







# Learning-Based Balance Control of Wheel-Legged Robots

Leilei Cui , *Student Member, IEEE*, Shuai Wang , *Member, IEEE*, Jingfan Zhang , *Student Member, IEEE*, Dongsheng Zhang, Jie Lai, Yu Zheng , *Senior Member, IEEE*, Zhengyou Zhang , *Fellow, IEEE*, and Zhong-Ping Jiang , *Fellow, IEEE*

**Abstract**—This letter studies the adaptive optimal control problem for a wheel-legged robot in the absence of an accurate dynamic model. A crucial strategy is to exploit recent advances in reinforcement learning (RL) and adaptive dynamic programming (ADP) to derive a learning-based solution to adaptive optimal control. It is shown that suboptimal controllers can be learned directly from input-state data collected along the trajectories of the robot. Rigorous proofs for the convergence of the novel data-driven value iteration (VI) algorithm and the stability of the closed-loop robot system are provided. Experiments are conducted to demonstrate the efficiency of the novel adaptive suboptimal controller derived from the data-driven VI algorithm in balancing the wheel-legged robot to the equilibrium.

**Index Terms**—Machine learning for robot control, optimization and optimal control, wheeled robots.

## I. INTRODUCTION

WHEEL-legged robots combine both the mobility of mobile robots and the dexterity of legged robots, and therefore they can move fast on the flat ground and pass through the bumpy road [1]–[3]. However, for some wheel-bipedal robots (similar to the one shown in Fig. 1), because there are only two contact points between the robot and the ground, the wheel-legged robot is a non-minimum phase system. The balance control plays a pivotal role in the real-world applications of the wheel-legged robots. Since the wheel-legged robot is a nonlinear and underactuated system, balance controller design is a difficult task, and most existing works [2], [4]–[6] adopt model-based linear quadratic regulator (LQR) technique to solve the balance problem, of which the performance highly depends



Fig. 1. Our wheel-legged robot in a (a) squatting or (b) standing pose.

on the accuracy of the dynamic model. Due to the complex mechanical structure of the robot, it is hard to derive an accurate dynamic model for the wheel-legged robot. Besides, when the physical parameters of the robot change, such as the height of the robot, the parameters of the model-based controller have to be manually tuned again to guarantee the stability, which compromises the automation of the robot. These facts hinder the widespread application of conventional model-based balance control approaches. Although conventional adaptive control can tackle this problem, the optimality of the controller, which can endow the closed-loop system desirable transient performance, cannot be guaranteed. On the contrary, when the robot is running, data can be collected. Hence, developing a learning-based control method to optimally balance the wheel-legged robot is important, yet fundamentally challenging. The interested reader should consult [7] for a recent tutorial on learning-based control.

In this letter, based on RL and ADP [8]–[10], we propose a novel learning-based approach to solve the optimal balance control problem of the wheel-legged robot in the absence of the accurate dynamic model. Only the data of the wheel-legged robot, including the states and the control inputs, is required for the training process. In the last few decades, VI has been well studied for the systems described by Markov decision processes (MDPs) with the discrete time, state and action spaces [11]. For real-world robotics applications, the motion of the robot is often described by continuous-time differential equations with continuous and infinite state and action spaces. Discretization of the continuous system and space may increase computational burden, sacrifice system performance, and incur instability for the closed-loop system. Besides, for traditional RL

Manuscript received February 24, 2021; accepted July 12, 2021. Date of publication July 27, 2021; date of current version August 16, 2021. This work was supported in part by the National Science Foundation under Grant EPCN-1903781. The first two authors contributed equally to this work. (Corresponding author: Leilei Cui.)

Leilei Cui and Zhong-Ping Jiang are with the Control and Networks Lab, Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: l.cui@nyu.edu; zjiang@nyu.edu).

Shuai Wang, Dongsheng Zhang, Jie Lai, Yu Zheng, and Zhengyou Zhang are with the Tencent Robotics X, Tencent Holdings, Shenzhen, Guangdong 518057, China (e-mail: shuaiwanghit@gmail.com; donsenshang@tencent.com; cometlx@163.com; petezheng@tencent.com; zhengyou@tencent.com).

Jingfan Zhang is with the Department of Electrical & Electronic Engineering, School of Engineering, The University of Manchester, Manchester M13 9PL, U.K. (e-mail: jingfan.zhang@postgrad.manchester.ac.uk).

This letter has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2021.3100269>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3100269

algorithms, the convergence of the algorithm and the stability of the closed-loop system are not considered theoretically, a fact that prevents traditional learning algorithms from being applied to safety-critical robotics applications. With these observations in mind, we depart from the conventional RL algorithms [12], [13] and focus on the development of a new data-driven VI algorithm for a wheel-legged robot that is described by ordinary differential equations. The convergence of the data-driven VI algorithm and the stability of the robot in closed-loop with the adaptive optimal controller generated by the proposed algorithm are provided. Compared with the policy iteration approach [14], the VI algorithm does not require a stabilizing controller to initialize the training process, and it is more desirable for an uncertain system.

### A. Related Work

Recently, Boston Dynamics published a video [1], which shows a wheel-legged robots is capable of handling goods. There is a tail attached to the body of the robot to help balance. However, no technical details are disclosed. A common feature of most of the existing balance control methods for wheel-legged robots is that they are model-based approaches. Via passivity-based control, Wang *et al.* [3] have developed a nonlinear balance controller for the wheel-legged robot, and its performance is validated by experiments when the height of the robot changes. In [4], the dynamic model of the wheel-legged is derived and linearized. Then LQR is applied to calculate the optimal gain of the controller. In [2], [5], the whole body dynamics for the wheel-legged robot is derived. Then LQR is combined with the whole body dynamics to calculate the driving inputs at each joint. In [6], a two-wheeled inverted pendulum is considered as a template model for the wheel-legged robot and LQR is applied to design the tracking controller. Zambella *et al.* [15] propose a balance controller for the whole body dynamics. An unneglectable weakness of the aforementioned model-based balance control is that the performance of the balance controller highly depends on the accuracy of the dynamic model, but in practice, due to the complex mechanical structure of the robot, it is hard to derive such an accurate model. RL algorithms shed lights on this problem.

