IMFlow: Inverse Modeling with Conditional Normalizing Flows for Data-Driven Model Predictive Control

Yi Zhang[1] and Lars Mikelsons[2]

Abstract— **Inverse modeling is the process uncovering the relationships from the system observations to its inputs. It is essential in various fields such as control, robotics, and signal processing. We propose an inverse modeling method using amortized variational inference based on conditional normalizing flows (IMFlow). IMFlow is data-driven and can therefore be applied to black-box environments with limited observability and unknown complexity. Besides, the probabilistic modeling characteristics of conditional normalizing flows allow IMFlow to cope with unknown system uncertainties. We deploy IMFlow as a probabilistic model predictive controller, which estimates the control inputs as stochastic processes based on reference signals and system responses. In addition, we also adjust IMFlow to an online model-free reinforcement learning setting. We demonstrate our proposed method achieves the same accuracy in comparison to the standard model predictive control method using white-box models.**

I. INTRODUCTION

In general, system identification refers to the process of establishing a mathematical model for a system using measurements of the input and output signals of the system [1]. Model Predictive Control (MPC) applies the identified model to predict the system's responses to control inputs, and to choose the best control inputs through optimizing an objective. Therefore, system identification plays a fundamental role in designing a model predictive controller.

Denoting $\boldsymbol{u}_t$ as input signals, $\boldsymbol{x}_t$ as the state signals, and $\boldsymbol{y}_t$ as the observable output signals, the identified models g, h satisfy:

$$\boldsymbol{x}_{t+1} = g(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{1a}$$

$$\boldsymbol{y}_t = h(\boldsymbol{x}_t). \tag{1b}$$

Over a prediction horizon $[N_1, N_2]$, a MPC optimization problem aims to find the optimal control sequence $\boldsymbol{u}_{t+N_1:t+N_2}$, by minimizing an objective function with subject to system constraints. Given a reference signal (target) $\boldsymbol{r}_{t+N_1:t+N_2}$, the objective function J is usually quadratic for the sake of convex optimization [2] as follows:

$$\min_{\boldsymbol{u}} J(\boldsymbol{r}_t, \boldsymbol{y}_t) = \min_{\boldsymbol{u}} \sum_{i=N_1}^{N_2} \boldsymbol{r}_{t+i} - \boldsymbol{y}_{t+i|t} {}_2, \tag{2}$$

subject to

$$\boldsymbol{x}_{t=0} = \boldsymbol{x}_0,\ \boldsymbol{x}_{t+1} = g(\boldsymbol{x}_t, \boldsymbol{u}_t),\ \boldsymbol{y}_t = h(\boldsymbol{x}_t), \tag{3a}$$

$$\boldsymbol{u}_{lb} < \boldsymbol{u}_{t+j|t} < \boldsymbol{u}_{ub},\ \forall j \in [N_1, N_2], \tag{3b}$$

$$\boldsymbol{y}_{lb} < \boldsymbol{y}_{t+i|t} < \boldsymbol{y}_{ub},\ \forall i \in [N_1, N_2]. \tag{3c}$$

[1,2]Yi Zhang and Lars Mikelsons are with Chair of Mechatronics, Faculty of Applied Computer Science, University of Augsburg, Germany {yi.zhang, lars.mikelsons}@uni-a.de

The symbols $\boldsymbol{y}_{t+i|t}$, $\boldsymbol{u}_{t+i|t}$, $\boldsymbol{x}_{t+i|t}$ stand for predicted signals by models g, h based on $\boldsymbol{y}_t$, $\boldsymbol{u}_t$, $\boldsymbol{x}_t$.

In order to circumvent explicit system identification for control tasks, inverse modeling methods can be applied, i.e., deriving inputs $\boldsymbol{u}$ directly from $\boldsymbol{y}$. Generally, the inverse problem is solved as an optimization problem by minimizing the least square error between the simulated and measured system responses. In most practical cases, the inverse problem is ill-posed because not all state variables or initial conditions are known [3]. Besides, for systems with higher complexity and unknown uncertainty, optimization-based methods are quite computationally burdensome. To tackle these challenges, deep learning based methods are exploited and improve the prediction accuracy and efficiency. In [4], common deep learning architectures are compared in predictive control tasks. The research [5] put forward to train neural networks with additional theory-guided loss evaluating governing equations and boundary/initial conditions. In [6], backpropagation neural networks are applied in inverse modeling with small datasets. In [7], a neural network is implemented as an adaptive filter in the adaptive inverse control framework. However, the above mentioned methods do not take inverse uncertainty quantification into account.

TABLE I: Comparison of RL-based MPC methods

	model-based	**model-free**
online	GP [8][9][10] Neural Network [11][12] Dynamic Mirror Descent [13]	Q-Learning [14] Deep Q-Network [15][16] Actor–Critic [17][18] Policy Gradient [19][20]
pros	data-efficient more explainable	computationally efficient suitable for various tasks
cons	can be computationally complex; tend to overfitting	dependent on environment large dataset required
offline	MBOP [21] MPUR [22] UMBRELLA [23] PETS [24]	Q-ensemble with BC [25] Actor–Critic [26] Analytic Policy Gradient [27]
pros	less computationally complex; relatively stable	diverse data allowed steadily performant
cons	highly dependent on data quality; less adaptive	sub-optimally performant

In terms of the iterative prediction process of future control input (action) utilizing a simulation model (agent) in interaction with the real system (environment), learning-based MPC is very similar to model-based reinforcement learning (RL) [28]. The difference is, model-based RL usually aims to find a global policy for a long horizon, while MPC focuses on optimizing a local policy for an finite horizon [29]. Model-based approaches build a predictive model of an environment and derive a controller from it, while model-free approaches learn a direct mapping from states to actions [30][31]. RL optimizes the closed-loop performance in the MPC scheme using data from the real system, rather than purely improving the model predictive accuracy. Therefore, RL-based methods outperform among learning-based MPC [32].

