2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Adaptive Stochastic Nonlinear Model Predictive Control with Look-ahead Deep Reinforcement Learning for Autonomous Vehicle Motion Control

Baha Zarrouki[1,2], Chenyang Wang[2] and Johannes Betz[2]

***Abstract*— Propagating uncertainties through nonlinear system dynamics in the context of Stochastic Nonlinear Model Predictive Control (SNMPC) is challenging, especially for high-dimensional systems requiring real-time control and operating under time-variant uncertainties such as autonomous vehicles. In this work, we propose an Adaptive SNMPC (aSNMPC) driven by Deep Reinforcement Learning (DRL) to optimize uncertainty handling, constraints robustification, feasibility, and closed-loop performance. To this end, our SNMPC uses Polynomial Chaos Expansion (PCE) for efficient uncertainty propagation, limits its propagation time through an Uncertainty Propagation Horizon (UPH), and transforms nonlinear chance constraints into robustified deterministic ones. We conceive a DRL agent to proactively anticipate upcoming control tasks and to dynamically reduce conservatism by determining the most suitable constraints robustification factor κ, and to enhance feasibility by choosing optimal UPH length T_u. We analyze the trained DRL agent's decision-making process and highlight its ability to learn context-dependent optimal parameters. We showcase the enhanced robustness and feasibility of our DRL-driven aSNMPC through the real-time motion control task of an autonomous passenger vehicle when confronted with significant time-variant disturbances while achieving a minimum solution frequency of 110Hz. The code used in this research is publicly accessible as open-source software: https://github.com/bzarr/TUM-CONTROL**

I. Introduction

One approach to deal with poor closed-loop performance caused by uncertainties in Model Predictive Control (MPC) algorithms is to consider uncertainties in the MPC design and to guarantee the state and control constraints' satisfaction with a predefined probability, a concept known as Stochastic MPC [1]. Nevertheless, achieving real-time behavior with Stochastic Nonlinear Model Predictive Control (SNMPC) algorithms presents significant challenges. This stems from the difficulty in handling uncertainty propagation within nonlinear systems, frequently resulting in feasibility problems and a substantial computational overhead.

Various methods have been explored to address this issue. Works [2] and [3] employ Gaussian Regression for uncertainty propagation, which is computationally intensive and impractical for real-time systems. As in works [4], [5], [6], our SNMPC approach employs Polynomial Chaos Expansion (PCE) techniques for effectively propagating uncertainties while accounting for nonlinear hard constraints on state expectations and probabilistic constraints. PCE serves as a highly efficient substitute for Monte Carlo simulations in the propagation of probabilistic uncertainties. To prevent infeasibility caused by uncertainty propagation divergence, we employ an Uncertainty Propagation Horizon (UPH) concept that limits the time for propagation of uncertainties through system dynamics.

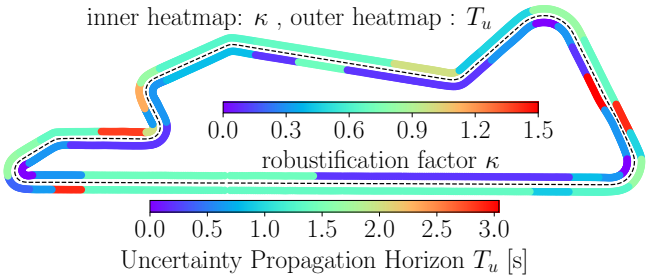

Fig. 1. Deep Reinforcement Learning agent's optimal decisions adapting Stochastic NMPC parameters κ and T_u online for the motion control task of an autonomous passenger vehicle on Monteblanco track (driving clockwise). The agent alters both parameters context-dependent to reduce conservatism, enhance feasibility and improve closed-loop performance.

However, this SNMPC approach remains sensitive to the length of the UPH and to the variance sensitivity factor κ of the robustified nonlinear constraints to uncertainties which may lead to conservatism and poor closed-loop performance. In this paper, we propose to learn both parameters κ and T_u and automatically adapt them online with state-of-the-art Deep Reinforcement Learning (RL) techniques. Based on the current closed-loop performance assessment and future control reference trajectories, we design the RL agent to determine context-dependent optimal SNMPC parameters proactively. This promises a better closed-loop performance and an increased feasibility of dealing with a wide range of uncertainties and different dynamic situations.

Recent advancements have enabled RL agents [7] to effectively handle complex systems and to achieve performance levels comparable to human experts, not only in simulated environments but also in real-world applications [8][9]. RL algorithms push the agent to interact with its environment actively, facilitating the learning of a policy that optimizes a desired behavior by maximizing the rewards received as feedback. Several works combine MPC with learning [10][11]. The authors of [12] and [13] employ MPC as a

[1] Chair of Automotive Technology, Technical University of Munich
[2] Professorship of Autonomous Vehicle Systems, TUM School of Engineering and Design, Technical University of Munich, 85748 Garching, Germany; Munich Institute of Robotics and Machine Intelligence (MIRMI), {baha.zarrouki, 16chenyang.wang, johannes.betz}@tum.de

This work is a result of the joint research project ATLAS-L4, supported by the German Federal Ministry for Economic Affairs and Climate Action.

[*] Code: https://github.com/bzarr/TUM-CONTROL

function approximator within RL. This approach ensures that the policy satisfies state and input constraints while meeting safety requirements. Other learning-based MPC approaches use Bayesian Optimization [14] [15] or Gaussian Process Regression [16] [17]. The authors of [18] learn to adapt system dynamics from data collected during system operation. A sample-based learning MPC is proposed in [19] to approximate safe sets and the value function from historical data. While RL is utilized as an end-to-end controller [20] [21], MPC is introduced as a safety filter for an RL-based controller [22] [23]. Furthermore, RL can also be used to learn controller design. Work [24] used RL algorithms to determine the prediction horizon of MPC to decrease computational complexity. Work [25] extends the previous work to learn other MPC meta-parameters that are non-differentiable wrt. the MPC outputs. Meanwhile, works [26] introduce a Deep RL driven Weights-varying NMPC to automatically learn and online adapt the cost function weighting matrices optimizing multi-control objectives.