RL based controller design methods for robots have attracted considerable attention from both engineers and researchers recently. In [16], the RL technique is applied to tune the parameters of the impedance controller for robotic knee prosthesis. In [17], in order to suppress the vibration, the deep deterministic policy gradient (DDPG) algorithm [12] is used to design the controller for a flexible manipulator. In [18], DDPG is applied to learn the variable impedance controller for a manipulator for contact sensitive tasks. DDPG has also been combined with traditional control technique and been applied on the balance control of a mobile robot in [19]. Lee *et al.* [20] apply RL technique to contact-rich manipulation tasks. Hwangbo *et al.* [21] also apply RL algorithm to train a neural network policy for legged robots. In the aforementioned works, the convergence of the algorithm and the stability of the robot system cannot be guaranteed

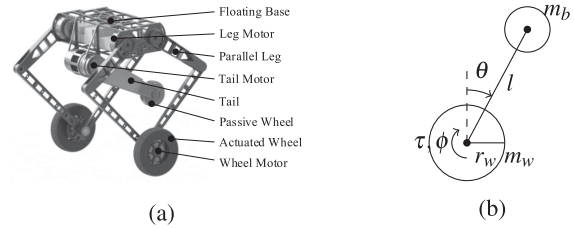


Fig. 2. (a) Mechanical structure of the wheel-legged robot. (b) Template model of the robot.

theoretically. Poor data efficiency of the algorithm is another limitation.

### B. Contributions of Our Work

The primary contributions of this letter are summarised as follows.

- 1) Based on RL and ADP techniques, a data-driven VI algorithm is proposed to generate a near-optimal balance controller in the absence of the accurate dynamics of the wheel-legged robot. To the best of our knowledge, this is the first application of the learning-based control method for the balance of wheel-legged robots with experimental validations.
- 2) The convergence of the data-driven VI algorithm and the stability of the robot in closed-loop with the adaptive optimal controller are theoretically analyzed.
- 3) Under different circumstances, physical experiments are conducted to validate the effectiveness and robustness of the learned optimal controller.

**Notations.** In this letter,  $\mathbb{R}$  denotes the set of real numbers.  $\|\cdot\|$  denotes the Euclidean norm for a vector.  $\otimes$  denotes the Kronecker product. A bold letter represents a vector or a matrix. An italic letter represents a scalar. For a matrix  $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$ ,  $\text{vec}(\mathbf{A}) = [\mathbf{a}_1^T, \dots, \mathbf{a}_n^T]^T$ . For a symmetric matrix  $\mathbf{W} = [w_{i,j}]$ ,  $\text{vecs}(\mathbf{W}) = [w_{1,1}, 2w_{1,2}, \dots, 2w_{1,n}, w_{2,2}, 2w_{2,3}, \dots, 2w_{n-1,n}, w_{n,n}]^T$ . For a vector  $\mathbf{x} \in \mathbb{R}^n$ ,  $\text{vecv}(\mathbf{x}) = [x_1^2, x_1x_2, \dots, x_1x_n, x_2^2, \dots, x_n^2]^T$ .

## II. DYNAMIC MODELLING AND PROBLEM FORMULATION

### A. Dynamic Modelling

As shown in Fig. 2(a), the wheel-legged robot is composed of a floating-based body, a pair of parallel-type legs with active wheels attached to it and a tail. There are six actuators in total, including four leg motors and two wheel motors. For the simplification of the VI algorithm verification, the tail is locked in a fixed position in the experiments. When the four leg motors are actuated, the joint angles are regulated, resulting in the change of the attitude and height of the robot.

The aim of this letter is to balance the robot to the upward position via regulating the torque of the wheels. The weight of the wheel-legged robot mainly concentrates on the floating-based body, which is composed of four motors, batteries, a on-board micro-computer, and circuit boards, and on the active wheels with motors attached to them. Compared with the

weight of the floating-based body and the wheels, the weight of the parallel-type legs is negligible, and therefore, as shown in Fig. 2(b), we can consider the wheel inverted pendulum (WIP) as a template model for the wheel-legged robot. According to [22], the dynamic model is

$$\begin{aligned} m_{11}\ddot{\phi} + m_{12}\cos(\theta)\ddot{\theta} &= m_{12}\sin(\theta)\dot{\theta}^2 + \tau \\ m_{12}\cos(\theta)\ddot{\phi} + m_{22}\ddot{\theta} &= G\sin(\theta) - \tau \end{aligned} \quad (1)$$

where  $m_{11} = (m_b + m_w)r_w^2 + I_w$ ,  $m_{12} = m_b l r_w$ ,  $m_{22} = m_b l^2 + I_b$ ,  $G = m_b g l$ . In detail,  $\phi$  and  $\dot{\phi}$  denote the rotational angle and angular velocity of the wheel,  $\theta$  denotes the tilt angle (pitch angle) of the robot, and  $\tau$  denotes the torque applied on the wheel.  $m_b$  and  $m_w$  denote the masses of the float-based body and the wheel, respectively.  $I_b$  and  $I_w$  denote the momentum of inertia of the float-based body and the wheel, respectively.  $r_w$  is the radius of the wheel, and  $l$  is the height of the body.  $\dot{x} = \dot{\phi} r_w$  denotes the linear velocity of the robot. Defining  $\mathbf{x} = [\theta, \dot{\theta}, \dot{x} - \dot{x}_d]^T$ , and linearizing (1) at  $\theta = 0$  yields

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\tau \quad (2)$$

where  $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{Gm_{11}}{m_{11}m_{22}-m_{12}^2} & 0 & 0 \\ -\frac{r_w Gm_{12}}{m_{11}m_{22}-m_{12}^2} & 0 & 0 \end{bmatrix}$  and  $\mathbf{B} = \begin{bmatrix} 0 \\ -\frac{m_{12}+m_{11}}{m_{11}m_{22}-m_{12}^2} \\ \frac{(m_{22}+m_{12})r_w}{m_{11}m_{22}-m_{12}^2} \end{bmatrix}$ .