In Table I, we summarize the state-of-the-art RL-based MPC methods in black-box environments according to model-based/model-free approaches [33] and data collection methods (online/offline) [34][35].

From the perspective of model-free or model-based, IMFlow is close to model-free because the policy is direct inverse modeling from outputs to inputs. However, IMFlow does not optimize the policy through a reward based on the interaction between the agent and the environment and thus is different from RL. In order to keep the strengths of both online and offline approaches, we propose to train a supervised probabilistic model with conditional normalizing flows (cNFs) for directly capturing state-to-input relationship offline, then adapt the model online for precise control tasks. Besides, the way of online adaptation also relates to test-time adaptation [36]. From the perspective of RL, the offline supervised training and online sample-specific adaptation balances the exploration and exploitation trade-off [37].

In other researches, based on sampling-based MPC methods [38], e.g., model predictive path integral (MPPI) [39] and sample-efficient cross-entropy method (iCEM) [40], normalizing flows are trained to learn the sampling distribution of control inputs, proposed in [41] (NFMPC), [42] (FlowMPPI), and [43] (FlowMPPI, FlowiCEM). The sampling-based approaches sample repeatedly random action trajectories, evaluate them under the model, and re-fit the sampling distribution to the best. Unlike that, IMFlow do not sample action trajectories randomly. Besides, in these methods, the control sequences are sampled point-by-point with cNFs. Unlike standard cNFs, we modify the architecture to predict control sequences as a time series, which is more suitable for dynamic systems.

Contributions: We propose a novel method IMFlow[1] to perform inverse modeling using conditional normalizing flows. The identified inputs from outputs are modeled as stochastic processes through a probabilistic mapping. Then, the trustworthiness of inputs is indicated with confidence intervals. We put forward three modes to implement IMFlow into MPC framework for black-box environments. Firstly, IMFlow can be applied in an open loop to efficiently obtain a long sequence of control inputs. Secondly, in closed-loop mode, an additional subspace encoder can be integrated to IMFlow for acquiring information from the system feedback. Thirdly, if the closed-loop approach can not achieve demanding accuracy, IMFlow can also be efficiently re-trained partially online. We compare the proposed methods with standard MPC in terms of accuracy, computation time and performance in trajectory tracking tasks.

[1]Code available https://github.com/yiyi1zhang/IMFlow

II. PRELIMINARIES

Normalizing flows learn complex distributions from a simple base distribution with invertible transformations under the change of variables formula [44]. For variational inference, conditional normalizing flows are applied to learn a common inference function, which maps each observation to its approximate posterior [45].

A. Conditional Normalizing Flows

Denote the observation as $\boldsymbol{y}$, the target as $\boldsymbol{u}$, and the parameters of a cNF as ϕ, the posterior distribution $p_\phi(\boldsymbol{u}|\boldsymbol{y})$ is

$$p_\phi(\boldsymbol{u}|\boldsymbol{y}) = p(\boldsymbol{z}|\boldsymbol{y})\ \det\left(\frac{d\boldsymbol{z}}{d\boldsymbol{u}}\right)\Bigg|. \tag{4}$$

In the training process, $\boldsymbol{z} = f_\phi(\boldsymbol{u};\boldsymbol{y})$, it is assumed $p(\boldsymbol{z}|\boldsymbol{y}) = p(\boldsymbol{z}) = N(\boldsymbol{0},\mathbb{I})$. Using a batch of dataset $\{\boldsymbol{u}^{(m)},\boldsymbol{y}^{(m)}\}$, $m = 1,\dots,M$, ϕ is optimized by minimizing the Kullback-Leibler divergence between the ground truth and the model-induced posterior [46]:

$$\mathbb{E}_{p(\boldsymbol{y})}KL(p(\boldsymbol{u}|\boldsymbol{y})||p_\phi(\boldsymbol{u}|\boldsymbol{y})) = \frac{1}{M}\sum_{m=1}^{M}[\frac{1}{2}\ \ f_\phi(\boldsymbol{u}^{(m)};\boldsymbol{y}^{(m)})\ \ ^2$$
$$-\,log\ \det \boldsymbol{J}_{f_\phi}^{(m)}], \tag{5}$$

where $\boldsymbol{J}_{f_\phi}$ is the Jacobian matrix of f_ϕ evaluated at $\boldsymbol{u}$. In the inference process, the posterior $p_\phi(\boldsymbol{u}|\boldsymbol{y})$ is obtained by first sampling $\boldsymbol{z} \sim p(\boldsymbol{z}|\boldsymbol{y}) = N(\boldsymbol{0},\mathbb{I})$, then passing it through the bijective mapping $f_\phi^{-1}(\boldsymbol{z};\boldsymbol{y})$.

The cNF can be interpreted as an indirect correlation map between target and observation through a likelihood-free modeling from the joint distribution of target and observation to the latent space.

