Neural-iLQR: A Learning-Aided Shooting Method for Trajectory Optimization

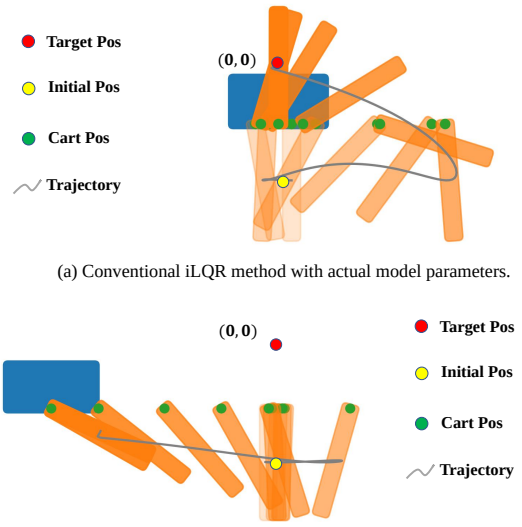
Zilong Cheng[1,*], Yulin Li[2,*], Kai Chen[2], Jun Ma[2], Tong Heng Lee[1]

***Abstract*— Iterative linear quadratic regulator (iLQR) has gained wide popularity in addressing trajectory optimization problems with nonlinear system models. However, as a model-based shooting method, it relies heavily on an accurate system model to update the optimal control actions and the trajectory determined with forward integration, thus becoming vulnerable to inevitable model inaccuracies. Recently, substantial research efforts in learning-based methods for optimal control problems have been progressing significantly in addressing unknown system models, particularly when the system has complex interactions with the environment. Yet a deep neural network is normally required to fit substantial scale of sampling data. In this work, we present Neural-iLQR, a learning-aided shooting method over the unconstrained control space, in which a neural network with a simple structure is used to represent the local system model. In this framework, the trajectory optimization task is achieved with simultaneous refinement of the optimal policy and the neural network iteratively, without relying on the prior knowledge of the system model. Through comprehensive evaluations on two illustrative control tasks, the proposed method is shown to outperform the conventional iLQR significantly in the presence of inaccuracies in system models.**

I. Introduction

The last decade has witnessed substantial achievements in the context of trajectory optimization, pervading different application domains including unmanned aerial vehicles (UAVs) [1], autonomous driving [2], quadrupeds [3], mobile manipulators [4], etc. However, it is still an open and challenging problem on the generation of a satisfying trajectory in complex scenarios, especially when the state/control space is inherently high-dimensional, the system model is nonlinear, and the non-smooth contact and constraints are introduced in most of the robot systems [4]–[6]. For such problems, the model-based method and the learning-based method are extensively investigated.

In terms of the model-based method in trajectory optimization, a second-order shooting method called differential dynamic programming (DDP) [7] has gained popularity in the robotics community due to its high efficiency in dealing with nonlinear dynamic systems. In each stage, with the quadratic approximation of the system dynamics around the nominal state-input trajectory, it drives the current input trajectory towards the optimal direction. To further reduce the computation time, iterative linear quadratic regulator (iLQR) was proposed as a variant of DDP following the similar structure but using first-order approximation of the system dynamics instead. Remarkably, iLQR has been successfully applied to robot systems with precise models [8]–[10]. However, considering complex dynamics of a robot system, the inevitable existence of model inaccuracy could lead to the deviation of obtained solution from the optimal control policy. Fig. 1 shows a typical failure case in model-based iLQR when modeling inaccuracy is introduced.



(a) Conventional iLQR method with actual model parameters.

(b) Conventional iLQR method with inaccurate model parameters.

Fig. 1. Visualization of the trajectory(middle point of the pole) generated by iLQR method in cartpole control problem using numerical simulation. In this problem, the cart moves along the horizontal line to keep the pole vertical. (a) shows the result of conventional iLQR with accurate model parameters, while (b) fails due to model inaccuracy. Opacity of the pole increases over time. For the cart, only the final position is shown while middle ones are represented by the green points for clearance.

On the other hand, model-free methods have recently shown promising results in learning the system model and the optimal policy. Generally, a neural network is constructed and trained to generate the dynamic model or optimal policy in different ways. In [11], the control problem is modeled as a Markov Decision Process (MDP) [12] and optimal policy is then learned using Reinforcement Learning (RL) [13]. In order to learn long-term transition dynamics instead of step transitions, latent variables recurrent network is utilized to improve the performance in long prediction horizon [14]. Although learning-based methods outperform model-based methods in its versatility when dealing with complex control problems, deep neural networks and substantial scale of exploration samples are normally required to reach the

* indicates equal contribution.
[1] Zilong Cheng and Tong Heng Lee are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (e-mail: zilongcheng@u.nus.edu; eleleeth@nus.edu.sg).
[2] Yulin Li, Kai Chen, and Jun Ma are with The Hong Kong University of Science and Technology, China (e-mail: yline@connect.ust.hk; kchen916@connect.hkust-gz.edu.cn; jun.ma@ust.hk).

satisfying solution.