One traditional adaptive SMPC [27][28] uses an adaptive law to iteratively update the chance constraints using a scaling factor that is proportional to the deviation of the empirical probability of violation from a desired violation level. The authors of [29] adjust individual chance constraints based on the empirical cumulative distribution function updated by online additive disturbance information, aiming for reduced conservatism. Prior research has predominantly concentrated on parameter adaptation in control systems to align with static reference profiles based on historical data. In contrast, our approach highlights the adaptation of SNMPC parameters to accommodate dynamic control references that change over time. In motion control use-case, work [30] adapts linear SMPC risk parameter using a cumulative distribution function and based on the current error between measured and predicted state without look-ahead capabilities.

In summary, our work presents three main contributions:

1) We conceive an Adaptive SNMPC (aSNMPC) framework employing a look-ahead RL design. This agent autonomously adapts the SNMPC-specific parameters under varying uncertainties.
2) We provide evidence of the increased robustness and feasibility of our aSNMPC compared to a standard SNMPC, particularly in the presence of fluctuating external disturbances. We conduct a comparative analysis of the effect of disturbance assumptions and illustrate the generalization capabilities of our aSNMPC when operating in previously unseen environments.
3) We assess the impact of learning and adapting each parameter on SNMPC performance, and we conduct a context-dependent analysis of the RL agent's decision-making process.

II. A GENERAL SNMPC BASED ON POLYNOMIAL CHAOS EXPANSION (PCE) AND AN UNCERTAINTY PROPAGATION HORIZON (UPH)

We use the following notation: Given a variable $z \in \mathbb{R}$, we define $\boldsymbol{z} = \boldsymbol{z}^{(\cdot)} = [z_0, z_1, \ldots, z_n]^T \in \mathbb{R}^n$ as the vector composed of z variables. Additionally, $\boldsymbol{Z} = [\boldsymbol{z}^{(0)}, \boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}] \in \mathbb{R}^{n \times m}$ represents the matrix consisting of concatenated $\boldsymbol{z}$ vectors.

In this work, we consider the following general Stochastic Nonlinear MPC (SNMPC) that aims to solve a stochastic OCP subject to probabilistic constraints, offering an efficient method to transform chance constraints into robust deterministic constraints. To deal with the difficulty in propagating stochastic uncertainties through nonlinear dynamics, we use Polynomial Chaos Expansion (PCE) [4] [5]. PCE is an efficient alternative to Monte Carlo sampling methods. However, uncertainty propagation can cause divergence of uncertain states' evolution and too tightened constraints that lead to infeasibility. To address this, we propose an Uncertainty Propagation Horizon (UPH) [6] that limits the time for uncertainty propagation within the prediction horizon.

Problem 1 general Stochastic Nonlinear MPC

$$
\begin{aligned}
\min_{\substack{\boldsymbol{x}(.),\\ \boldsymbol{u}(.)}} \quad & \int_{\tau=0}^{T_p} l(\mathbb{E}[\boldsymbol{x}(\tau)], \boldsymbol{u}(\tau)) d\tau \\
& \quad + m(\mathbb{E}[\boldsymbol{x}(T_p)]) \\
\text{s. t.} \quad & \boldsymbol{x}(0) = \boldsymbol{x}_0, \\
& \dot{\boldsymbol{x}}(t) = f(\mathbb{E}[\boldsymbol{x}(t)], \boldsymbol{u}(t)), && t \in [0, T_p), \\
& \text{—transformed chance constraints} \\
& \mathbb{E}[h(\boldsymbol{x}, \boldsymbol{u})] + \kappa \sqrt{\mathrm{Var}[h(\boldsymbol{x}, \boldsymbol{u})]} \leq \bar{\boldsymbol{h}}, && t \in [0, T_p), \\
& \mathbb{E}[h^{\mathrm{e}}(\boldsymbol{x}(T_p))] \\
& \quad + \kappa \sqrt{\mathrm{Var}[h^{\mathrm{e}}(\boldsymbol{x}(T_p))]} \leq \bar{\boldsymbol{h}}^{\mathrm{e}}, \\
& \text{—hard nonlinear constraints} \\
& g(\mathbb{E}[\boldsymbol{x}(t)], \boldsymbol{u}(t)) \leq \bar{\boldsymbol{g}}, && t \in [0, T_p), \\
& g^{\mathrm{e}}(\mathbb{E}[\boldsymbol{x}(T_p)]) \leq \bar{\boldsymbol{g}}^{\mathrm{e}}, \\
& \text{—hard linear constraints} \\
& J_{\mathrm{bx}} \mathbb{E}[\boldsymbol{x}(t)] \leq \bar{\boldsymbol{x}}, && t \in [0, T_p), \\
& J_{\mathrm{bx}}^{\mathrm{e}} \mathbb{E}[\boldsymbol{x}(T_p)] \leq \bar{\boldsymbol{x}}^{\mathrm{e}} \\
& J_{\mathrm{bu}} \boldsymbol{u}(t) \leq \bar{\boldsymbol{u}}, && t \in [0, T_p),
\end{aligned}
\tag{1}
$$

Here, $\boldsymbol{x} \in \mathbb{R}^{n_x}$ represents the state vector, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ represents the control vector, T_p stands for the prediction horizon and f denotes the system dynamics. The initial state is denoted as $\boldsymbol{x_0}$. Additionally, $l : \mathbb{R}^{n_{\mathrm{x}}} \times \mathbb{R}^{n_{\mathrm{u}}} \to \mathbb{R}$ defines the stage cost, while $m : \mathbb{R}^{n_{\mathrm{x}}} \to \mathbb{R}$ defines the terminal cost. Problem 1 incorporates hard linear constraints on the states' expectations and on the control inputs formulated with help of J_{bx}, J_{bx}^{e} and J_{bu}. Furthermore, it handles hard nonlinear constraints on the states' expectations: g and g^{e}. Problem 1 transforms nonlinear probabilistic inequality constraints into estimated deterministic surrogates in expectation and variance of the nominal path and terminal nonlinear inequality constraints: h and h^{e}. Here, $\kappa = \sqrt{(1-p)/p}$ denotes the constraints robustification factor, meant to tighten the constraints according to the variance of the nonlinear constraints affected by uncertainties, where $p \in (0, 1]$ is the desired probability of violating the nonlinear constraints h.

