gap.

A. Updated Estimating S_t

Although we do not have access to the true analytical $\mathbf{S}$, we can use the finite MPC horizon as a proxy to evaluate how a change in the selected action will degrade the estimated MPC cost. Furthermore, through the use of differentiable MPC, we can establish how perturbations of states along the prediction horizon will influence the selected action. Therefore, we propose the following approximation of the true sensitivity matrix S:

$$\hat{S}_{t:t+t_p} = \left. \frac{\partial \hat{J}_t^{MPC}}{\partial u_0} \right|_{\text{eff}} \cdot \frac{\partial u_0}{\partial \mathbf{e}_{t:t+t_p}}. \tag{5}$$

This approximation consists of two parts:

1) **Approximated sensitivity of the action on the objective:** When the optimal control action u_0^* is active at a hard constraint, the sensitivity is strictly non-zero and can be extracted directly as the Lagrange multiplier of that constraint.
   However, when u_0^* is unconstrained, it lies at the local minimum of the objective function, meaning the exact first-order derivative evaluated at u_0^* is zero ($\nabla_{u_0} \hat{J}_t^{MPC} = 0$). Evaluating the sensitivity exactly at

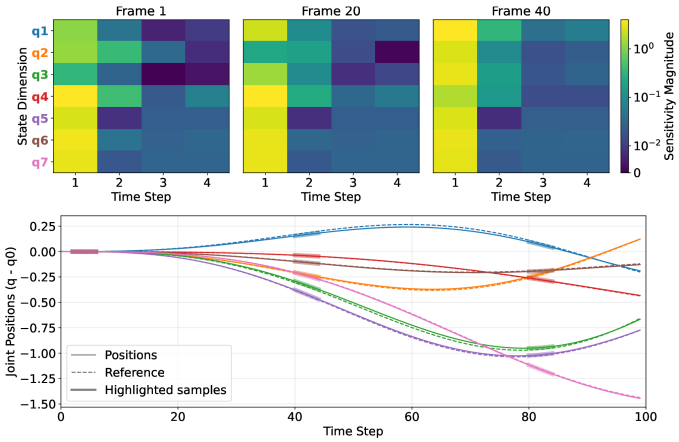

Fig. 1. The cost map shows the influence of errors in dynamics model across states and across prediction horizon on estimated task cost. Generated from provided pretrained CaDeLaC [15] model.

this point would erroneously yield a weight of zero, masking the true penalty incurred if prediction errors perturb the controller away from this optimal action.

To capture the local curvature, how severely the cost degrades as we move away from the optimal action, we define an *effective* sensitivity. This is formulated as the expected magnitude of the cost gradient subject to a small perturbation ξ around the optimal action:

$$\left.\frac{\partial \hat{J}_t^{MPC}}{\partial u_0}\right|_{\text{eff}} = \mathbb{E}_\xi \left[\left| \left.\frac{\partial \hat{J}_t^{MPC}}{\partial u_0}\right|_{u_0^*+\xi} \right| \right]. \tag{6}$$

In practice, assuming ξ is drawn from a discrete uniform distribution $\xi \in \{-\epsilon, \epsilon\}$, we approximate this expectation by querying the sensitivities at $u_{\text{test}} = u_0^* \pm \epsilon$ and averaging the slopes. By setting u_{test} as an equality constraint within the differentiable MPC solver, we can extract the corresponding gradients and average their absolute magnitudes.

2) **Sensitivity of the controller to the dynamics model errors** $\frac{\partial u_0}{\partial \mathbf{e}_{t:t+t_p}}$. This sensitivity can be obtained through a differentiable implementation of MPC. To do so, we evaluate the sensitivities along the prediction horizon.

Issues and Assumptions:

1) MPC objective must be similar to the task objective, i.e., the optimal control problem must be well formulated.
2) Sensitivities are evaluated around the predicted trajectory, not the executed trajectory, which is used as a reference for model training. Fortunately, if the model used for MPC is reasonably accurate, the predicted and executed trajectories are well aligned. In the limit, as the prediction error approaches zero, the predicted and executed trajectories match exactly. However, this may not be an issue if the sensitivities decay faster than the point at which the actual and predicted trajectories diverge.