B. SUBNET

It was proposed to use SUBNET [47] for subspace modeling in nonlinear MPC system. SUBNET consists of a subspace encoder ψ_η and two models g_ϑ, h_φ. In the horizon $[0,H)$, the subspace encoder ψ_η extracts information from previous inputs and outputs for N steps to an estimated initial state $\hat{x}_{t|t}$. Then, the models g_ϑ, h_φ propagate the initial state to the future states $\hat{x}_{t+1:t+H-1|t}$ and outputs $\hat{y}_{t+1:t+H-1|t}$. For $i = 0,\dots,H-1$, the process is formulated as:

$$\hat{x}_{t|t} = \psi_\eta(u_{t-N:H-1}, y_{t-N:t}) \tag{6a}$$
$$\hat{x}_{t+i+1|t} = g_\vartheta(\hat{x}_{t+i|t}, u_{t+i}, \hat{e}_{t+i|t}) \tag{6b}$$
$$\hat{y}_{t+i|t} = h_\varphi(\hat{x}_{t+i|t}) \tag{6c}$$
$$\hat{e}_{t+i|t} = y_{t+i} - \hat{y}_{t+i|t} \tag{6d}$$

The models $\psi_\eta, g_\vartheta, h_\varphi$ are optimized by minimizing the mean squared error (MSE) between $\hat{y}_{t+i|t}$ and y_{t+i}.

We apply the subspace encoder to obtain implicit state information from the system feedback.

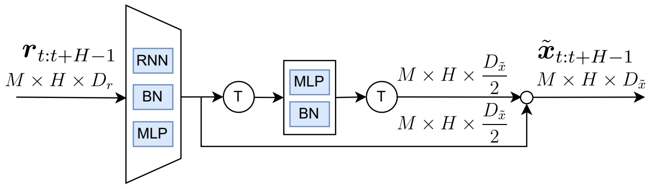
III. METHODS

Our proposed method IMFlow is developed from BayesFlow for the sake of identifying inputs as time series from observations. We aim to predict the control inputs in a finite time horizon from given reference signals (open loop) or from given reference signals along with the system outputs (closed loop). Moreover, the closed-loop IMFlow can be adapted into an online RL setting.

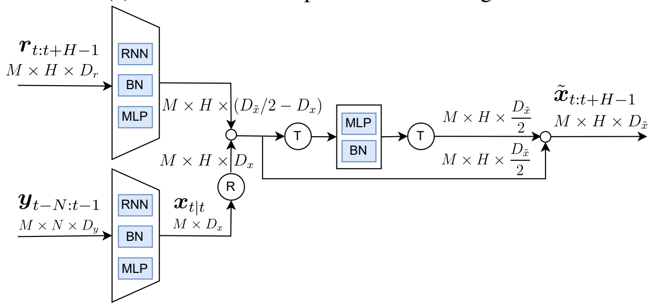
A. *From BayesFlow to IMFlow*

BayesFlow [46] is a parameter identification architecture, consisting of a summary network (SM) for squeezing information from stochastic observations $\boldsymbol{y}$, and a cNF for transforming the base distribution $\boldsymbol{z} \sim N(\boldsymbol{0}, \mathbb{I})$ to the target posterior distribution of the underlying parameters of the observations.

Our IMFlow inherits the basic structure of BayesFlow. SM models the reference signals $\boldsymbol{r}$ (and system outputs $\boldsymbol{y}$) to the hidden variable $\tilde{\boldsymbol{y}}$. In order to process the information in both time and feature dimensions, we construct SM with double lanes. On the one lane, only the feature dimension is processed. On the other lane, the time dimension is further learned after the feature dimension is processed. The architectures are shown in Fig.1.


$\boldsymbol{r}_{t:t+H-1}$
$M \times H \times D_r$
RNN
BN
MLP
T
MLP
BN
T
$M \times H \times \frac{D_{\tilde{x}}}{2}$
$M \times H \times \frac{D_{\tilde{x}}}{2}$
$\tilde{\boldsymbol{x}}_{t:t+H-1}$
$M \times H \times D_{\tilde{x}}$

(a) one-head SM: input is reference signals $\boldsymbol{r}$.


$\boldsymbol{r}_{t:t+H-1}$
$M \times H \times D_r$
RNN
BN
MLP
$M \times H \times (D_{\tilde{x}}/2 - D_x)$
T
MLP
BN
T
$M \times H \times \frac{D_{\tilde{x}}}{2}$
$\tilde{\boldsymbol{x}}_{t:t+H-1}$
$M \times H \times D_{\tilde{x}}$
$M \times H \times D_x$
$M \times H \times \frac{D_{\tilde{x}}}{2}$
R
$\boldsymbol{y}_{t-N:t-1}$
$M \times N \times D_y$
RNN
BN
MLP
$\boldsymbol{x}_{t|t}$
$M \times D_x$

(b) two-head SM: inputs are reference signals $\boldsymbol{r}$ and system outputs $\boldsymbol{y}$.