To predict the temporal evolution of probability distribution moments for stochastic variables, PCE relies on a series of multivariate orthogonal polynomials. The coefficients of the orthogonal polynomials, i.e. PCE coefficients, are computed using a solution of a least-squares regression problem and several samples of uncertain variables, taking into account assumed standard deviations. In our case, we consider propagating n_s sampled points around the current measured state to account for uncertainties. As we account for uncertainties in system states as well as in nonlinear constraints, we compute the following PCE coefficients:

1) $\boldsymbol{C}^{(\boldsymbol{x})} = A \cdot \tilde{\boldsymbol{X}}_{t+1}$ are coefficients for uncertain states and $\tilde{\boldsymbol{X}} \in \mathbb{R}^{n_s} \times \mathbb{R}^{n_x}$ contains n_s generated state samples $\tilde{\boldsymbol{x}}_i$
2) $\boldsymbol{c}^{(\boldsymbol{h})} = A \cdot \tilde{\boldsymbol{h}}(\tilde{\boldsymbol{X}}, \boldsymbol{u})$ are coefficients for nonlinear chance constraints, with $\tilde{\boldsymbol{h}}(\tilde{\boldsymbol{X}}, \boldsymbol{u}) \in \mathbb{R}^{n_s}$ a vector containing the inequality constraint applied to the individual state samples $\tilde{\boldsymbol{x}}_i$

Here, the matrix $A \in \mathbb{R}^L \times \mathbb{R}^{n_s}$ is the solution of the l_2-norm least-squares regression problem:

$$\min_{\boldsymbol{C}^{(\boldsymbol{x})}} \quad ||\tilde{\boldsymbol{X}} - \tilde{\boldsymbol{\Phi}} \boldsymbol{C}^{(\boldsymbol{x})}||_2^2 \tag{2}$$

with $\tilde{\boldsymbol{\Phi}} \in \mathbb{R}^{n_s} \times \mathbb{R}^L$ containing the multivariate polynomials evaluated at each disturbance sample $\tilde{\boldsymbol{w}}^{(i)}, \forall i \in \{1, \dots, n_s\}$, and L is the total number of the PCE terms. We sample the disturbance/uncertainty realizations using Hammersley low-discrepancy sequence, considering the assumed standard deviations $\boldsymbol{\sigma}_w$.

The propagation of uncertain state samples and constraints through T_p is limited by Uncertainty Propagation Horizon (UPH): T_u. After reaching the UPH, the propagation of the samples is stopped and only the last variables' expectations at $t = T_u$ are propagated until T_p [6].

The following equation determines the evolution of the uncertain states' realizations:

$$\tilde{\boldsymbol{X}}_{t+1} = \begin{cases} f(\tilde{\boldsymbol{X}}_t, \boldsymbol{u}), & \text{if } t \in \{0, ..., N_{u-1}\} \\ \boldsymbol{O}_{n_s \times n_x}, & \text{if } t \in \{N_u, ..., N_{p-1}\} \end{cases} \tag{3}$$

Here, N_u and N_p denote the number of shooting nodes within the UPH and prediction horizon respectively. We compute the expectations and variances in Problem 1 as [6]:

$$\begin{cases} \mathbb{E}[\boldsymbol{x}_t] = \boldsymbol{c}_0^{(\boldsymbol{x})} \\ \mathbb{E}[h(\boldsymbol{x}, \boldsymbol{u})] = c_0^{(h)} \\ \text{Var}[h(\boldsymbol{x}, \boldsymbol{u})] = \sum_{k=1}^{L-1} (c_k^{(h)})^2 \\ \\ \mathbb{E}[\boldsymbol{x}_t] = \boldsymbol{x}_t = f(\mathbb{E}[\boldsymbol{x}_{t-1}], \boldsymbol{u}) \\ \mathbb{E}[h(\boldsymbol{x}, \boldsymbol{u})] = h(\mathbb{E}[\boldsymbol{x}_t], \boldsymbol{u}) \\ \text{Var}[h(\boldsymbol{x}, \boldsymbol{u})] = 0 \end{cases} \begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \\ \\ \text{, if } t \in \{0, ..., N_{u-1}\} \\ \\ \\ \\ \\ \text{, if } t \in \{N_u, ..., N_{p-1}\} \end{array} \tag{4}$$

Determining suitable UPH length and robustification factor κ is challenging, especially when the system faces changing environments, control tasks, or disturbance magnitude. In this work, we design the SNMPC framework to automatically adapt, context-dependent, its robustification factor κ and UPH length T_u with DRL.

III. Example Application: Stochastic NMPC for Trajectory Following of Autonomous Vehicles

We apply the general SNMPC (Problem 1) for the task of the longitudinal and lateral motion control of our Volkswagen T7 Multivan autonomous research passenger vehicle [31] to follow a given trajectory subjected to state estimation uncertainties: $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}) + \boldsymbol{w}_t$. Here, $\boldsymbol{w}_t$ is the disturbance and $\boldsymbol{x} = [x_{\text{pos}}, y_{\text{pos}}, \psi, v_{\text{lon}}, v_{\text{lat}}, \dot{\psi}, \delta_f, a]^T$ is the state vector with the yaw angle ψ, the yaw rate $\dot{\psi}$, the steering angle at the front wheel δ_f and the acceleration a. The control vector $\boldsymbol{u} = [j, \omega_f]^T$, where j is the longitudinal jerk, and ω_f is the steering rate. We adopt a dynamic nonlinear single-track model as our prediction model. We refer to [6] for the full dynamics definition. We define the stage- and terminal costs as $l(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{y}_{\text{ref}}\|_W^2$ and $m(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y}^e(\boldsymbol{x}) - \boldsymbol{y}_{\text{ref}}^e\|_{W^e}^2$ with W and W_e being the weighting matrices.