IV. Experimental Results

In our experimental analysis, we aim to evaluate whether the proposed weighting improves closed-loop control performance. To do so, we consider a trajectory tracking task for the 7DoF Franka Panda manipulator with different payloads, following the setup introduced in [15]. As our base architecture, we employ Context-Aware Deep Lagrangian Model (CaDeLaC) [15], which augments a nominal physics model with a history-conditioned residual network to capture unmodeled dynamics. All experiments start from the same pretrained CaDeLaC dynamics model and fine-tune it on a dataset collected from closed-loop MPC rollouts while tracking randomized reference trajectories in MuJoCo [16] simulation. Each sample contains the measured joint positions and velocities, the applied MPC torques, and the sensitivity terms were calculated using acados [14], [17] MPC implementation using the provided pretrained CaDeLaC model. The learned dynamics state is $x = [q, \dot{q}] \in \mathbb{R}^{14}$ and the control input is the 7-dimensional joint torque vector.

Figure 1 visualizes the estimated cost sensitivity over state dimensions and over the prediction horizon. The map is strongly non-uniform, with early prediction steps and a subset of joint coordinates dominating the downstream cost, which motivates the potential usefulness of control-aware dynamics model loss weighting.

A. Training Setup

We fine-tune the pretrained model with multi-step rollout losses of length $p \in \{1, 4, 12\}$ using the task-aligned objective in (1). In the experiments reported here, we train primarily with a position objective, meaning that the loss is calculated based only on joint positions. We also evaluated variants trained on both positions and velocities, as well as velocities alone, but the position objective gave the best closed-loop tracking performance.

The weighting term S_{t+k} determines which components of the prediction error are emphasized during training. We compare the following weighting strategies: *uniform* (ones), *state standardization* (stand), *range normalization* (norm), *MPC stage-cost based* (mpc_cost), and our *task-aligned sensitivity* derived from the MPC sensitivities. Among the sensitivity-based variants, we evaluate both standardized sensitivities (ours) and an averaged standardized sensitivity map over the dataset (ours_avg).

All compared models use the same architecture, optimizer, and initialization. Thus, differences in performance can be attributed to the loss weighting strategy rather than model capacity or controller design. The purpose of varying the rollout length is to test whether the benefit of task-aligned weighting becomes more pronounced as training shifts from one-step prediction to longer-horizon model accuracy.

B. Controller Evaluation

After fine-tuning, each model is deployed inside the same MPC controller without changing the controller cost, horizon, or constraints. We evaluate on 20 randomized environments in which mass and position of the payload are sampled as in [15], and report the average closed-loop tracking cost over a 10 s episode. The controller runs at 50 Hz, so this metric reflects the accumulated tracking quality over the full task.