Fig. 1: Summary network architectures of IMFlow. (RNN: recurrent neural network, BN: batch normalization, MLP: multiple layer perceptron, T: transpose, R: repeat, ∘: concatenate, H: horizon size, N: window size, D_*: dimension)

To model stochastic processes in the latent space, the base distribution of the cNF in IMFlow is not a standard normal distribution but a standard Wiener process $\boldsymbol{w}(t)$. A standard Wiener process [48] (often called Brownian motion) is a random variable depending continuously on $t \in [0, H]$ and satisfies:

(1) $\boldsymbol{w}(0) = \boldsymbol{0}$;
(2) for $0 \leq s < t \leq H$, $\boldsymbol{w}(t) - \boldsymbol{w}(s) \sim \sqrt{t-s}\, N(\boldsymbol{0}, \mathbb{I})$;
(3) for $0 \leq s_1 < t_1 < s_2 < t_2 \leq H$, $\boldsymbol{w}(t_1) - \boldsymbol{w}(s_1)$ and $\boldsymbol{w}(t_2) - \boldsymbol{w}(s_2)$ are independent.

We discretize the Wiener process with a time step Δt_s as $d\boldsymbol{w} \sim \sqrt{\Delta t_s}\, N(\boldsymbol{0}, \mathbb{I})$. It indicates the prediction at later time step has larger variance than the earlier one, which agrees with real scenarios. Correspondingly, the target posterior is the joint distribution of H steps, $p_\phi(\boldsymbol{u}_{t:t+H-1|t} | \tilde{\boldsymbol{y}}_{t:t+H-1|t})$. We use RealNVP [49] as the architecture of normalizing flows.

B. *Open-Loop IMFlow for Fast Planning*

Assuming the system is stable and the path is free of obstacle, we can generate control input sequences quickly from reference signals. We train the one-head SM (Fig.1a) and the forward cNF with split $\{\boldsymbol{u}, \boldsymbol{r}\}$ pairs at the length of H time steps, then sample the base distribution $\boldsymbol{w}(t)$, $t = 1 : H$ and pass the samples into the inverse cNF as well as $\boldsymbol{r}$ into SM. Because the open-loop procedure has no interaction with the system, the calculation is very efficient and can be used in fast planning. Besides, the estimated inputs are modeled as stochastic processes and thus provide the possibility of uncertainty quantification.

C. *Closed-Loop IMFlow for Predictive Control*

However, in the long term, the inputs generated by open-loop IMFlow may cause noticeable drift from the reference signals to the real system responses. Then, the closed-loop IMFlow with the two-head SM (Fig.1b) is an option. We apply the subspace encoder of SUBNET as another head in SM to acquire an implicit initial state $\tilde{\boldsymbol{x}}_{t|t}$. Pre-training the SUBNET can accelerate convergence of IMFlow. In the training phase, the subspace encoder is re-trained to further optimize the extracted information from the previous system responses. Both $\boldsymbol{y}$ and $\boldsymbol{r}$ in the training set are observable system output signals. To distinguish $\boldsymbol{y}$ from $\boldsymbol{r}$ and improve the robustness of prediction for noisy models, we add small white noise to $\boldsymbol{y}$ during training. The loss function is a sum of Kullback-Leibler divergence5 and the MSE between the reference and the predicted system responses by SUBNET. During deployment in the MPC framework, the subspace encoder handles the last system feedback $\boldsymbol{y}_{t-N:t-1}$, while the other head in SM receives the next reference $\boldsymbol{r}_{t:t+H-1}$, and the inverse cNF calculates $\boldsymbol{u}_{t:t+H-1}$ from the sampled latent variable.

D. *Online-RL IMFlow for Predictive Control*

When the test data lies out of or in the tail of distribution of the training set, the trained IMFlow cannot estimate control input precisely. Here, we re-train part of IMFlow, for example, the last layers of the subspace encoder for previous system outputs, by minimizing the distance between system responses and references. The algorithm is an adaptation of closed-loop IMFlow, as shown in Algorithm 1. The update rate of system is $\Delta t_p = H \times \Delta t_s$. The window size N is assumed to be an integer multiple of the horizon size H.

Algorithm 1: Online-RL IMFlow MPC

```
Data: $k = 1 : K$, $\boldsymbol{y}_0$, $\boldsymbol{x}_0$, $\boldsymbol{r}_{1:K}$, $\delta$.
$k \leftarrow 1$, $\boldsymbol{u}_0 \leftarrow \boldsymbol{0}$;
while $k \leq K$ do
    $\boldsymbol{y}_k, \boldsymbol{x}_k \leftarrow$ model$(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})$;
    if $MSE(\boldsymbol{y}_k, \boldsymbol{r}_k) > \delta$ then
        re-train part of IMFlow by $min\, MSE(\boldsymbol{y}_k, \boldsymbol{r}_k)$;
    end
    $\boldsymbol{r}_{kH:(k+1)H-1} \leftarrow$ interpolate$(\boldsymbol{r}_{k:k+1})$;
    if $k \leq N/H$ then
        $\boldsymbol{y}_{past} \leftarrow$ repeat$(\boldsymbol{y}_0) \cup$ interpolate$(\boldsymbol{y}_{0:k})$;
    else
        $\boldsymbol{y}_{past} \leftarrow$ interpolate$(\boldsymbol{y}_{k-N/H:k})$;
    end
    $\boldsymbol{u}_{kH:(k+1)H-1} \leftarrow$ IMFlow$(\boldsymbol{y}_{past}, \boldsymbol{r}_{kH:(k+1)H-1})$;
    $\boldsymbol{u}_k \leftarrow \overline{\boldsymbol{u}_{kH:(k+1)H-1}}$
end
Result: $\boldsymbol{u}_{0:K}$
```