To deal with the aforementioned drawbacks, a learning-aided shooting method named Neural-iLQR is presented for trajectory optimization over the unconstrained control space, which avoids the requirement of any prior knowledge of the system model in trajectory optimization tasks. In this approach, random trials are performed such that a dataset is collected, and then a filtered neural network is used to fit the measurement data locally in the current iteration. Subsequently, the well-trained neural network is implemented in the execution of the backward pass of iLQR method such that the trajectory optimization problem is solved iteratively. Moreover, we evaluate several critical factors pertinent to the optimization results by comparing the performance of different network structures and other related parameters. Finally, we demonstrate that the online-retraining mechanism could prevent the optimization process from trapping into the local minimum, whereby the optimality of the obtained trajectory can be further improved compared to conventional iLQR method.

II. RELATED WORK

Similar framework utilizing iLQR has been applied on complex robot systems, and these works prominently show the high applicability of the iLQR scheme. In [15], a locally weighted projection regression (LWPR) method is implemented to complete the non-parametric model identification, and then the iLQR method is performed as a feedforward controller to realize the objective of tracking. In [16], a constrained, time-varying LQR problem is solved based on the quadratic cost function and the linearized system using Sequential Linear Quadratic (SLQ) algorithm (a continuous-time version of iLQR) [17] for a mobile manipulator. In general, a nominal system model is required in the implemention of these model-based methods.

To improve the robustness toward system modeling inaccuracy, several closely related works are developed very recently. In [18], an iLQR framework called the curious iLQR is proposed, in which the system dynamics is reflected by the Bayesian modeling. In [19], a multi-layer neural network is used to model the dynamics of off-road and on-road vehicles which is then used in iLQR controller. These methods are not sample efficient due to the large size of the built neural network. A recent work combines iLQR and RL in an augmenting way [20], which uses RL-based terminal cost and shortens the iLQR horizon adaptively in each stage to relax the requirement of an accurate system model. However, it stills requires a nominal system model to drive the optimization process.

III. NEURAL-ILQR

A. Problem Definition

Generally, the trajectory planning problem can be expressed as a nonlinear optimization problem given in (1), where $x(\tau) \in \mathbb{R}^n$ and $u(\tau) \in \mathbb{R}^m$ denote the state and action at the time stamp τ, respectively; $J_\tau\big(x(\tau), u(\tau)\big)$ denotes the cost at each time stamp τ with respective to the pair of state and action; ϕ_T represents the terminal cost at the time stamp T; $f\big(x(\tau), u(\tau)\big)$ is the dynamic function of the system which is not restricted to be linear; x_0 denotes the initial state of the system.

$$\begin{aligned} \underset{\big(x(\tau),u(\tau)\big)\in\mathbb{R}^n\times\mathbb{R}^m}{\text{minimize}} \quad & \phi_T(x(T)) + \sum_{\tau=0}^{T-1} J_\tau\big(x(\tau), u(\tau)\big) \\ \text{subject to} \quad & x(\tau+1) = f\big(x(\tau), u(\tau)\big) \\ & \tau = 0, 1, \cdots, T-1 \\ & x(0) = x_0, \end{aligned} \tag{1}$$

B. Overview of Neural-iLQR

With the development of the conventional model-based iLQR, Neural-iLQR is proposed in this section with a detailed analysis to solve the optimization problem (1). Essentially, Neural-iLQR utilizes the neural network to learn the dynamic function and procedures of iLQR can be performed entirely based on the measurement data with the incorporation of the neural network.

As shown in Fig. 2, the implementation of Neural-iLQR is summarized. In the beginning, an empty dataset is required to be initialized with p random trials of prediction horizon T performed in the dynamic system (i.e., the system runs arbitrarily with a sequence of control actions chosen randomly). Upon completion of the dataset initialization, a neural network structure can be chosen to fit the system dynamic function. After the setup, we conduct the backward pass and forward pass iteratively following the conventional DDP/iLQR method except for that the gradient and hessian of the dynamic function are replaced by the analytical derivatives of the neural network. Moreover, an online data collection and neural network retraining mechanisms are incorporated.