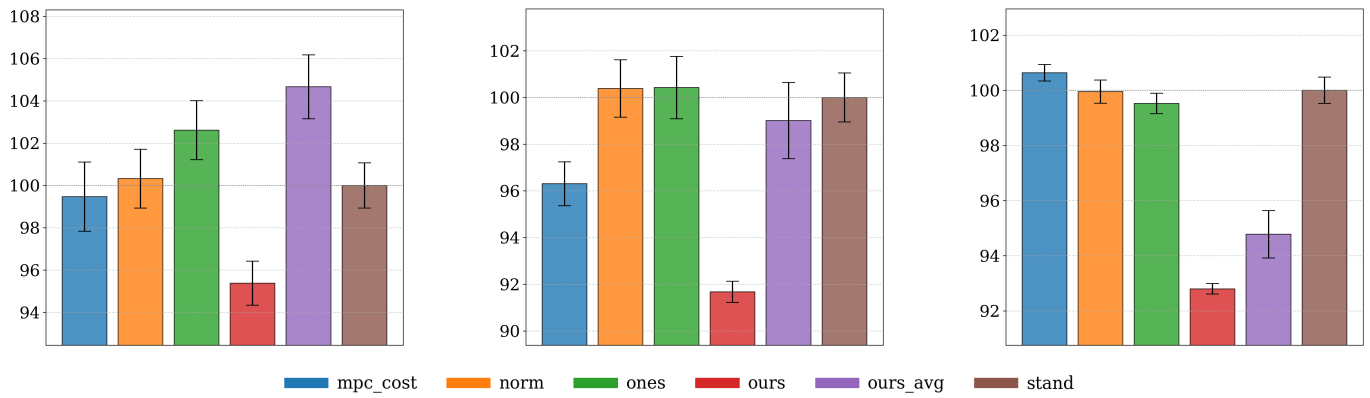

Fig. 2. Closed-loop tracking cost comparison. Each bar reports the mean and standard deviation over the five runs for a given weighting strategy, normalized relative to the `stand` baseline and expressed in percent. From left to right, the rollout lengths are $p = 1$, $p = 4$, and $p = 12$. The sensitivity-based weighting `ours` becomes more advantageous as the rollout horizon increases.

Our primary performance metric is the weighted tracking cost induced by the MPC objective,

$$J_{\text{track}} = \frac{1}{T^{task}} \sum_{t=0}^{T^{task}} \big[(q_t - q_t^{\text{ref}})^\top Q_q (q_t - q_t^{\text{ref}}) + (\dot{q}_t - \dot{q}_t^{\text{ref}})^\top Q_{\dot{q}} (\dot{q}_t - \dot{q}_t^{\text{ref}})\big], \tag{7}$$

computed from the realized closed-loop trajectories. For each weighting strategy, the reported bar plots show the top five runs across fine-tuning epochs, and the mean cost is normalized relative to the standardization baseline. Figure 2 summarizes these comparisons across rollout lengths $p \in \{1, 4, 12\}$.

The quantitative results in Figure 2 confirm our hypothesis: employing control-aware weighting during dynamics model training improves the closed-loop performance of the predictive controller. Crucially, our proposed sensitivity-weighted approach (`ours`) outperforms all evaluated baselines across every tested rollout length during training.

Even for a single-step rollout ($p = 1$), `ours` establishes a clear lead, yielding the best result and reducing the tracking cost to 95.4% of the `stand` baseline. Importantly, as the rollout horizon increases, the performance gap between our method and the baselines widens substantially. While generic weighting strategies such as `ones`, `norm`, and `mpc_cost` stagnate near 100%, `ours` dominates the longer rollouts, further reducing the normalized cost to 91.7% at $p = 4$ and 92.8% at $p = 12$.

Moreover, we performed an ablation study by averaging the sensitivities over the dataset (`ours_avg`), which resulted in a significant performance drop, proving that local, sample-dependent sensitivity is key to allocating model capacity effectively. This suggests that the main benefit comes from using accurate sample-dependent sensitivities, not just from emphasizing the same states or imposing a generic decay along the prediction horizon. Overall, the gains grow with rollout length, indicating that per-step sensitivity weighting is especially useful when training must preserve long-horizon control-relevant accuracy in samples where it is necessary.

V. Conclusion

The proposed sensitivity-weighted loss function demonstrates the benefit of aligning model training with the ultimate goal of closed-loop optimal control. By leveraging the differentiable MPC framework to extract analytical sensitivities, we showed that weighting multi-step prediction errors by their downstream task impact significantly improves closed-loop performance over generic statistical normalizations.