IV. EXPERIMENTS

We validate open- and closed-loop IMFlows with a multi-body dynamic vehicle model. Then, we compare the prediction accuracy and computation time as the horizon size grows. For closed-loop IMFlow, we investigate the prediction deviation and precision against the intensity of noise added onto the model. At last, we apply open-loop, closed-loop, and online-RL IMFlow in trajectory tracking with a single-track kinematic model. The results for the multi-body model are compared with a nonlinear MPC using a dynamic single-track model, while the results of single-track model are compared with a linear MPC with a kinematic single-track model. The MPC experiments are supported by the *do-mpc*[2] package. We also compare the results of horizon test, noise test, and trajectory tracking with two RL methods Analytic Policy Gradient (APG) [27] and Probabilistic Ensembles Trajectory Sampling (PETS) [24]. Due to instability of PETS for the multi-body vehicle model, we only evaluate PETS on trajectory tracking tasks.

A. Dataset

We summarize the two datasets generated with the multi-body and the single-track model respectively in Table II. The implementation of both models are from *CommonRoad*[3]. The datasets are generated by solving the ordinary differential equations of the models from various inputs signals, such as, sinus, exponential, logarithmic, step, and triangle functions. The sizes of training, validation and test set are 2000, 500 and 100. For training and validation, all samples are segmented into the length of horizon size (5). The time step Δt_s is 0.01 s. The simulation lasts from 0 s to 3 s.

B. Ablation Study

We compare IMFlow methods with MPC with a nonlinear dynamic single-track model for a multi-body vehicle. The following three constructions are considered:

- cNF for point-by-point prediction ("cNF")
- open-loop IMFlow ("o.IMFlow")
- closed-loop IMFlow with previous system outputs ("c.IMFlow")

The networks are trained on a GPU and evaluated on a CPU. All of the above three methods are sampled for 50 times. Fig.2 shows the comparison of control inputs prediction and the resulting system responses with IMFlow methods and MPC. The open- and closed-loop IMFlow with time series modeling outperform the cNF significantly and are slightly less accurate than the MPC. Both methods achieve the same accuracy, as shown in Table III. Compared with the reference input sequences, the IMFlow methods predict smoother actions. From the perspective of computation time, the cNF is the fastest and about 15 times faster than MPC in terms of time per simulation time step. IMFlows with 50 sampling times take comparable time to MPC with one pass. The closed-loop IMFlow is about 40 % slower than the open-loop IMFlow. The computation time of IMFlows can be further reduced by less sampling. For sampling only one time, open-loop IMFlow takes $1/5$ and closed-loop IMFlow takes $1/3$ computation time in comparison to MPC.

[2]https://www.do-mpc.com/en/latest/
[3]https://github.com/CommonRoad

TABLE II: Dataset

model	states	inputs	outputs
#	29	2	6
multi-body	xyz-position & velocity, yaw, pitch, roll angle & rate of center, front & rear, steering angle, angular velocity of four wheels	steering velocity, long. acceleration	x position, y position, steering angle, long. velocity, yaw angle & rate
#	4	2	4
single-track	x-position, y-position, long. velocity, yaw angle	steering angle, long. acceleration	x-position, y-position, long. velocity, yaw angle

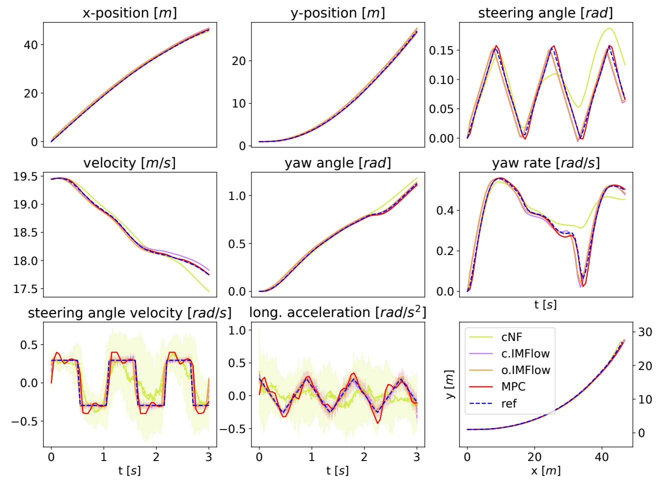


Fig. 2: Comparison between IMFlows and MPC

TABLE III: Comparison to MPC

methods	cNF	o.IMFlow	c.IMFlow	MPC
norm. RMSE	0.0438	0.0198	0.0176	0.00811
com. time 50 [ms]	0.139	1.87	2.71	–
com. time 1 [ms]	0.139	0.581	0.964	2.59

C. Horizon Test

We investigate the influence of the length of horizon on the prediction accuracy and computation time of open- and closed-loop IMFlow in comparison with MPC. Because of computational limitations and model inaccuracy, a larger control horizon can lead to larger errors.

As the horizon size grows, the normalized root mean squared errors (RMSEs) of IMFlows and MPC increases linearly, while APG is relatively insensitive to horizon size and remains at an average level. At larger horizon sizes, IMFlows overtake MPC and achieve lower deviation. Besides, closed-loop IMFlow takes less computation time than MPC when the horizon size is larger than 10, while open-loop IMFlow with 50 sampling times saves about 1/3 time in comparison to MPC. Surprisingly, APG works more than 10 times faster than the other methods and stays at 1.5 ms as horizon changes (shown in Fig.3). It is important to note that IMFlow has a far more complex architecture than APG, and that vectorized sampling causes IMFlow to slowdown.