*Remark 1:* In some papers, the cart inverted pendulum (CIP) is considered as the template model for the wheel-legged robot. For CIP, the state vector is  $\mathbf{x}_c = [\theta, \dot{\theta}, \dot{x} - \dot{x}_d]^T$ , and the control input  $u$  is the propulsive force applied to the cart. The linearized system of CIP is  $\dot{\mathbf{x}}_c = \mathbf{A}_c \mathbf{x}_c + \mathbf{B}_c u$ . Although the models of CIP and WIP are different, both can be described as a linear system. Therefore, our proposed VI algorithm can be applicable to both WIP and CIP. For WIP, the data  $\mathbf{x}$  and  $\tau$  should be collected, while for CIP, the data  $\mathbf{x}_c$  and  $u$  should be collected.

### B. Problem Formulation

In order to balance the wheel-legged robot optimally, we study the following infinite horizon LQR problem

$$\begin{aligned} \min_{\tau} J &= \int_0^\infty \mathbf{x}^T(t) \mathbf{Q} \mathbf{x}(t) + r \tau^2(t) dt \\ \text{s.t. } \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u \end{aligned} \quad (3)$$

where  $\mathbf{Q}$  is real symmetric and positive semi-definite,  $(\mathbf{A}, \sqrt{\mathbf{Q}})$  is observable, and  $r > 0$ . Conventionally, (3) can be solved by solving the following algebraic Riccati equation (ARE),

$$\begin{aligned} \mathbf{A}^T \mathbf{P}^* + \mathbf{P}^* \mathbf{A} - \frac{1}{r} \mathbf{P}^* \mathbf{B} \mathbf{B}^T \mathbf{P}^* + \mathbf{Q} &= 0 \\ u^*(t) &= -\frac{1}{r} \mathbf{B}^T \mathbf{P}^* \mathbf{x}(t). \end{aligned} \quad (4)$$

However, in (4), the accurate dynamics of system (2) is required. Due to the difficulty in deriving the accurate dynamic model of

the wheel-legged robot in practical application, in this letter, the following problem is formulated.

*Problem* In the absence of the accurate dynamics of the wheel-legged robot, design a model-free VI algorithm such that the LQR problem (3) can be solved by an adaptive optimal controller learned from the data collected along the trajectories of the system (2).

## III. ADP BASED VALUE ITERATION

### A. Preliminaries for VI

*Lemma 1* ([23]): If  $(\mathbf{A}, \mathbf{B})$  is stabilizable and  $(\mathbf{A}, \sqrt{\mathbf{Q}})$  is observable, then  $\lim_{s \rightarrow -\infty} \mathbf{P}(s) = \mathbf{P}^*$ , for any  $S \geq 0$ , where  $\mathbf{P}(s) = \mathbf{P}^T(s)$  is the positive definite solution to the following differential Riccati equation

$$\begin{aligned} -\frac{d\mathbf{P}(s)}{ds} &= \mathbf{A}^T \mathbf{P}(s) + \mathbf{P}(s) \mathbf{A} - \frac{1}{r} \mathbf{P}(s) \mathbf{B} \mathbf{B}^T \mathbf{P}(s) + \mathbf{Q} \\ \mathbf{P}(s_f) &= \mathbf{S}. \end{aligned} \quad (5)$$

In (5),  $s \in (-\infty, s_f]$  denotes the algorithmic time, which can be different from the system evolution time of (2). If the accurate dynamics of the robot is known,  $\mathbf{P}^*$  can be obtained by solving (5) backward in the algorithmic time  $s$ . However, in practice it is hard to obtain such an accurate dynamic model. Therefore, we will show how to solve (5) via the data-driven VI algorithm.

### B. Data-Driven VI

As we know,  $\mathbf{x}(t)$  is the solution to the dynamics (2), which can be measured by on-board IMU and the motor encoders.  $u$  is the driving inputs of the robot. Then according to (2), the following equation holds

$$\begin{aligned} \frac{d}{dt} [\mathbf{x}^T(t) \mathbf{P}(s) \mathbf{x}(t)] &= [\mathbf{A}\mathbf{x}(t) + \mathbf{B}\tau(t)]^T \mathbf{P}(s) \mathbf{x}(t) \\ &+ \mathbf{x}(t)^T \mathbf{P}(s) [\mathbf{A}\mathbf{x}(t) + \mathbf{B}\tau(t)] \\ &= \mathbf{x}^T(t) \mathbf{H}(s) \mathbf{x}(t) + 2r\tau(t) \mathbf{K}(s) \mathbf{x}(t) \end{aligned} \quad (6)$$

where  $\mathbf{H}(s) = \mathbf{A}^T \mathbf{P}(s) + \mathbf{P}(s) \mathbf{A}$  and  $\mathbf{K}(s) = \frac{1}{r} \mathbf{B}^T \mathbf{P}(s)$ . Integrating (6) from  $t_i$  to  $t_{i+1}$  with respect to  $t$  yields

$$\begin{aligned} \mathbf{x}^T(t) \mathbf{P}(s) \mathbf{x}(t) \Big|_{t=t_i}^{t_{i+1}} &= \int_{t_i}^{t_{i+1}} \text{vecv}^T(\mathbf{x}(\nu)) d\nu \text{vecs}(\mathbf{H}(s)) \\ &+ 2r \int_{t_i}^{t_{i+1}} \tau(\nu) \mathbf{x}^T(\nu) d\nu \text{vec}(\mathbf{K}(s)). \end{aligned} \quad (7)$$