C. Backward Pass of Neural-iLQR

The primary idea behind the backward pass of DDP/iLQR can be straightforwardly represented by the Bellman equation given by

$$V_\tau\big(x(\tau)\big) = \min_{u(\tau)} \Big\{ J_\tau\big(x(\tau), u(\tau)\big) + V_{\tau+1}\Big(f\big(x(\tau), u(\tau)\big)\Big) \Big\}, \tag{2}$$

where $V_\tau(x(\tau))$ and $V_{\tau+1}(f(x(\tau), u(\tau)))$ denote the value functions with respect to the current state $x(\tau)$ and the state at the next time stamp $x(\tau+1)$, respectively.

The perturbed Q-function is then introduced [9] and approximated by the second-order Taylor expansion as in (3), where $\delta x(\tau)$ and $\delta u(\tau)$ represent the amount of change with respect to the nominal state and action at the time τ.

$$\begin{aligned} & Q_\tau\big(\delta x(\tau), \delta u(\tau)\big) \\ & \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta x(\tau) \\ \delta u(\tau) \end{bmatrix}^T \begin{bmatrix} 0 & \big(Q_\tau^T\big)_x & \big(Q_\tau^T\big)_u \\ (Q_\tau)_x & (Q_\tau)_{xx} & (Q_\tau)_{xu} \\ (Q_\tau)_u & (Q_\tau)_{ux} & (Q_\tau)_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x(\tau) \\ \delta u(\tau) \end{bmatrix}, \end{aligned} \tag{3}$$

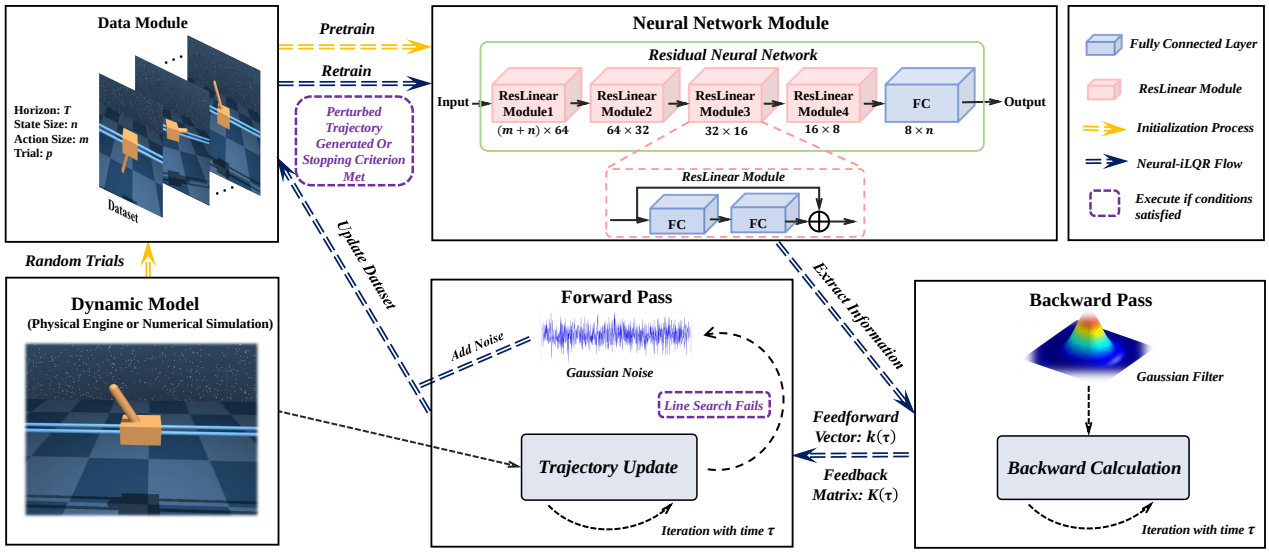


Fig. 2. Overview of the Neural-iLQR pipeline. After the initialization (yellow arrow) of the dataset and neural network, the information flow of Neural-iLQR (deep blue arrow) forms a closed loop. We use MuJoCo [21] physics engine and numerical simulation to simulate the dynamics of the system. Gaussian noise and retraining of the neural network are only needed when the criterion (purple box) are satisfied in the Neural-iLQR loop.

where

$$\begin{aligned}
(Q_\tau)_x &= (J_\tau)_x + N_x^\top (V_{\tau+1})_x \\
(Q_\tau)_u &= (J_\tau)_u + N_u^\top (V_{\tau+1})_x \\
(Q_\tau)_{xx} &= (J_\tau)_{xx} + N_x^\top (V_{\tau+1})_{xx} N_x + (V_{\tau+1})_x \cdot N_{xx} \\
(Q_\tau)_{ux} &= (J_\tau)_{ux} + N_u^\top (V_{\tau+1})_{xx} N_x + (V_{\tau+1})_x \cdot N_{ux} \\
(Q_\tau)_{uu} &= (J_\tau)_{uu} + N_u^\top (V_{\tau+1})_{xx} N_u + (V_{\tau+1})_x \cdot N_{uu}.
\end{aligned} \tag{4}$$