The system is subject to combined longitudinal and lateral acceleration limits: that we formulate using the transformed nonlinear chance constraints, i.e. robustified constraints:

$$\begin{aligned} &\mathbb{E}[h(\boldsymbol{x}, \boldsymbol{u})] + \kappa \sqrt{\text{Var}[h(\boldsymbol{x}, \boldsymbol{u})]} \leq 1 \\ &h(\boldsymbol{x}, \boldsymbol{u}) = \left(\frac{a_{\text{lon}}}{a_{x_{\max}}}\right)^2 + \left(\frac{a_{\text{lat}}}{a_{y_{\max}}}\right)^2 \end{aligned} \tag{5}$$

Where $a_{\text{lon}} = a$ denotes the longitudinal acceleration, and $a_{\text{lat}} = v_{\text{lon}}\dot{\psi}$ the lateral acceleration. The maximum allowed value for the lateral acceleration $a_{y_{\max}}$ is 5.866m/s^2. The maximum value for the longitudinal acceleration $a_{x_{\max}}$ is adapted based on the current vehicle velocity and ranges in $[-4.5\text{m/s}^2, 3\text{m/s}^2]$ Note that these limits are defined based on the vehicle interface, i.e. these limits are not the limits of the dynamics. Furthermore, we formulate linear hard constraints on the steering angle expectation δ_f and on the steering rate control input at the front wheel ω_f:

$$\begin{aligned} &|\mathbb{E}[\delta_f]| \leq 0.61\text{rad} \\ &|\omega_f| \leq 0.322\text{rad/s} \end{aligned} \tag{6}$$

We assume that the predicted states v_{lon}, v_{lat} and $\dot{\psi}$ are subject to uncertainties. We assume that the uncertainties are Gaussian disturbances and the standard deviations are known, such as:

$$\boldsymbol{\sigma}_w^{\text{SNMPC}} = [\sigma_{\text{vlon}}, \sigma_{\text{vlat}}, \sigma_{\dot{\psi}}]^T = [\sigma_{\text{vlon}}^{\text{sim}}, \sigma_{\text{vlat}}^{\text{sim}}, \sigma_{\dot{\psi}}^{\text{sim}}]^T \tag{7}$$

IV. Learning SNMPC parameters with Look-ahead Deep Reinforcement Learning

Our Adaptive SNMPC (aSNMPC) automatically adjusts the nonlinear constraints robustification factor κ and the UPH length T_u according to changing environments and driving tasks. We leverage a Deep Neural Network (DNN) policy to dynamically tune these parameters online. This policy is trained offline in the simulation through state-of-the-art Deep RL (DRL) algorithms (Figure 2).

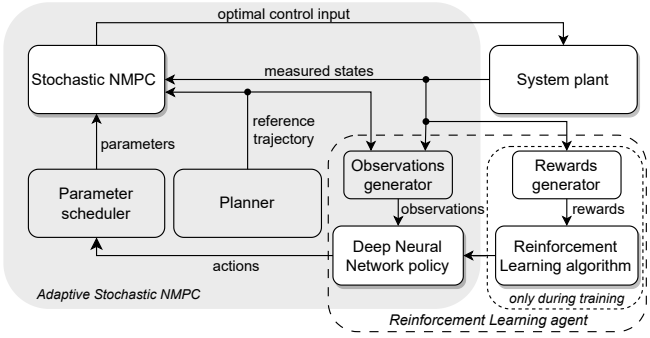

Fig. 2. Architecture of the Deep RL driven aSNMPC

The DNN policy computes new actions: κ and T_u (Sec.IV-A), and the parameter scheduler alters the SNMPC parameters every predefined switching time instance $T_{sw} = n \cdot T_{\text{s,sim}}, n \in \mathbb{N}$, where $T_{\text{s,sim}}$ is the environment discretization time. The new actions are computed with a feed-forward step through the DNN using observations generated based on current measured states and the future reference trajectory (Sec.IV-C).

During the training/learning phase, the DRL algorithm trains the DNN policy based on the collected rewards (Sec.IV-B) after a certain number of steps in the environment n_{steps}. Note that the DRL agent does not replace the SNMPC, i.e., control the vehicle motion, but only controls the SNMPC parameters. After the training is done, the rewards generator, as well as the DRL algorithm, are no longer needed. The rest of the architecture remains the same for the deployment phase. In this work, we adopt the Proximal Policy Optimization (PPO) method [32] as our DRL algorithm. PPO has gained attention for its ability to optimize policies with minimal hyperparameter adjustments. Figure 3 delineates the architecture of the DNN policy we conceive for this problem. We design a comprehensive training environment where the agent encounters diverse curvature profiles, and varying velocity patterns encompassing acceleration, deceleration, and velocity-keeping phases, along with time-varying disturbance magnitudes. This setup challenges the agent to adapt and make optimal decisions across a spectrum of contexts.

A. Defining the action space

The DRL agent has two actions: the nonlinear constraints factor robustification κ and the UPH length T_u. The UPH is within the propagation horizon, i.e. $T_u \in [0, T_p]$. Given that $T_u \in \mathbb{N}$, we choose a discrete action space which comprises $N_p = \frac{T_p}{T_s}$ discrete steps, each corresponding to a shooting node of the SNMPC.

To maintain a standardized action scale for constraint robustification, we segment κ into 21 discrete intervals, spanning from $\kappa_{\min} = 0$ to $\kappa_{\max} = 2$ at intervals of 0.1. Experiments have shown that DRL agents choose only robustification factor values less than 2.