Define

$$\begin{aligned} \mathbf{D}_{xx} &= \left[ \text{vecv}(\mathbf{x}(t)) \Big|_{t_0}^{t_1}, \text{vecv}(\mathbf{x}(t)) \Big|_{t_1}^{t_2}, \dots, \text{vecv}(\mathbf{x}(t)) \Big|_{t_l}^{t_{l+1}} \right]^T \\ \mathbf{I}_{xx} &= \left[ \int_{t_0}^{t_1} \text{vecv}(\mathbf{x}(\nu)) d\nu, \int_{t_1}^{t_2} \text{vecv}(\mathbf{x}(\nu)) d\nu, \dots, \right. \\ &\quad \left. \int_{t_l}^{t_{l+1}} \text{vecv}(\mathbf{x}(\nu)) d\nu \right]^T \end{aligned}$$

$$\mathbf{I}_{x\tau} = \left[ \int_{t_0}^{t_1} \tau(\nu) \mathbf{x}(\nu) d\nu, \int_{t_1}^{t_2} \tau(\nu) \mathbf{x}(\nu) d\nu, \dots, \int_{t_l}^{t_{l+1}} \tau(\nu) \mathbf{x}(\nu) d\nu \right]^T. \quad (8)$$

In (8),  $t_0 < t_1 < \dots < t_l < \dots < t_{l+1}$ .  $[t_i, t_{i+1}]$  is the data collecting time interval.  $\mathbf{D}_{xx}$ ,  $\mathbf{I}_{xx}$ , and  $\mathbf{I}_{x\tau}$  can be constructed by the collected data from the system. Writing (7) from the interval  $[t_1, t_2]$  to  $[t_l, t_{l+1}]$  into a compact form yields the following equation

$$\mathbf{M} \begin{bmatrix} \text{vecs}(\mathbf{H}(s)) \\ \text{vec}(\mathbf{K}(s)) \end{bmatrix} = \mathbf{D}_{xx} \text{vecs}(\mathbf{P}(s)) \quad (9)$$

where  $\mathbf{M} = \begin{bmatrix} \mathbf{I}_{xx} & 2r\mathbf{I}_{x\tau} \end{bmatrix}$ .

*Assumption 1:*  $\mathbf{M}$  has full column rank, i.e.  $\text{rank}(\mathbf{M}) = \frac{n(n+1)}{2} + mn$ , where  $m = 1, n = 3$ .

*Remark 2:* The rank condition in the above assumption is a terminology from adaptive control [24], which is named persistent excitation (PE) condition. In order to satisfy it, one can add exploration noise, such as sinusoidal signals and random noise, to the driving input  $u$ . We will explain it in detail in Section IV on the experimental case study.

Then according to the least squares method and Assumption 1, (9) can be rewritten as

$$\begin{bmatrix} \text{vecs}(\mathbf{H}(s)) \\ \text{vec}(\mathbf{K}(s)) \end{bmatrix} = \mathbf{M}^\dagger \mathbf{D}_{xx} \text{vecs}(\mathbf{P}(s)) \quad (10)$$

where  $\mathbf{M}^\dagger = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T$ . Differentiating (10) with respect to  $s$  on both sides and substituting (5), we have the following differential equation

$$\begin{bmatrix} \frac{d\text{vecs}(\mathbf{H}(s))}{ds} \\ \frac{d\text{vec}(\mathbf{K}(s))}{ds} \end{bmatrix} = \mathbf{M}^\dagger \mathbf{D}_{xx} \text{vecs}(-\mathbf{H}(s) + r\mathbf{K}^T(s)\mathbf{K}(s) - \mathbf{Q}). \quad (11)$$

*Theorem 1:* If the Assumption 1 is satisfied,  $\mathbf{P}(s_f) = 0$ ,  $\mathbf{H}(s_f) = 0$ ,  $\mathbf{K}(s_f) = 0$ , and  $\mathbf{H}(s)$ ,  $\mathbf{K}(s)$  are the solution to (11), then the following equations hold:

$$\begin{aligned} \lim_{s \rightarrow -\infty} \mathbf{H}(s) &= \mathbf{A}^T \mathbf{P}^* + \mathbf{P}^* \mathbf{A} \\ \lim_{s \rightarrow -\infty} \mathbf{K}(s) &= \frac{1}{r} \mathbf{B}^T \mathbf{P}^* \end{aligned} \quad (12)$$

where  $\mathbf{P}^*$  is the solution to the ARE (4).

*Proof:* Define

$$\begin{aligned} \mathbf{y} &= \begin{bmatrix} \text{vecs}(\mathbf{H}(s)) \\ \text{vec}(\mathbf{K}(s)) \end{bmatrix} \\ \mathbf{f}(\mathbf{y}) &= \mathbf{M}^\dagger \mathbf{D}_{xx} \text{vecs}(-\mathbf{H}(s) + r\mathbf{K}^T(s)\mathbf{K}(s) - \mathbf{Q}). \end{aligned} \quad (13)$$

In (13), when the data is given,  $\mathbf{M}^\dagger$  and  $\mathbf{D}_{xx}$  are constant.  $\text{vecs}(\mathbf{H})$  is linear with respect to  $\mathbf{y}$  and  $\text{vecs}(\mathbf{K}^T \mathbf{K})$  consists

of the quadratic terms of  $\mathbf{y}$ . Therefore,  $\mathbf{f}(\mathbf{y})$  is locally Lipschitz with respect to  $\mathbf{y}$  at  $\mathbf{y} = 0$ . Hence, according to [25, Theorem 3.1], there exists some  $\delta > 0$  such that  $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$  has an unique solution on the interval  $[s_f - \delta, s_f]$  with  $\mathbf{y}(s_f) = 0$ . From (6) to (11), we know that  $\mathbf{H}(s) = \mathbf{A}^T \mathbf{P}(s) + \mathbf{P}(s) \mathbf{A}$  and  $\mathbf{K}(s) = \frac{1}{r} \mathbf{B}^T \mathbf{P}(s)$  satisfy (11). Therefore, they are the unique solution to (11). According to Lemma 1, the theorem can be proved. ■

Therefore, instead of solving the differential Riccati equation (5), one can calculate the optimal solution of the algebraic Riccati equation (4) by solving (11) backwards. For the construction of (11), the accurate dynamic model is not a requirement, and  $\mathbf{M}$ ,  $\mathbf{D}_{xx}$  can be obtained by the input-state data collected along the trajectories of the robot. (11) can be solved backward by Euler method. Based on the guaranteed convergence of (11), a data-driven ADP based VI algorithm is proposed and detailed in Algorithm 1.