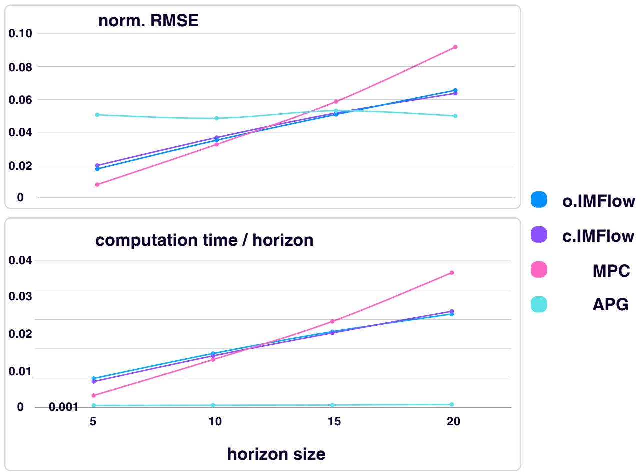


Fig. 3: Influence of horizon size

D. Influence of noise

To train the closed-loop IMFlow, we use a noise factor multiplied onto additive white noise in the previous system outputs to distinguish the system outputs from the reference. In the deployment of MPC, the training with noise also enhances the robustness of IMFlow against noise in the system. We compare the normalized RMSE of closed-loop IMFlow trained with differently intense noise in the previous system responses. The closed-loop IMFlow achieves lower deviation for controlling noisy systems. On the contrary, the performance of the standard MPC without any estimator and APG are heavily impaired by noise. Besides, we can observe the mean prediction interval width (MPIW) of IMFlow. MPIW remains constantly at 0.06 with increasing noise factor, indicating IMFlow is able to identify the control inputs precisely despite of noise in the system. Because of the capability pf probabilistic modeling, IMFlow is able to overcome some disturbance to the system and thus can be applied in robust control.

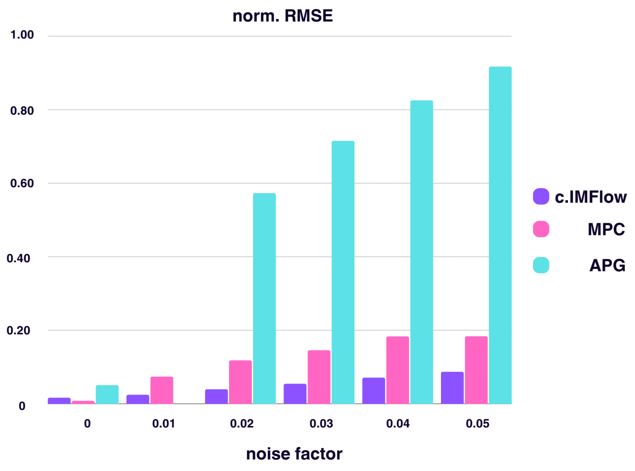
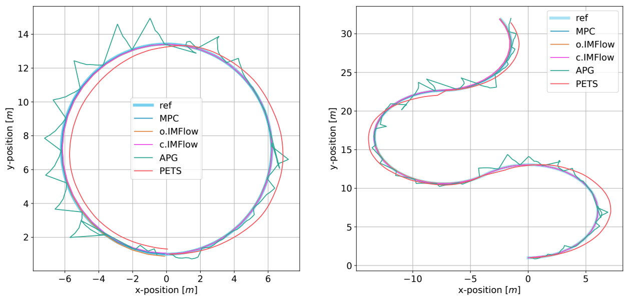


Fig. 4: Influence of noise factor

E. Trajectory Tracking

We implement the open- and closed-loop IMFlow for trajectory tracking of a kinematic single-track vehicle model. As shown in Fig.5a and Table IV, MPC is the most accurate method because of fully comprehension of the single-track model. Two IMFlow methods also achieve comparable accuracy as PETS. However, APG diverges accumulatively after extrapolation and must be adjusted frequently. Besides, we measure the computation time of one single simulation step with sampling the latent variable of IMFlows for 50 times in Table IV. In comparison to the results for the multi-body system, as shown in Table III, the computation time of two IMFlows remains the same, while MPC takes 20 % less time. Based on this, we could infer that, as the system complexity increases, the computation time of MPC will increase as well. IMFlows, in contrast, do not respond strongly to system complexity since they regard the system as a black box. The computational time with sampling is slightly longer than that of the deterministic APG, and 10 times shorter than that of the probabilistic PETS. Considering the re-training expense, PETS is much more flexible than IMFlow and APG due to its low computational complexity while training. On the other hand, IMFlow as an optimizer-free method takes 10 times longer than PETS for each iteration, which can be further accelerated. In comparison to the other optimizer-free method, APG, IMFlow outperforms in terms of both computational time and the frequency of necessary re-training. However, PETS can be unstable because of the unpredictable behaviors of the sampling based optimizer.