B. Designing the reward function

We design our DRL agent to optimize the SNMPC's performance by minimizing the following aspects:

1) infeasibility
2) constraint violations
3) lateral deviation

Accordingly, we give a high reward for small lateral deviations, no reward when the constraints are violated and we impose a penalty when the SNMPC can't find a solution under the chosen parameters. Hence, the reward function is defined as following:

$$R = \begin{cases} -A, & \text{if SNMPC is infeasible} \\ 0, & \text{if constraints are violated} \\ A \cdot \exp\left(-\frac{e_{\text{lat}}^2}{\sigma_{\text{lat}}^2}\right), & \text{else} \end{cases} \tag{8}$$

Here, A represents the peak of reward, i.e., the maximum reward fed back to the agent, and σ_{lat} represents the spread in e_{lat} direction. Here, e_{lat} represents the lateral deviation computed based on the undisturbed states in simulation. When $e_{\text{lat}} = 0$, the agent gets the highest reward A. In this work, we adopt $A = 1$ and $\sigma_{\text{lat}} = 1\text{m}$. Note that $e_{\text{lat}} = \max[|e_{\text{lat},1}|, |e_{\text{lat},2}|, \ldots, |e_{\text{lat},n}|]$ with $n = \frac{T_{sw}}{T_{\text{s,sim}}}$, i.e., e_{lat} represents the absolute maximum measured lateral deviation between two parameter switching intervals T_{sw}.

C. Defining the observations

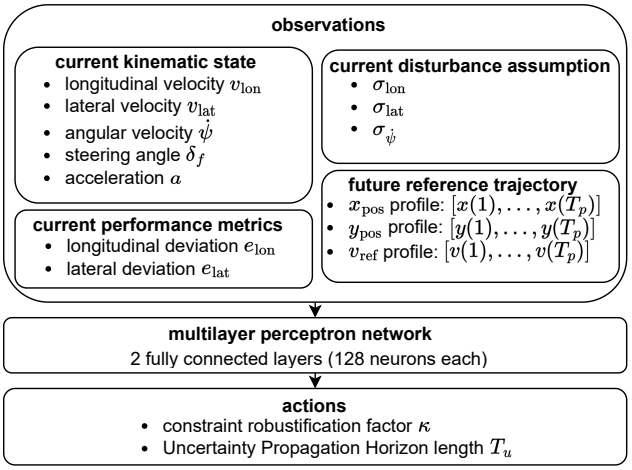

Fig. 3. Model of the Deep Neural Network driven aSNMPC

The observation space represents the agent's perceptual input from the environment. We provide the agent with a comprehensive set of relevant information necessary for informed decision-making as in Figure 3. We equip the agent with the capability to anticipate the future by providing it with the current reference trajectory within the prediction horizon. This foresight enables the agent to proactively determine the optimal actions (κ and T_u) that align with the specified objectives, guided by the anticipated dynamic profiles that the SNMPC should follow. The agent receives detailed future profiles, including x and y positions as well as velocity. To allow for generality, we disassociate the x and y coordinates from the map by transforming the reference trajectory to the ego vehicle's coordinates. This allows the

agent to make informed decisions regarding future combined longitudinal and lateral behaviors, distinguishing between scenarios like high-speed curves and straights or curves with substantial lateral uncertainty.

The agent is able to monitor its performance by assessing the outcomes of its most recent actions through the current longitudinal- and lateral deviations. Furthermore, we design the agent to observe the current ego kinematic state while decoupling it from specific coordinates, a design choice aimed at enhancing the generality of the learned policy.

Lastly, given that the SNMPC relies on the assumption of uncertainty distributions, the agent has access to these assumptions. When dealing with large uncertainties, the likelihood of constraint violations increases, resulting in a smaller robustification factor κ. This, in turn, requires a reduced UPH to maintain system stability.

Based on the discussions above, we define a general algorithm that learns and adapts the SNMPC parameters with DRL (Algorithm 1).

Algorithm 1 DRL-driven Adaptive SNMPC

Set SNMPC parameters: T_p, T_s, T_u, κ, n_s, $\boldsymbol{\sigma}_{w,\text{SNMPC}}$, W and W_e
for $i \in \{1, ..., N\}$ **do**
  Update initial state $\boldsymbol{x}_0$ with current measurements
  Generate samples $\tilde{\boldsymbol{X}}$ around the initial state $\boldsymbol{x}_0$
  Compute the PCE matrix solution A of Eq.2
  Update current SNMPC reference
  if switching time ($i \bmod T_{sw} = 0$) **then**
    Generate observations
    Execute the current RL policy
    Update SNMPC parameter κ and T_u
    if training phase **then**
      Generate reward signal
      if learning-step ($i \bmod n_{\text{steps}} \cdot T_{sw} = 0$) **then**
        Execute DRL algorithm for policy update
      end if
    end if
  end if
  for $j \leq N_p$ **do**
    Propagate the uncertain states $\mathbb{E}[\boldsymbol{x}_j]$ (Eq.4)
    Estimate nonlinear constraints $\mathbb{E}[h_j]$ and Var$[h_j]$
  end for
  Solve the SNMPC problem (Eq.1)
  Apply the first control input $\boldsymbol{u}_0^*$ on the real system
end for

V. Simulation Results: DRL driven Adaptive SNMPC

A. Simulation setup

We conduct a performance comparison between our Adaptive SNMPC (aSNMPC) approach and its static counterpart, Static SNMPC (sSNMPC) [6]. Both are subjected to significant additive Gaussian disturbances that impact the quality of state estimates. Experiments are carried out in the TUM-CONTROL simulation framework. The SNMPC is implemented using the ACADOS library [33] and CasADi in Python 3.9 employing the SQP RTI as the NLP Solver and HPIPM QP Solver, with a maximum limit of 50 iterations. The SNMPC configurations are detailed in Table III. The RL agent's training is carried out on an AMD Ryzen 7950X 5.70 GHz CPU using the PPO algorithm implemented within the Stable Baselines3 framework [34]. Configuration details are outlined in Table I. The training process in TUM-CONTROL simulation environment spans approximately 7.5 hours for 10^6 DRL agent steps, equivalent to $10^6 \cdot \frac{T_{sw}}{T_{\text{s,sim}}} = 125 \cdot 10^6$ SNMPC steps.