Because the linearized model (2) is only valid within the neighborhood of the equilibrium, the data must be collected within this region. Theoretically, the proposed data-driven VI algorithm does not need any stable initial controller to start the learning process. In the real-world application, in order to guarantee the safety of the robot during the data collection phase, i.e. the robot does not fall to the ground, an initial controller  $\tau_0$  is necessary to balance the robot around the equilibrium. The initial controller can be learned by the proposed VI algorithm in the Gazebo robotic simulator [26], where safety of the robot can be ignored during the data collection phase. If the approximate range of the physical parameters of the robot can be obtained, we can also design the initial PID controller via robust control technique. In order to satisfy the PE condition, some noise  $\beta(t)$  can be added to  $\tau_0$  to excite the robot system.

---

#### Algorithm 1: VI Algorithm.

---

- 1: Choose an initial controller  $\tau_0(\mathbf{x})$  and a noise  $\beta(t)$ , and let  $\tau(t) = \tau_0(\mathbf{x}) + \beta(t)$ .
  - 2: Apply  $\tau(t)$  to the robot and collect the data  $\mathbf{x}(t)$  and  $\tau(t)$  from  $t_0$  to  $t_{l+1}$ .
  - 3: Construct  $\mathbf{D}_{xx}$ ,  $\mathbf{I}_{xx}$ ,  $\mathbf{I}_{x\tau}$  according to (8).
  - 4: Set  $\mathbf{y}(s_f) = \mathbf{0}$ ,  $k = 0$ , step size  $h$  and the small threshold  $\epsilon$ .
  - 5: **repeat**
  - 6:  $k \leftarrow k + 1$
  - 7:  $\mathbf{y}(s_f - kh) = \mathbf{y}(s_f - (k-1)h) - h\mathbf{f}(\mathbf{y}(s_f - (k-1)h))$
  - 8: **until**  $|\mathbf{y}(s_f - kh) - \mathbf{y}(s_f - (k-1)h)| < \epsilon$
  - 9: Reconstruct  $\mathbf{K}(s_f - kh)$  from  $\mathbf{y}(s_f - kh)$  according to (13).
  - 10: Use  $\tau = -\mathbf{K}(s_f - kh)\mathbf{x}$  as the approximated optimal balance controller.
- 

*Remark 3:* The data-driven ADP based VI algorithm is developed for the linearized system but the input-state data is collected from the nonlinear wheel-legged robot, which may induce some disturbances for the VI algorithm. When the data is sampled around the equilibrium, the disturbance is small, and as proved in [27], [28],  $\mathbf{K}(s_f - kh)$  will lie in a ball centered by  $\mathbf{K}^*$ .



The radius of the ball decreases as the norm of the disturbance decreases.

*Remark 4:* Due to the short computational time, the proposed VI algorithm can be implemented both online and offline. For online learning, when robot is running and new data comes in, the new matrices  $\mathbf{D}_{xx}$ ,  $\mathbf{I}_{xx}$ ,  $\mathbf{I}_{x\tau}$  can be constructed. The value iteration loop can be re-executed and the controller is updated.

#### IV. EXPERIMENTS

##### A. Experimental Setup

For the measurement, the pitch angle  $\theta$  and its angular velocity  $\dot{\theta}$  are measured by the on-board IMU. The rotational angular velocities of the left and right wheels,  $\dot{\phi}_l$  and  $\dot{\phi}_r$ , are measured by the motor encoders. The sampling frequency is 100 Hz. CPU PICO-WHU4 is used to calculate the control law, the control frequency of which is 1000 Hz.

The linear velocity of the robot  $\dot{x}$  is calculated by  $\dot{x} = \frac{\dot{\phi}_l + \dot{\phi}_r}{2} r_w$ , and the yaw angular velocity of the robot  $\dot{\psi}$  is calculated by  $\dot{\psi} = \frac{\dot{\phi}_r - \dot{\phi}_l}{w_d} r_w$ , where  $r_w = 0.1$  m is the wheel radius and  $w_d = 0.47$  m is the width of the robot.

The ADP controller generated by Algorithm 1 is used to control the forward and backward movement of the robot, where the reference linear velocity  $\dot{x}_d$  is given by the remote controller. In order to steer the yaw of the robot, we adopt the controller  $\tau_{\psi} = 0.3(\dot{\psi} - \dot{\psi}_d)$ , where  $\dot{\psi}_d$  is the reference yaw angular velocity given by the remote controller. Then the torques for the left and right wheel are calculated by  $\tau_l = \frac{\tau}{2} + \tau_{\psi}$ , and  $\tau_r = \frac{\tau}{2} - \tau_{\psi}$ . As  $\tau_l + \tau_r = \tau$ ,  $\tau_{\psi}$  does not change the total torque applied on the wheels of the robot. Therefore, the yaw movement will not influence the balance of the robot. In this section, the unit of angle is converted to degree for better understanding.