We denote the neural network that fits the system dynamic function f as N. The optimal solution to the perturbed control action $\delta u(\tau)^*$ at the time stamp τ can then be calculated by

$$\delta u(\tau)^* = \underset{\delta u(\tau)}{\operatorname{argmin}} \quad Q_\tau\big(\delta x(\tau), \delta u(\tau)\big), \tag{5}$$

which gives

$$\delta u(\tau)^* = k(\tau) + K(\tau)\delta x(\tau), \tag{6}$$

where $k(\tau) \in \mathbb{R}^m$ and $K(\tau) \in \mathbb{R}^{m\times n}$ are the feedforward vector and feedback matrix for the perturbed Q-function at the time stamp τ, respectively, and they can be explicitly represented by

$$k(\tau) = -\left(Q_\tau\right)_{uu}^{-1} \left(Q_\tau\right)_u \tag{7a}$$

$$K(\tau) = -\left(Q_\tau\right)_{uu}^{-1} \left(Q_\tau\right)_{ux}. \tag{7b}$$

It is worth noting that using a neural network to learn a dynamic system with high dimensions is challenging. The gradient of the neural network model contains a large amount of noise and ineffective information compared to the gradient of the dynamic model, for which we apply a Gaussian filter to smooth the trained neural network and extract useful information. Besides, we ignore N_{xx}, N_{ux} and N_{uu} due to the significant noise resulted from second-order derivatives of the neural network.

D. Forward Pass of Neural-iLQR

The last and the principal step in Neural-iLQR is the forward pass. An indispensable line search strategy is performed at the beginning of the forward pass to find a trajectory with better performance based on the given feedback matrices and feedforward vectors at different time stamps. The basic idea of the line search [7] can be realized by

$$\delta u(\tau) = \alpha k(\tau) + K(\tau)\delta x(\tau), \tag{8}$$

where α is the step size parameter to be determined in the line search iteration. Then the forward roll-out is conducted by feeding the perturbed action sequence to the system dynamic model f:

$$\begin{aligned}
u(\tau) \quad &= \hat{u}(\tau) + \delta u(\tau) \\
x(\tau+1) &= f\big(x(\tau), u(\tau)\big).
\end{aligned} \tag{9}$$

The nominal trajectory $(\hat{x}, \hat{u})$ can be updated to a new feasible trajectory obtained in the forward pass, and the updated feasible trajectory can be used to initiate the next Neural-iLQR iteration and retrain the neural network.

The stopping criterion for one iteration can be chosen as the difference of the objective function value between the current and the latest nominal trajectory, once the criterion is satisfied, we will retrain the neural network on the updated dataset. For the scenario when it is still not possible to improve the dynamic system performance as the maximum number of line search iterations is reached. The solving process is considered as trapping into local minimum in this situation. We add some perturbations to the sequence of

actions when generating the current trajectory, and the neural network is then retrained to escape from the local minimum. In this way, the trajectory could continue converging.

IV. Experiments

In this section, we comprehensively evaluate Neural-iLQR using numerical simulation and MuJoCo [21] physic engine with two illustrative examples: a vehicle tracking problem and a cartpole control problem. To demonstrate the practicality and effectiveness of Neural-iLQR, we use the conventional iLQR method as the benchmark to show that the proposed model-free method is comparable to or even better than the model-based counterpart. Particularly, the robustness of our method is testified when model inaccuracy is introduced. Furthermore, the influence of several critical factors in Neural-iLQR is discussed thoroughly.

A. System Setups

1) Vehicle Tracking Problem Formulation: The state vector of the vehicle is defined as $x = [p_x \quad p_y \quad \theta \quad v]^T$, where p_x and p_y denote the position of the center of the rear axis in the Cartesian coordinates; θ denotes the heading angle of the vehicle; v denotes the velocity of the vehicle. The action vector is defined as $u = [\omega \quad a]^T$, where ω denotes the steering angle and a denotes the acceleration. Details of the derivation of dynamic function could be refereed from [7]. The control target is to drive the vehicle in a straight line at a specified speed, and the reference trajectory is chosen as $r = [0 \quad -10 \quad 0 \quad 8]^\top$; the objective function is chosen as a typical linear quadratic form with constant weighting matrices Q and R and constant reference trajectory r:

$$J = \sum_{\tau=0}^{T} \Big(\big(x(\tau) - r\big)^\top Q \big(x(\tau) - r\big) + u(\tau)^\top R u(\tau) \Big). \quad (10)$$