TABLE I
Hyper parameters of the Proximal Policy Optimization

Parameter	Value
Learning rate α	0.0007
Policy update every n_{steps}	512
Discount factor γ	0.99
GAE factor λ	0.95
Clipping parameter ϵ	0.2
Episode length in RL steps	40
Training steps N	10^6

We design the DRL agent with the following termination conditions: an episode terminates when the agent completes one lap. Furthermore, the episode truncates if the SNMPC problem becomes infeasible. After each training episode, the standard deviations of the simulated disturbances are randomized according to the ranges in Tab.II, and their values are known to the agent. We train the agent in a racetrack environment, and we evaluate its capability of generalization on other unseen racetracks (Sec.V-G). All the experiments in the following sections are evaluated by letting the SMPC control the vehicle for 110s $= \frac{110}{T_{\text{s,sim}}} = 5500$ SNMPC steps.

TABLE II
Standard deviation ranges used to sample and simulate changing state estimation disturbance distributions

state	σ		σ_{min}	σ_{max}
x,y position	$\sigma_{x,y}$	[m]	0.1	0.3
yaw angle	σ_{ψ}	[rad]	0.008	0.017
longitudinal velocity	$\sigma_{v,\text{lon}}$	[m/s]	0.5	1.0
lateral velocity	$\sigma_{v,\text{lat}}$	[m/s]	0.5	1.0
yaw rate	$\sigma_{\dot{\psi}}$	[rad/s]	0.04	0.08
steering tire angle	σ_{δ_f}	[rad]	0.001	0.0017

TABLE III
SNMPC parameters and disturbance configuration

Parameter	Value
Simulation sampling time $T_{\text{s,sim}}$	0.02s
SNMPC discretization time T_s	0.08s
Prediction horizon T_p	3.04s
Default UPH T_u	2s
Default robustification factor κ	0.42
Parameter switching time T_{sw}	$125 \cdot T_{\text{s,sim}}$
Disturbance ranges switch $T_{\boldsymbol{\sigma},sw}$	30 s
Simulation duration per evaluation	110s

B. Trajectory Following Performance

Once the agent is trained, we run an evaluation simulation affected by time-varying external Gaussian disturbances $\boldsymbol{\sigma}_w^{\text{sim}} = [\sigma_x, \sigma_y, \sigma_\psi, \sigma_{v,\text{lon}}, \sigma_{v,\text{lat}}, \sigma_{\dot{\psi}}, \sigma_{\delta_f}]^T$, where the standard deviations are altered every $T_{\boldsymbol{\sigma},sw} = 30$s, i.e. every 1500 SNMPC steps, and are randomly sampled from the ranges as in Tab.II. This differs from [6], where the disturbance magnitude is time-invariant. Note that the boxplot results may differ compared to [6], as the simulation time is shorter.

In Figure 4, we evaluate both MPCs in three different scenarios, focusing on absolute maximum lateral deviation. Minimizing this metric is crucial for ensuring lane-keeping and avoiding collisions with other road participants. While both SNMPCs perform similarly under nominal conditions, i.e. no disturbances, aSNMPC significantly outperforms sSNMPC when exposed to time-variant disturbances (see Tab.II). With accurate disturbance assumptions, aSNMPC achieves a **18.73%** improvement over sSNMPC in limiting maximum lateral deviation.

Importantly, aSNMPC demonstrates robust behavior by only showcasing a marginal **2.8%** degradation in maximum deviation when significant disturbances are present, compared to its optimal performance in the absence of disturbances.

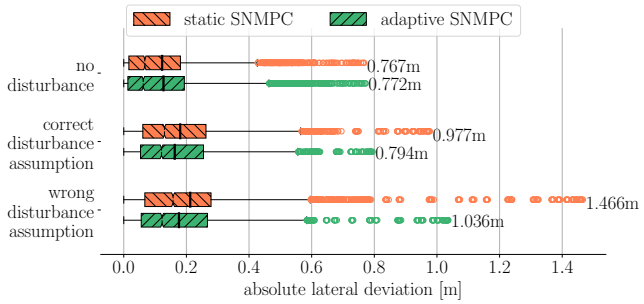

Fig. 4. Monteblanco racetrack: Effect of the disturbance on the sSNMPC and aSNMPC w.r.t. the absolute lateral deviation.

Conversely, when incorrect disturbance assumptions are made, i.e. $\boldsymbol{\sigma}_w^{\text{SNMPC}} \neq \boldsymbol{\sigma}_w^{\text{sim}}$, aSNMPC still outperforms sSNMPC by a significant **29.3%**, underscoring its robustness against outliers and extreme events. However, aSNMPC's performance declines by **23.3%** compared to correct disturbance assumptions, highlighting the critical importance of accurate disturbance assumptions and the need for a dedicated disturbance estimation unit when deploying SNMPC on a vehicle.

C. Analyzing Reinforcement Learning agent decisions

In Figure 1, we observe and analyze the decision-making patterns of the RL agent during a full lap. Before entering a curve while decelerating on a straight, the agent increases the UPH to its maximum, i.e. chooses to propagate uncertainty across the entire prediction horizon. It then reduces the robustification factor κ until it reaches its minimum value, ensuring the SNMPC is not overly conservative and that the vehicle follows the reference trajectory with minimal deviation.

During a curve, the agent gradually increases κ to a mid-range value, thereby slightly tightening the constraints. Also, the DNN gradually decreases T_u and stabilizes it at approximately 1.5s after the curve to prevent infeasibility caused by significant lateral disturbances encountered on straight paths. After a curve, while accelerating on a straight, the agent reduces both κ and T_u, i.e. stops tightening constraints, allowing the vehicle to operate at its longitudinal acceleration limits.

While maintaining a constant velocity on a straight, the agent increases κ to tighten constraints and mitigate the risk of aggressive control.

D. Computational Load

The computational times (Table IV) include all the computational workload the SNMPC needs to find a solution, including solving the OCP, observations generation, feedforward of the observations through the trained DNN policy to get the new parameters and updating the SNMPC parameters. Note that these evaluations are done using a standard laptop with an i7-11850H 2.50GHz CPU. The computational demands of the DRL-driven aSNMPC are still similar to the nominal SNMPC, achieving a minimum problem-solving frequency of 109.98Hz and mean of 149.4Hz, which aligns with our real-time control requirements of a minimum 50Hz solution frequency. Adapting the parameters κ and T_u reduces the maximum computational time by preventing overly conservative constraint robustification and unnecessary uncertainty propagation.