##### B. Training Process

Firstly, the training process is conducted when the robot is at a high height  $l = 0.65$  m. The parameters of Algorithm 1 are  $t_{i+1} - t_i = 0.1$  s,  $h = 0.001$ , and  $\epsilon = 0.001$ . For the cost function in (3),  $\mathbf{Q} = \text{diag}[1400, 70, 70]$ ,  $r = 70$ . In order to satisfy the PE condition in Assumption 1, the exploration noise  $\beta$  is set as  $\beta(t) = 0.5 \sin(\pi t) + 0.3 \sin(2\pi t)$ . In order to guarantee the safety of the robot during the data-collection phase, an initial PID controller with gain matrix  $\mathbf{K} = [-7, -2, -2]$  is fine-tuned for the low height case, which is not fine-tuned for the high height case. The collected data is shown in Fig. 3, where 1210 sets of data in 12.1 s are used for the training process. In this figure, due to the influence of the sensor noise and the added sinusoidal noise, the pitch angle  $\theta$  and the velocity  $\dot{x}$  fluctuates heavily around the equilibrium. Therefore, the PE condition in Assumption 1 can be easily satisfied.

In (8), in order to construct  $\mathbf{I}_{xx}$  and  $\mathbf{I}_{x\tau}$ , the continuous signals for  $x$  and  $u$  are required. Due to the discreteness of the sampled data, here we use the trapezoidal integration to calculate the integration. The step size of the trapezoidal integration is 0.01 s, which is same as the sampling period.

The VI algorithm is executed on the computer with CPU Intel(R) Core(TM) i7-8565 U @ 1.80 GHz and 8 GB RAM.

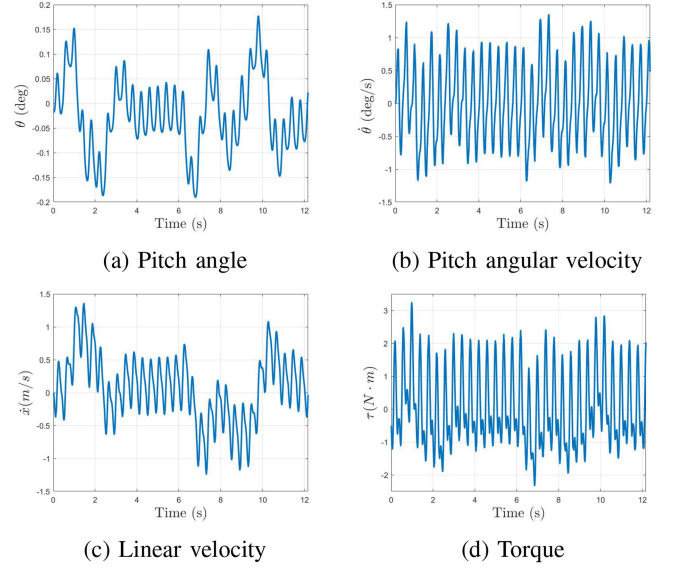


Fig. 3. Collected training data ( $l = 0.65$  m).

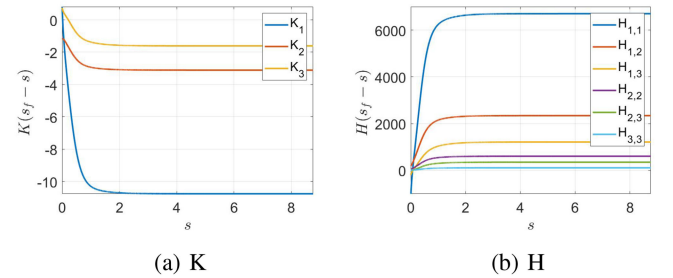


Fig. 4. Evolution of training parameters ( $l = 0.65$  m).

Matlab 2018a is applied as calculation software. For computational time, it takes 0.3156 s until the algorithm converges. The result of the VI algorithm is shown in Fig. 4. In Fig. 4 a, it is shown that when  $s > 1.5$ , the gain matrix  $\mathbf{K}(s_f - s)$  converges to  $[-10.76, -3.11, -1.61]$ . In Fig. 4 (b), the matrix  $\mathbf{H}(s_f - s)$  also converges to

$$\mathbf{H} = \begin{bmatrix} 6701.36 & 2341.44 & 1212.24 \\ 2341.44 & 606.72 & 350.36 \\ 1212.24 & 350.36 & 111.39 \end{bmatrix}. \quad (14)$$

These coincide with the theoretic analysis of the algorithm, and demonstrate the convergence property of the algorithm.

The same training process is conducted when the robot is at a low height  $l = 0.38$  m. For computational time, it takes 0.1269 s until the algorithm converges. Particularly, 766 sets of data are collected in 7.66 s for the training process. For DDPG, 98 000 sets of data (the number of episodes is 245 and the number of steps for each episode is 400) are required for the training of a inverted pendulum [29], the degree of freedom of which is one. Therefore, the proposed VI algorithm is more data-efficient than the conventional RL. The exploration noise  $\beta(t) = 0.5 \sin(\pi t)$ , and the rest keeps the same. The evolution of the matrices  $\mathbf{K}(s_f - s)$  and  $\mathbf{H}(s_f - s)$  are shown in Fig. 5. In

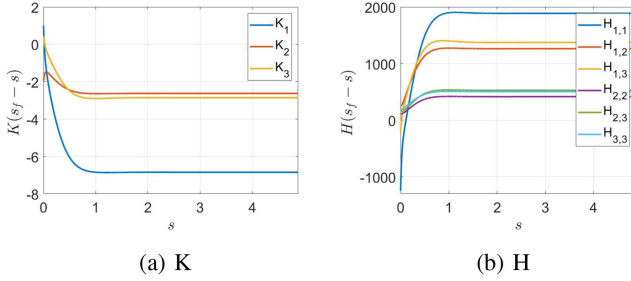


Fig. 5. Evolution of training parameters ( $l = 0.38 \text{ m}$ ).