2) Cartpole Control Problem Formulation: The state vector of the cartpole is defined as $x = [\theta \quad \omega \quad p \quad v]^\top$, where θ and ω denote the angle between the pole and the vertical direction and its corresponding angular velocity, respectively; p and v denote the position and the velocity of the cart, respectively. The control action F means the force applied on the cart horizontally. The system dynamic function follows [22], and the detailed derivation is skipped here for brevity. We are going to swing up the pole and keep it balanced around the upward position by applying a force to the cart; the objective function is chosen as a typical linear quadratic form with the terminal state penalty:

$$J = x(T)^\top Q_T x(T) + \sum_{\tau=0}^{T-1} \Big(x(\tau)^\top Q x(\tau) + u(\tau)^\top R u(\tau) \Big). \quad (11)$$

Noted that dynamic functions of the systems are only required for calculation in conventional model-based iLQR, and it can be fully replaced by the simulation environment and the neural network in the implementation of Neural-iLQR.

B. Overall Performance of Neural-iLQR

Provided with accurate dynamic model, the conventional iLQR method is proven to achieve effective optimization outcomes in such two examples indicated in Fig. 3. After several iterations, the trajectory will converge asymptotically, and the objective function is reduced to a satisfying point; thus, it is a good benchmark for comparison with our proposed model-free method. The optimal objective function values reached by the conventional model-based iLQR are 10192 and 1642 respectively for vehicle tracking and cartpole control examples.

As observed from Fig. 3, the proposed model-free Neural-iLQR method can successfully generate a reliable trajectory for the control problem, attaining objective function values at 992.9 and 1178, which shows comparable optimization performance compared to the conventional iLQR method. We apply simple neural networks to fit the local trajectory data in the current iteration and use its filtered gradient information to guide the optimization directions. When it gets stuck at some local minimum points using the current neural network, the online data collection and retraining mechanisms discussed in Section III-D empower the proposed method with the ability to avoid trapping into the local minimum. It renders possibility for the Neural-iLQR to performance even better than the conventional model-based iLQR as shown in Fig. 3.

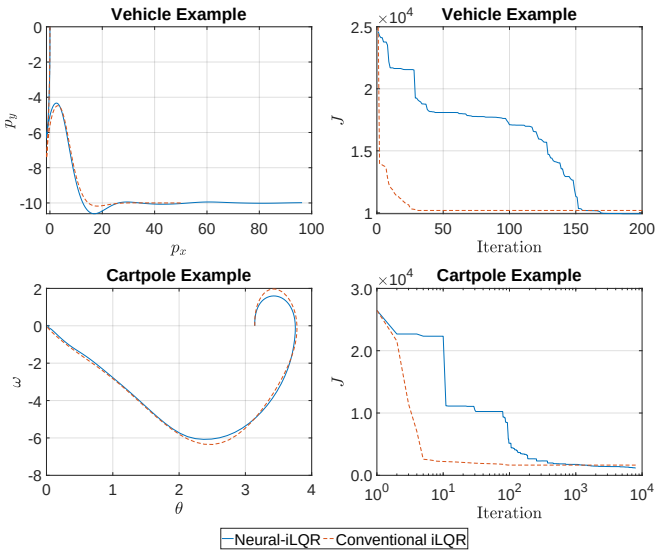


Fig. 3. Comparison of the trajectory and objective function value generated with Neural-iLQR and the conventional iLQR. residual neural network is applied in this experiment, the trials number is 100, and the standard deviation of the Gaussian filter is chosen as 5.

C. Comparison of Effects on Critical Factors

The influence of the selected three critical factors in the Neural-iLQR architecture is discussed in this section, and we find that these critical factors could directly affect the optimization performance of Neural-iLQR, and thus it renders possibility to continuously improve the results by adjusting these factors in the proposed architecture.

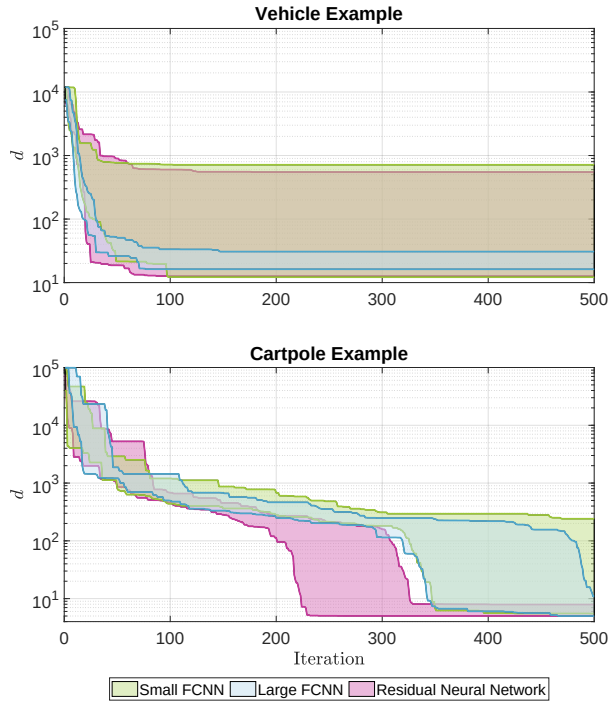


Fig. 4. Objective function value in iterations using Neural-iLQR with different neural network structure.