TABLE IV
CONTROLLER COMPUTATIONAL TIME (CPU:I7-11850H)

	sSNMPC	DRL-aSNMPC
Maximum [ms]	10.099	9.092
Mean [ms]	6.612	6.693

E. Enhancing feasibility with the DRL-aSNMPC

We conduct simulations with larger disturbances than described in Tab.II:

$$\begin{aligned} 0.8\text{m/s} \leq \sigma_{v,\text{lon}} \leq 1.5\text{m/s} \\ 0.7\text{m/s} \leq \sigma_{v,\text{lat}} \leq 1.2\text{m/s} \\ 0.05\text{rad/s} \leq \sigma_{\dot{\psi}} \leq 0.08\text{rad/s} \end{aligned} \tag{9}$$

In Figure 5, we illustrate the solver's status during the simulation. Propagating uncertainties through a fixed UPH, as in sSNMPC, leads to an infeasible problem when facing large external disturbances. However, adapting the UPH based on different driving tasks and uncertainties, as in RL driven aSNMPC, renders the problem feasible at each time step. This further showcases the importance of adapting UPH in a dynamic environment compared to SNMPC with fixed UPH.

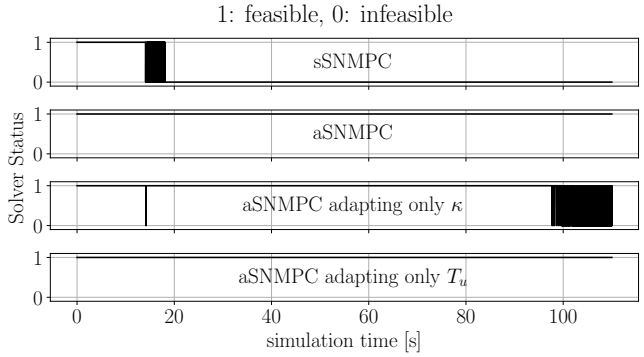

Fig. 5. Status of the solver for both sSNMPC and aSNMPC under large time-varying external disturbances, black areas denote that the status oscillates with a high frequency.

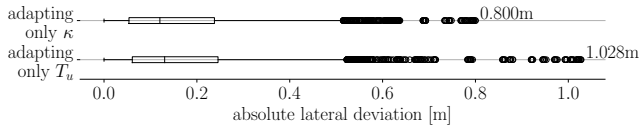

Fig. 6. Monteblanco racetrack: effect of the disturbance on two RL driven aSNMPC trained only to adapt κ and T_u respectively w.r.t. the absolute lateral deviation.

F. Impact of learning the robustification factor κ and the Uncertainty Propagation Horizon T_u

To demonstrate the influence of each parameter, we modify the action space (as discussed in Sec.IV-A) and train two distinct DRL agents. Each agent is designed to learn and adapt only a specific parameter:

1) DRL Agent 1: trains and learns only the constraints robustification factor κ, i.e., aSNMPC adapting only κ.
2) DRL Agent 2: trains and learns only the UPH length T_u, i.e. aSNMPC adapting only T_u.

Figure 6 illustrates the absolute lateral deviation evaluation for each agent. We notice that adapting the robustification factor κ affects the performance more than adapting the UPH length T_u. DRL Agent 1 shows a 18.11% improvement compared to the Static SNMPC (Fig.4), as adapting κ leads to less conservatism.
Conversely, as depicted in Fig.5, adapting only κ results in infeasibility. In contrast, adapting solely the UPH length T_u, makes the problem feasible, underscoring the importance of UPH in enhancing SNMPC's feasibility [6].

G. Generalization and robustness

The RL agent is only trained on the Monteblanco racetrack. To assess its adaptability and generalization capabilities, we challenge the agent to adapt SNMPC parameters in two previously unexplored tracks while exposed to significant disturbances: Modena (Fig. 7) and Las Vegas Motor Speedway (LVMS) (Fig.8). We notice that the DRL-driven aSNMPC compared to the sSNMPC consistently achieves performance comparable to its training on the Monteblanco racetrack as described in Sec.V-B. On both unfamiliar tracks, our aSNMPC outperforms the sSNMPC under different disturbance settings. Notably, when we assume an accurate disturbance assumption, our aSNMPC showcases an improvement of 31% on Modena and 28.4% on LVMS compared with the standard SNMPC. When disturbance assumptions are inaccurate, the disparity widens, resulting in a 39% improvement on Modena and 31% on LVMS.

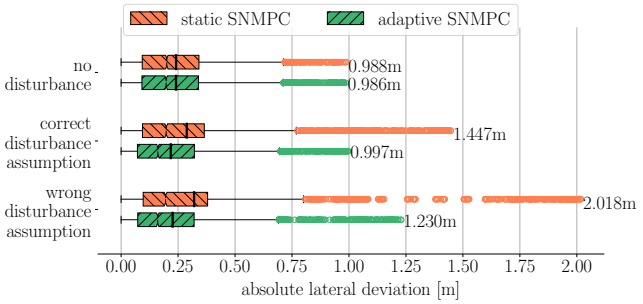

Fig. 7. Inexperienced Modena racetrack: effect of the disturbance on the aSNMPC and sSNMPC w.r.t. the absolute lateral deviation during 110s = 5500 simulation steps subject to varying disturbance ranges.

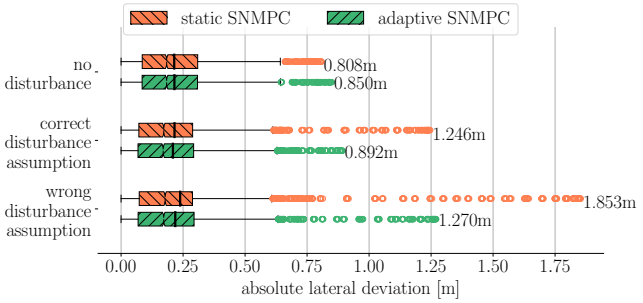

Fig. 8. Inexperienced Las Vegas Motor Speedway (LVMS): effect of the disturbance on the aSNMPC and sSNMPC w.r.t. the absolute lateral deviation during 110s = 5500 simulation steps subject to varying disturbance ranges.