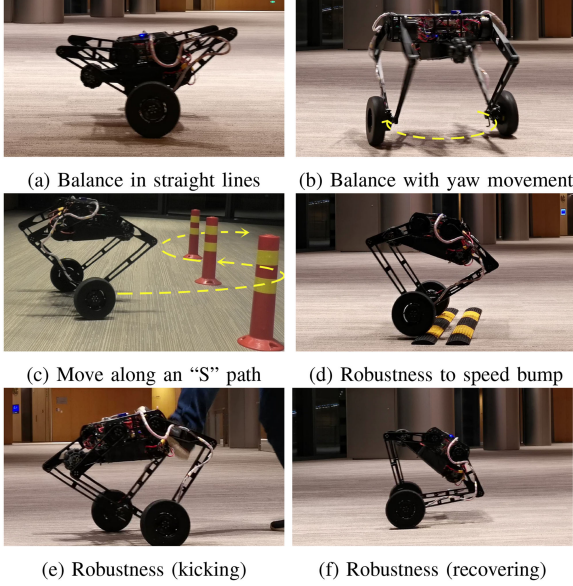


Fig. 6. Photos of experiments in various scenarios. These and many more maneuvers in action are shown in the accompanying video.

result,  $\mathbf{K}(s_f - s)$  converges to  $[-6.85, -2.63, -2.86]$ , which is close to the empirical-tuned controller for the low height case, and  $\mathbf{H}(s_f - s)$  converges to

$$\mathbf{H} = \begin{bmatrix} 1885.95 & 1261.76 & 1372.31 \\ 1261.76 & 414.50 & 526.95 \\ 1372.31 & 526.95 & 503.11 \end{bmatrix}. \quad (15)$$

The gain matrix  $\mathbf{K}$  obtained by the training process will be tested in the next section to show its balance performance.

### C. Test Process

In order to demonstrate the balancing performance of the ADP controller trained by Algorithm 1, we conducted various experiments for the robot with different heights. For the height  $l = 0.38 \text{ m}$ , the learned gain matrix  $\mathbf{K} = [-6.85, -2.63, -2.86]$  is applied to balance the robot. For the height  $l = 0.65 \text{ m}$ , the learned gain matrix  $\mathbf{K} = [-10.76, -3.11, -1.61]$  is applied to balance the robot with the yaw movement and external disturbances.

Photos during experiments are listed in Fig. 6. The detailed experiments are shown in the accompanying video. The experimental results are discussed below separately.

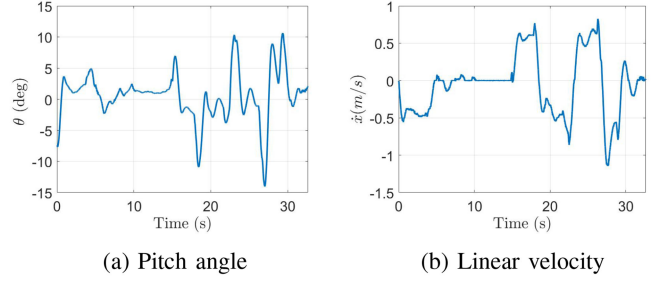


Fig. 7. Experimental data for moving in straight lines.

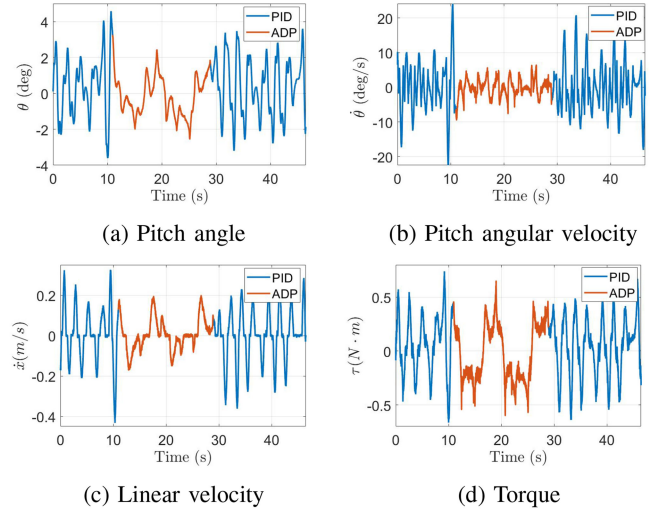


Fig. 8. Experimental data for comparison with different controllers.

1) *Move in straight lines.* Using ADP controller with the gain matrix  $\mathbf{K} = [-6.85, -2.63, -2.86]$ , the robot keeps balance at low height, with  $l = 0.38 \text{ m}$ , as shown in Fig. 6(a). As shown in Fig. 7(b), the robot stands still during  $10 \sim 15 \text{ s}$  and moves forward and backward later with the velocity following the reference signal  $\dot{x}_d$ . Meanwhile, the pitch angle  $\theta$  is vibrating around the equilibrium 0 (Fig. 7(a)).

The results demonstrate the stability of the robot in closed-loop with the ADP controller. Moreover, as the initial PID controller is fine-tuned for the low height case, and the ADP controller is close to the initial controller, they are not compared here.

2) *Comparison with different controllers.* In this experiment, we compare the performance of the initial controller ( $\mathbf{K} = [-7, -2, -2]$ ) used for training with the ADP controller generated by the VI algorithm ( $\mathbf{K} = [-10.76, -3.11, -1.61]$ ). During the experiment, as shown in Fig. 8, the PID controller is used at the beginning and the end, whose results are in blue; the ADP controller is used in the middle phase of the experiment, whose results are in red. From the result, both controllers are able to balance the robot, whose pitch angle is bounded around its equilibrium 0 (Fig. 8(a)). However, from Fig. 8(b) 8(c), it is clear that the amplitude and frequency of the velocities  $\dot{\theta}$ ,  $\dot{x}$  under ADP controller are much smaller than that under the initial controller. Correspondingly, the vibration of the motor torque is also less oscillating under the ADP controller. Hence, the ADP

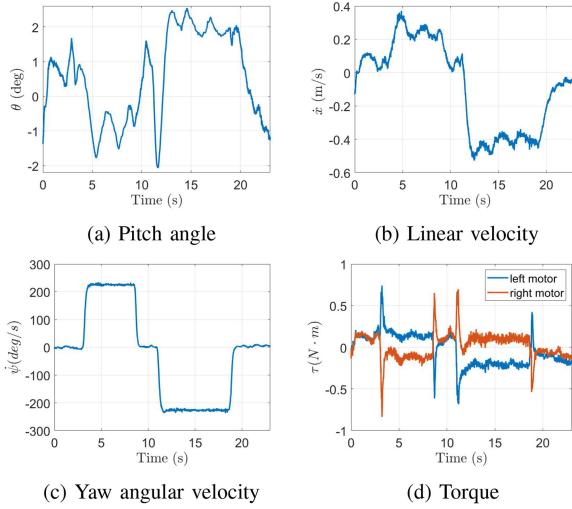


Fig. 9. Experimental data for balancing with yaw movement.