1) Neural Network Architecture: We propose three neural network structures and demonstrate the performance attained by each in this section. The first two are fully connected neural networks (FCNN) with two hidden layer and one output layer, structures are shown in Table I.

TABLE I
LAYER SIZE FOR THE FULLY CONNECTED NEURAL NETWORK

Network	First Hidden Layer	Second Hidden Layer	Output Layer
Small FCNN	$[(m+n)\times 128]$	$[128\times 64]$	$[64\times n]$
Large FCNN	$[(m+n)\times 1024]$	$[1024\times 512]$	$[512\times n]$

Shown in Fig. 2, the third type of the neural network is a residual neural network. It consists of four ResLinear modules and one fully connected linear output layer. Batch normalization and ReLU are required before each fully connected layer.

We define the deviation between the objective function values obtained by Neural-iLQR and the conventional iLQR method given the dynamic model during iterations as d and the iteration number when it first reaches the minimum objective function value as k. Neural-iLQR is randomly performed five times with each neural network structure, and the conventional iLQR method is performed with the same parameters as used in Neural-iLQR for the fair comparision.

Fig. 4 shows the deviation of the Neural-iLQR method with respect to different neural network structures after 500 Neural-iLQR iterations, respectively. From the results in Table II, it can be seen that the large FCNN uses less time than the other two in pretraining. Moreover, Neural-iLQR method with all the three neural network structures

TABLE II
COMPARISONS OF DIFFERENT NEURAL NETWORK ARCHITECTURES

	Network Structure	Pretraining Time (s)	d_{min}	d_{avg}	k_{min}
Vehicle Tracking	Small FCNN	246.3256	13	295	98
	Large FCNN	**121.0006**	16.2	**24**	77
	Residual Neural Network	205.6572	**12.7**	125	87
Cartpole Control	Small FCNN	63.7257	6	53	340
	Large FCNN	**28.1529**	5	**7**	350
	Residual Neural Network	210.2348	5	**6**	**220**

can achieve satisfying performance in terms of the objective function value but lead to various performance. The large FCNN structure shows the highest performance in the vehicle tracking example, but the residual neural network structure demonstrates the best performance both in its optimality and fast convergence in the cartpole control example. Basically, both the large FCNN structure and residual neural network structure provide satisfying results and show higher performance than the small FCNN structure as they can achieve better optimization results with fewer iterations.

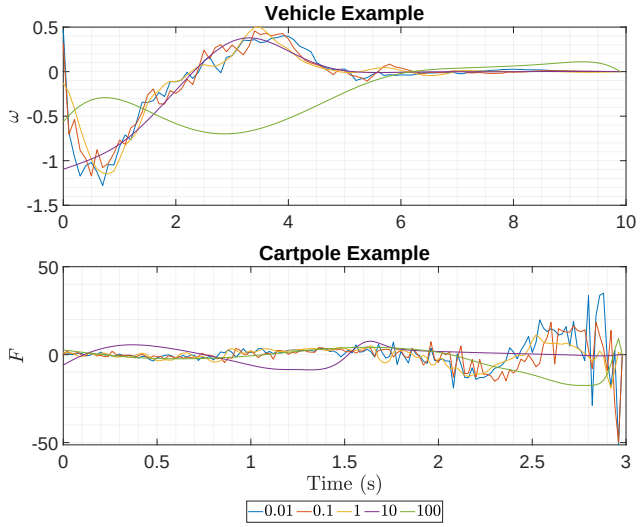


Fig. 5. Control action sequence obtained by Neural-iLQR with different standard deviation parameters in the Gaussian filter.

2) Gaussian Filter: Fig. 5 shows the control action sequences obtained by applying Gassuian filter with different standard deviation. We can see that the smoothness of the control signals in the two illustrative examples is significantly improved with larger standard deviation of the Gaussian filter. However, to the best knowledge of the authors, an extremely large standard deviation parameter also leads to unsatisfying optimization results due to the lack of sudden changes in control actions.