VI. Conclusions and Future Work

To enhance the feasibility of Stochastic Nonlinear Model Predictive Control (SNMPC) and to improve its closed-loop performance, we design a look-ahead Deep Reinforcement Learning (RL) agent to automatically learn and adapt two major SNMPC parameters: the nonlinear constraints robustification factor κ and Uncertainty Propagation Horizon (UPH) T_u. Leveraging the current kinematic state, disturbance assumptions, performance metrics, and future reference trajectory, the RL agent anticipates upcoming control tasks and dynamically selects the most suitable SNMPC configuration to optimize predefined objectives.
Our experimental findings reveal that the robustification factor κ affects the closed-loop performance and its adaptation improves it by being less conservative than the Static SNMPC. On the other hand, adapting the UPH length, T_u, has a substantial effect on feasibility, rendering previously infeasible static SNMPC problems feasible when subjected to strong disturbances.

In the context of motion control for autonomous vehicles following a raceline trajectory at speeds of up to 37.5m/s, our Adaptive SNMPC (aSNMPC) exhibits a significant improvement in limiting maximum lateral deviation. An analysis of the DRL agent's decisions reveals that its actions are contextually driven by the current vehicle state and upcoming driving situations. Our experimental results validate the robustness of our Deep RL-driven aSNMPC, indicating that it avoids overfitting to the training conditions.

Nevertheless, as evident in Figures 4, 7, and 8, it is apparent that both Static and Adaptive SNMPC performance experiences significant degradation when the disturbance assumptions are inaccurate. To this end, we propose two key directions for future research. First, we recommend extending the SNMPC framework with an online disturbance estimation unit. Second, we suggest incorporating the emulation of inaccurate disturbance assumptions during the agent's training phase to enhance its adaptability in challenging scenarios.

REFERENCES

[1] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[2] E. Bradford and L. Imsland, "Stochastic nonlinear model predictive control using gaussian processes," in *2018 european control conference (ECC)*. IEEE, 2018, pp. 1027–1034.

[3] E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio Chanona, "Stochastic data-driven model predictive control using gaussian processes," *Computers & Chemical Engineering*, vol. 139, p. 106844, 2020.

[4] L. Fagiano and M. Khammash, "Nonlinear stochastic model predictive control via regularized polynomial chaos expansions," in *2012 IEEE 51st IEEE conference on decision and control (cdc)*. IEEE, 2012.

[5] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz, "Stochastic nonlinear model predictive control with probabilistic constraints," in *2014 American control conference*. IEEE, 2014, pp. 2413–2419.

[6] B. Zarrouki, C. Wang, and J. Betz, "A stochastic nonlinear model predictive control with an uncertainty propagation horizon for autonomous vehicle motion control," *arXiv preprint arXiv:2310.18753*, 2023.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[8] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[9] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[10] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, "Fusion of machine learning and mpc under uncertainty: What advances are on the horizon?" in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 342–357.

[11] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[12] S. Gros and M. Zanon, "Reinforcement learning for mixed-integer problems based on mpc," 2020.

[13] W. Cai, A. B. Kordabad, H. N. Esfahani, A. M. Lekkas, and S. Gros, "Mpc-based reinforcement learning for a simplified freight mission of autonomous surface vehicles," 2021.

[14] C. D. McKinnon and A. P. Schoellig, "Experience-based model selection to enable long-term, safe control for repetitive tasks under changing conditions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2977–2984.

[15] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[16] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[17] A. D. Bonzanini, D. B. Graves, and A. Mesbah, "Learning-based smpc for reference tracking under state-dependent uncertainty: An application to atmospheric pressure plasma jets for plasma medicine," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 2, pp. 611–624, 2021.

[18] F. Fiedler and S. Lucia, "Model predictive control with neural network system model and bayesian last layer trust regions," in *2022 IEEE 17th International Conference on Control & Automation (ICCA)*. IEEE, 2022, pp. 141–147.

[19] U. Rosolia and F. Borrelli, "Sample-based learning model predictive control for linear uncertain systems," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 2702–2707.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[22] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[23] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, "Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[24] E. Bøhn, S. Moe, S. Gros, and T. A. Johansen, "Reinforcement learning of the prediction horizon in model predictive control," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 314–320, 2021.

[25] E. Bøhn, S. Gros, S. Moe, and T. A. Johansen, "Optimization of the model predictive control meta-parameters through reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106211, 2023.

[26] B. Zarrouki, V. Klös, N. Heppner, S. Schwan, R. Ritschel, and R. Voßwinkel, "Weights-varying mpc for autonomous vehicle guidance: a deep reinforcement learning approach," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 119–125.

[27] F. Oldewurtel, D. Sturzenegger, P. M. Esfahani, G. Andersson, M. Morari, and J. Lygeros, "Adaptively constrained stochastic model predictive control for closed-loop constraint satisfaction," in *2013 American Control Conference*. IEEE, 2013, pp. 4674–4681.

[28] D. Muñoz-Carpintero, G. Hu, and C. J. Spanos, "Stochastic model predictive control with adaptive constraint tightening for non-conservative chance constraints satisfaction," *Automatica*, vol. 96, pp. 32–39, 2018.

[29] T. L. Santos, V. M. Cunha, and A. Mesbah, "Stochastic model predictive control with adaptive chance constraints based on empirical cumulative distributions," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 257–11 263, 2020.

[30] J. Suh and K. Yi, "A new adaptive uncertainty propagation method based stochastic model predictive control for automated driving vehicles," in *2017 american control conference (acc)*. IEEE, 2017, pp. 5660–5665.

[31] P. Karle *et al.*, "Edgar: An autonomous driving research platform–from feature development to real-world application," *arXiv preprint arXiv:2309.15492*, 2023.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[33] 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*, Oct 2021. [Online]. Available: https://doi.org/10.1007/s12532-021-00208-8

[34] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html