(a) trajectory 1 (b) trajectory 2

Fig. 5: Trajectory tracking

TABLE IV: Performance of trajectory tracking

methods	o.IMFlow	c.IMFlow	MPC	APG	PETS
norm.RMSE	0.0205	0.0200	6.08e-4	0.0662	0.0234
com. time [ms]	10.5	14.9	1.81	9.97	124
retrain time /episode [ms]	-	45.8	-	132	4.39

V. CONCLUSIONS AND FUTURE WORKS

Conclusions: We explore the potential of conditional normalizing flows for probabilistic inverse modeling of black-box systems. Firstly, we construct a novel architecture IMFlow for identifying input sequences as time series from observed system responses. Secondly, the open-loop IMFlow achieves close accuracy as MPC and much higher accuracy in comparison to the cNF. Besides, IMFlow outperforms MPC for larger horizon sizes in terms of both accuracy and computation time, making it appropriate for fast planning with uncertainty quantification. However, in comparison to APG, IMFlow is less accurate for larger horizon and takes much longer computational time due to its complexity. Thirdly, the closed-loop IMFlow improves the robustness against noise in the system. The architecture is able to filter some weak noise and regularize the prediction because of the Bayesian inference characteristics. Fourthly, the closed-loop IMFlow enables precise control tasks by adapting the partial networks online efficiently.

Future works: The proposed model-free online-RL IMFlow exhibits its capability for adaptive model predictive control. The accuracy of trajectory tracking is comparable with the RL-based methods. However, the efficiency should be further improved. Besides, so far, the evaluation is based solely on simulation models, it could be challenging to apply IMFlows in real systems. Furthermore, recent researches on learning-based MPC in grey-box environments reveal the combination of first-principle models and data-driven methods can improve the prediction accuracy while accelerating online calculation. We will investigate the possibility of incorporating IMFlow with physics-based models. Last but not the least, like most data-driven model predictive controllers, the stability of our proposed methods are not proved. There is no theoretical guarantee that the control system's states converge to desired points. It should be considered in our future research.

VI. ACKNOWLEDGMENTS

We would like to thank our colleague Damian Boborzi genuinely for generously sharing his insightful idea and expertise about reinforcement learning with us. This research was supported by the OpenSCALING project, which is funded by the German Federal Ministry of Education and Research (BMBF) under grant number 01IS23062H.

REFERENCES

[1] B. M. Chen, T. H. Lee, and V. Venkataramanan, "System identification techniques," in *Hard Disk Drive Servo Systems*. London: Springer London, 2002, pp. 15–27.

[2] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5-6, pp. 1327–1349, 2021.

[3] T. Uhl, "The inverse identification problem and its technical application," *Archive of Applied Mechanics*, vol. 77, pp. 325–337, 2007.

[4] E. A. M. Perez and H. Iba, "Deep learning-based inverse modeling for predictive control," *IEEE Control Systems Letters*, vol. 6, pp. 956–961, 2021.

[5] N. Wang, H. Chang, and D. Zhang, "Deep-learning-based inverse modeling approaches: A subsurface flow example," *Journal of Geophysical Research: Solid Earth*, vol. 126, no. 2, 2021.

[6] C. Wanigasekara, A. Swain, S. K. Nguang, and B. Gangadhara Prusty, "Neural network based inverse system identification from small data sets," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–6.

[7] N. A. S. Alwan and Z. M. Hussain, "Deep learning for robust adaptive inverse control of nonlinear dynamic systems: Improved settling time with an autoencoder," *Sensors*, vol. 22, no. 16, 2022.

[8] J. Kocijan, R. Murray-Smith, C. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *Proceedings of the 2004 American Control Conference*, vol. 3, 2004, 2214–2219 vol.3.

[9] M. Maiworm, D. Limon, and R. Findeisen, "Online learning-based model predictive control with gaussian process models and stability guarantees," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8785–8812, 2021.

[10] S. Zhao, F. Yi, Q. Wang, and X. Wang, "Active learning gaussian process model predictive control method for quadcopter," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 2664–2669.

[11] Y. Lin, D.-D. Chen, M.-S. Chen, X.-M. Chen, and J. Li, "A precise bp neural network-based online model predictive control strategy for die forging hydraulic press machine," *Neural Computing and Applications*, vol. 29, pp. 585–596, 2018.

[12] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz, "Deep model predictive flow control with limited sensor data and online learning," *Theoretical and Computational Fluid Dynamics*, vol. 34, no. 4, pp. 577–591, Mar. 2020.

[13] N. Wagener, C.-A. Cheng, J. Sacks, and B. Boots, *An online learning approach to model predictive control*, 2019. arXiv: `1902.08967 [cs.RO]`.

[14] H. Shah and M. Gopal, "Model-free predictive control of nonlinear processes based on reinforcement learning," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 89–94, 2016.

[15] T. H. Oh and J. M. Lee, "Q-mpc: Integration of reinforcement learning and model predictive control for safe learning," in *14th International Symposium on Process Systems Engineering*, vol. 49, Elsevier, 2022, pp. 47–55.

[16] J. Wu, Z. Jin, A. Liu, L. Yu, and F. Yang, "A hybrid deep-q-network and model predictive control for point stabilization of visual servoing systems," *Control Engineering Practice*, vol. 128, pp. 105–314, 2022.

[17] X. Liu, L. Qiu, Y. Fang, K. Wang, Y. Li, and J. Rodríguez, "Predictive control of voltage source inverter: An online reinforcement learning solution," *IEEE Transactions on Industrial Electronics*, pp. 1–10, 2023.

[18] A. S. Morgan, D. Nandha, G. Chalvatzaki, C. D'Eramo, A. M. Dollar, and J. Peters, "Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6672–6678.

[19] X. Yang, H. Zhang, Z. Wang, H. Yan, and C. Zhang, "Data-based predictive control via multistep policy gradient reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 2818–2828, 2023.