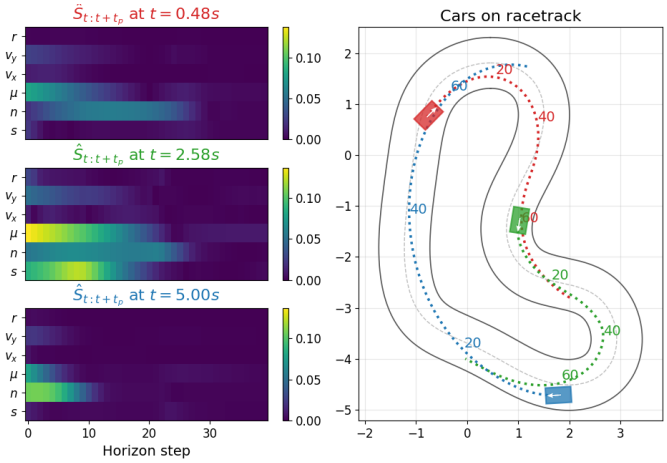

Fig. 3. Example sensitivities for a real-world F1Tenth autonomous racing task. Left: Sensitivity maps for the dynamic states at different points in time. Right: Corresponding car positions on the track. The sensitivity of the task objective to individual states changes heavily depending on the current track position and corresponding velocities.

VI. Future Work

As future work, we plan to extend this approach to other robotic domains with highly dynamic environments, such as autonomous racing. Figure 3 provides an initial outlook in this direction, illustrating sensitivity evaluations for an autonomous racing vehicle on a track. The importance of predicting specific state dimensions varies dramatically over time; for instance, prediction errors in lateral deviation (n) and track relative heading (μ) are heavily penalized when entering a sharp curve, whereas different states are prominent ahead of straights. Because the importance of specific states is highly contextual, applying our sensitivity-weighted loss represents a promising avenue for advancing high-speed closed-loop control of racing vehicles.

REFERENCES

[1] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," vol. 4, no. 28, p. eaaw1975. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.aaw1975

[2] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-driven MPC for quadrotors." [Online]. Available: http://arxiv.org/abs/2102.05773

[3] K. Y. Chee, T. Z. Jiahao, and M. A. Hsieh, "KNODE-MPC: A knowledge-based data-driven predictive control framework for aerial robots." [Online]. Available: http://arxiv.org/abs/2109.04821

[4] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural-MPC: Deep learning model predictive control for quadrotors and agile robotic platforms," vol. 8, no. 4, pp. 2397–2404. [Online]. Available: http://arxiv.org/abs/2203.07747

[5] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," vol. 4, no. 4, pp. 3758–3765, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/document/8768048/?arnumber=8768048

[6] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with gaussian process dynamics for autonomous miniature race cars," in *2018 European Control Conference (ECC)*. IEEE, pp. 1341–1348. [Online]. Available: https://ieeexplore.ieee.org/document/8550162/

[7] J. Węgrzynowski, P. Kicki, G. Czechmanowski, M. P. Krupka, and K. Walas, "Beyond constant parameters: Hyper prediction models and hypermpc," in *Proceedings of The 9th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, 27–30 Sep 2025, pp. 4885–4907. [Online]. Available: https://proceedings.mlr.press/v305/wegrzynowski25a.html

[8] Y. Tsuchiya, T. Balch, P. Drews, and G. Rosman, "Online adaptation of learned vehicle dynamics model with meta-learning approach," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 802–809.

[9] A.-m. Farahmand, A. M. S. Barreto, and D. N. Nikovski, "Value-aware loss function for model-based reinforcement learning."

[10] C. Voelcker, V. Liao, A. Garg, and A.-m. Farahmand, "Value gradient weighted model-based reinforcement learning." [Online]. Available: http://arxiv.org/abs/2204.01464

[11] K. Asadi, D. Misra, S. Kim, and M. L. Littman, "Combating the compounding-error problem with a multi-step model." [Online]. Available: http://arxiv.org/abs/1905.13320

[12] M. Hutson, I. Kauvar, and N. Haber, "Policy-shaped prediction: avoiding distractions in model-based reinforcement learning."

[13] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control." [Online]. Available: http://arxiv.org/abs/1810.13400

[14] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[15] L. Schulze, J. Peters, and O. Arenz, "Context-aware deep lagrangian networks for model predictive control," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 6939–6946.

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

[17] J. Frey, J. De Schutter, and M. Diehl, "Fast integrators with sensitivity propagation for use in CasADi," 2023.