3) Number of Trials in Dataset: Fig. 6 shows the deviation of Neural-iLQR with different numbers of trials. From Fig. 6, it shows that the convergence of the obtained trajectory could usually be improved with larger number of trials. However, an extremely large number of trials can lead to difficulty in training the neural network. The experiments in the two examples show that it is usually rational to choose the number of trials to be around 100.

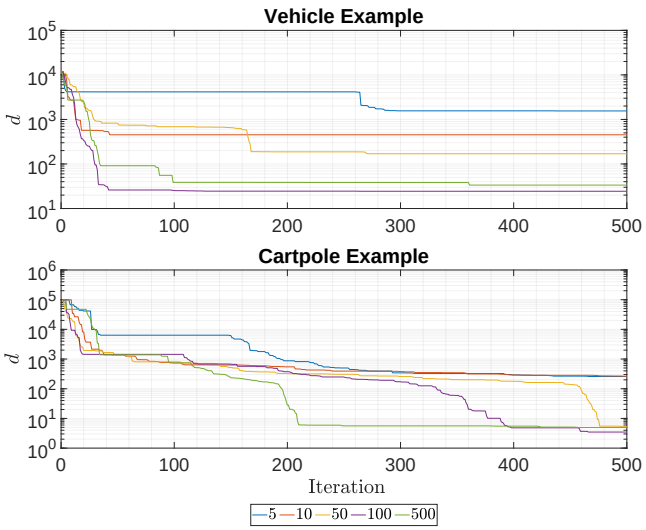


Fig. 6. Deviation of the objective function value from the conventional iLQR method by using Neural-iLQR with different number of trials.

TABLE III
COMPARISON OF NEURAL-ILQR AND MODEL-BASED ILQR FOR CARTPOLE CONTROL EXPERIMENT

Model Inaccuracy	Model-based iLQR			Neural-iLQR		
	Success	θ_{error}	Obj.Val ($\times 10^3$)	Success	θ_{error}	Obj.Val ($\times 10^3$)
0%	Yes	**4.750**	**2.334**	Yes	7.051	3.269
20%	Yes	7.872	3.436	Yes	**6.965**	**2.477**
40%	No	9.456	9.833	Yes	**7.984**	**3.458**
60%	No	9.287	12.699	Yes	**7.657**	**3.038**

D. Robustness to Modeling Inaccuracy

In this section, we further demonstrate the robustness of the proposed method against model inaccuracy compared to the conventional iLQR method in MuJoCo. We build a cartpole model and use the simulation model to conduct forward dynamics by feeding inputs to the environment. We can obtain the trajectory and collect the real-time data for training in Neural-iLQR, which is convincing and close to the scenarios in the real world. Modeling inaccuracy is inevitable in the real-world situation, in this case, we introduce the model inaccuracy to the system by adjusting the model parameter in MuJoCo.

As indicated in Fig. 7 and Table III, the conventional model-based iLQR shows its effectiveness in achieving satisfying optimization results with the accurate model. The objective function value reaches 2.334 and the mean square error (MSE) of θ for the generated trajectory is 4.750. However, its optimization performance will be significantly affected by the modeling inaccuracy and it may even fail when large inaccuracy is introduced. Meanwhile, we can see from the results in Table III that the proposed Neural-iLQR method shows comparable capability in generating the optimal trajectory towards the control target without any prior knowledge of the system model, and its robustness and adaptability to inaccurate model is demonstrated.

V. CONCLUSION

This paper investigates the development of Neural-iLQR, a learning-aided shooting method for trajectory optimization.