[20] Z. Xie, X. Huang, S. Luo, R. Zhang, and F. Ma, "A model predictive control trajectory tracking lateral controller for autonomous vehicles combined with deep deterministic policy gradient," *Transactions of the Institute of Measurement and Control*, 2023.

[21] A. Argenson and G. Dulac-Arnold, *Model-based offline planning*, 2021. arXiv: `2008.05556 [cs.LG]`.

[22] M. Henaff, A. Canziani, and Y. LeCun, *Model-predictive policy learning with uncertainty regularization for driving in dense traffic*, 2019. arXiv: `1901.02705 [cs.LG]`.

[23] C. Diehl, T. S. Sievernich, M. Krüger, F. Hoffmann, and T. Bertram, "Uncertainty-aware model-based offline reinforcement learning for automated driving," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1167–1174, 2023.

[24] K. Chua, R. Calandra, R. McAllister, and S. Levine, *Deep reinforcement learning in a handful of trials using probabilistic dynamics models*, 2018. arXiv: `1805.12114 [cs.LG]`.

[25] J. Deng, S. Sierla, J. Sun, and V. Vyatkin, "Offline reinforcement learning for industrial process control: A case study from steel industry," *Information Sciences*, vol. 632, pp. 221–231, 2023.

[26] J. Ruan, U. Inyang-Udohl, B. Nooning, *et al.*, "Offline model-free reinforcement learning with a recurrent augmented dataset for highly transient industrial processes," in *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–7.

[27] N. Wiedemann, V. Wüest, A. Loquercio, M. Müller, D. Floreano, and D. Scaramuzza, *Training efficient controllers via analytic policy gradient*, 2023. arXiv: `2209.13052 [cs.RO]`.

[28] J. Arroyo, C. Manna, F. Spiessens, and L. Helsen, "Reinforced model predictive control (rl-mpc) for building energy management," *Applied Energy*, vol. 309, pp. 118–346, 2022.

[29] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of an unmanned quadrotor," *Journal of Intelligent & Robotic Systems*, vol. 88, pp. 147–162, 2017.

[30] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, *Model-based reinforcement learning: A survey*, 2022. arXiv: `2006.16712 [cs.LG]`.

[31] M. Janner, J. Fu, M. Zhang, and S. Levine, *When to trust your model: Model-based policy optimization*, 2021. arXiv: `1906.08253 [cs.LG]`.

[32] A. B. Kordabad, D. Reinhardt, A. S. Anand, and S. Gros, "Reinforcement learning for mpc: Fundamentals and current challenges," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 5773–5780, 2023, 22nd IFAC World Congress.

[33] Q. Huang, "Model-based or model-free, a review of approaches in reinforcement learning," in *2020 International Conference on Computing and Data Science (CDS)*, 2020, pp. 219–221.

[34] S. Sujit, P. H. M. Braga, J. Bornschein, and S. E. Kahou, *Bridging the gap between offline and online reinforcement learning evaluation methodologies*, 2023. arXiv: `2212.08131 [cs.LG]`.

[35] S. Brandi, M. Fiorentini, and A. Capozzoli, "Comparison of online and offline deep reinforcement learning with model predictive control for thermal energy management," *Automation in Construction*, vol. 135, pp. 104–128, 2022.

[36] J. Liang, R. He, and T. Tan, *A comprehensive survey on test-time adaptation under distribution shifts*, 2023. arXiv: `2303.15361 [cs.LG]`.

[37] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018, pp. 25–45.

[38] D. D. Dunlap, E. G. Collins Jr, and C. V. Caldwell, "Sampling based model predictive control with application to autonomous vehicle guidance," in *Florida Conference on Recent Advances in Robotics*, 2008.

[39] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[40] C. Pinneri, S. Sawant, S. Blaes, *et al.*, "Sample-efficient cross-entropy method for real-time planning," in *Conference on Robot Learning*, PMLR, 2021, pp. 1049–1065.

[41] J. Sacks and B. Boots, *Learning sampling distributions for model predictive control*, 2022. arXiv: `2212.02587 [cs.RO]`.

[42] T. Power and D. Berenson, *Variational inference mpc using normalizing flows and out-of-distribution projection*, 2022. arXiv: `2205.04667 [cs.RO]`.

[43] T. Power and D. Berenson, "Learning a generalizable trajectory sampling distribution for model predictive control," *IEEE Transactions on Robotics*, pp. 1–18, 2024.

[44] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 3964–3979, Nov. 2021.

[45] C. C. Margossian and D. M. Blei, *Amortized variational inference: When and why?* 2024. arXiv: `2307.11018 [stat.ML]`.

[46] S. T. Radev, U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe, *Bayesflow: Learning complex stochastic models with invertible neural networks*, 2020. arXiv: `2003.06281 [stat.ML]`.

[47] G. I. Beintema, M. Schoukens, and R. Tóth, *Deep subspace encoders for nonlinear system identification*, 2023. arXiv: `2210.14816 [eess.SY]`.

[48] F. Gabbiani and S. Cox, *Mathematics for Neuroscientists* (Elsevier science & technology books). Elsevier Science, 2010, pp. 251–266.

[49] L. Dinh, J. Sohl-Dickstein, and S. Bengio, *Density estimation using real nvp*, 2017. arXiv: `1605.08803 [cs.LG]`.