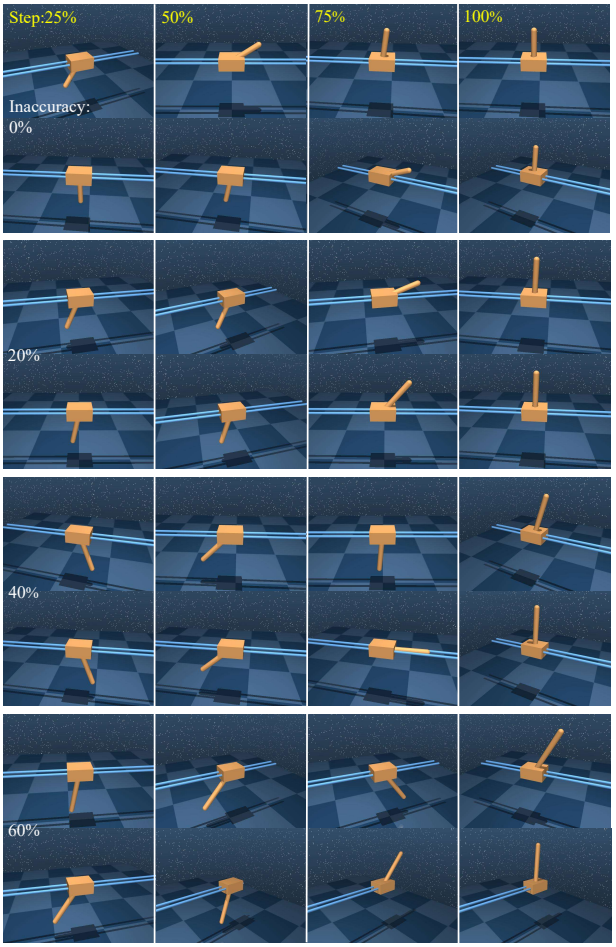


Fig. 7. Visualization of the comparisons between the conventional model-based iLQR (above) and the proposed Neural-iLQR (below) under different level of model inaccuracy.

In view of an unknown dynamic system, a neural network is utilized to fit the dynamic function in an iterative framework, which enables the use of the iLQR method in trajectory planning problems. The estimated gradient matrix of the dynamic function is derived, and the improved feedforward iteration is proposed to deal with the inaccuracy and imprecision in the optimization problem. As a result, the refined iLQR method can be applied completely without any prior information of the dynamic system. Moreover, the trajectory resulted from the Neural-iLQR method can be even better than the conventional iLQR method, as the local optimal point can be escaped with the deployment of the further exploration procedure. Finally, illustrative examples are used to validate the performance of the proposed Neural-iLQR method and detailed discussions are presented. It is worthwhile to highlight that due to the effectiveness and the universality of the proposed architecture, the framework could be suitably adjusted or extended to address practical issues in many real-world applications.

REFERENCES

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[2] W. Xu, Q. Wang, and J. M. Dolan, "Autonomous vehicle motion planning via recurrent spline optimization," in *2021 IEEE International Conference on Robotics and Automation*, 2021, pp. 7730–7736.

[3] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[4] J. P. Sleiman, F. Farshidian, and M. Hutter, "Constraint handling in continuous-time ddp-based model predictive control," in *2021 IEEE International Conference on Robotics and Automation*, 2021, pp. 8209–8215.

[5] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time UAV replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 1208–1214.

[6] F. Eiras, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "A two-stage optimization-based motion planner for safe urban driving," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 822–834, 2022.

[7] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation*, 2014, pp. 1168–1175.

[8] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *2004 International Conference on Informatics in Control, Automation and Robotics*. Citeseer, 2004, pp. 222–229.

[9] H. Zhang, J. Gong, Y. Jiang, G. Xiong, and H. Chen, "An iterative linear quadratic regulator based trajectory tracking controller for wheeled mobile robot," *Journal of Zhejiang University SCIENCE C*, vol. 13, no. 8, pp. 593–600, 2012.

[10] J. Ma, Z. Cheng, X. Zhang, M. Tomizuka, and T. H. Lee, "Alternating direction method of multipliers for constrained iterative LQR in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[11] W. Zhao, H. Liu, and F. L. Lewis, "Robust formation control for cooperative underactuated quadrotors via reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.

[13] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Belmont, MA, USA: Athena Scientific, 2012.

[14] N. R. Ke, A. Singh, A. Touati, A. Goyal, Y. Bengio, D. Parikh, and D. Batra, "Learning dynamics model in reinforcement learning by incorporating the long term future," *arXiv preprint arXiv:1903.01599*, 2019.

[15] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive optimal feedback control with learned internal dynamics models," in *From Motor Learning to Interaction Learning in Robots*. Berlin: Springer, 2010, vol. 264, pp. 65–84.

[16] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body MPC for a dynamically stable mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3687–3694, 2019.

[17] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation*, 2017, pp. 93–100.

[18] S. Bechtle, Y. Lin, A. Rai, L. Righetti, and F. Meier, "Curious iLQR: Resolving uncertainty in model-based RL," in *Conference on Robot Learning*. PMLR, 2020, pp. 162–171.

[19] A. Nagariya and S. Saripalli, "An iterative LQR controller for off-road and on-road vehicles using a neural network dynamics model," *arXiv preprint arXiv:2007.14492*, 2020.

[20] T. Zong, L. Sun, and Y. Liu, "Reinforced ilqr: A sample-efficient robot locomotion learning," in *2021 IEEE International Conference on Robotics and Automation*, 2021, pp. 5906–5913.

[21] 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.

[22] I. Osband, B. Van Roy, D. J. Russo, Z. Wen *et al.*, "Deep exploration via randomized value functions." *Journal of Machine Learning Research*, vol. 20, no. 124, pp. 1–62, 2019.