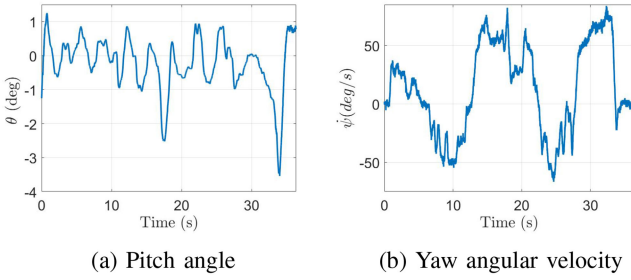


Fig. 10. Experimental data for moving along an "S" path.

controller can balance the robot and improve the performance of the initial PID controller.

3) *Balance the robot with yaw movement.* In the rest experiments, the robot is at the high height, and the gain matrix  $\mathbf{K} = [-10.76, -3.11, -1.61]$  generated by Algorithm 1 is used. The experiment is shown in Fig. 6(b).

As shown in Fig. 9(d), the torques of two wheel motors are almost opposite, so the robot rotates. Ideally, the robot can rotate without any movement in neither longitudinal nor lateral directions. However, in this experiment, when the robot starts rotating, it is not still (see the beginning of Fig. 9(b)). As a result, the robot moves slowly at the beginning. Due to the coupling between  $\dot{x}$  and  $\theta$ , the pitch angle varies in a small range  $[-2^\circ, 2.5^\circ]$  (Fig. 9(a)). Fig. 9(c) shows that the robot rotates at a constant angular velocity in the positive direction initially, and then turns to the opposite direction. Hence, the ADP controller can keep the balance of the robot when it is rotating, which enables the robot to move freely on the ground.

4) *Move along an "S" path.* Experiments 1) and 2) test the performance of the controller when the robot moves along a straight line and rotates separately. In this experiment, the two actions, including the longitudinal movement and yaw movement, are combined as S-turns, as shown in Fig. 6(c).

In result, the pitch angle  $\theta$  often varies slightly between  $\pm 1^\circ$  (Fig. 10(a)). It demonstrates the capability of the ADP controller to balance the robot. In addition, the yaw angular velocity in

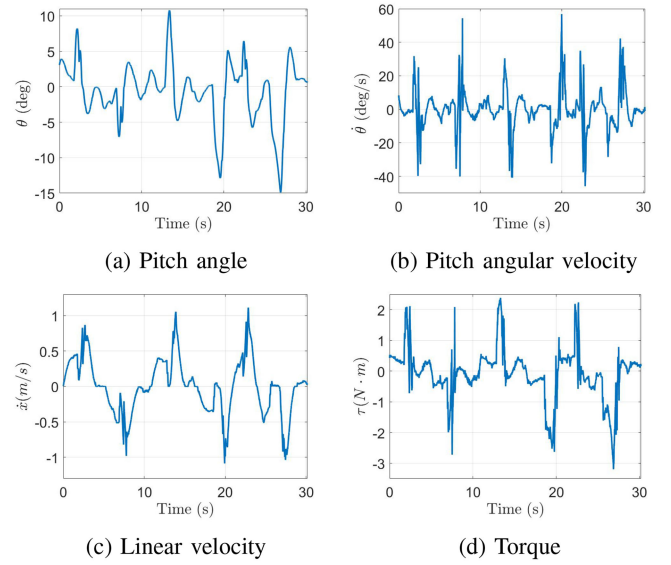


Fig. 11. Experimental data for robustness to the speed bump.

Fig. 9(c) implies that the robot turns left and right during the experiments.

5) *Robustness to the speed bump.* In the rest experiments, external disturbances are added to the robot. In this experiment, the robot goes through a speed bump forward and backward repetitively (Fig. 6(a)). The speed bump can be considered as a disturbance to the bottom of the robot.

As indicated in Fig. 11, the robot goes through the speed bumps six times with three times forward and three times backward. In detail, the robot slows down when it reaches the bump, and sometimes stops when reaching the bump (e.g.  $t = 12.7$  s in Fig. 11(c)). Meanwhile, the robot leans to the bump and the pitch angle increases (Fig. 11(a)). Then, the torque applied on the wheels increases to keep the balance of the robot (Fig. 11(d)), and the robot goes through the bump.

6) *Robustness to the kick.* In this experiment, an external disturbance is added to the top of the robot by kicking the robot body (Fig. 6(e) and 6(f)).

As indicated by the results in Fig. 12, the experimenter kicks the robot four times. Here, the first kick is discussed in detail, which corresponds to the time interval from 1.5 s to 4 s in Fig. 12. As the force of the kick acts towards the negative  $z$ -direction and positive  $x$ -direction (see Fig. 6(e)), the robot moves towards positive  $x$ -direction and the pitch angle becomes negative quickly due to the inertia. This is consistent with the sudden change of  $\theta$  in negative direction in Fig. 12(a) and  $\dot{x}$  in positive direction in Fig. 12(c) around  $t = 1.5$  s. In this situation, as the ADP control slows down the robot ( $t = 2$  s to 2.4 s), the pitch angle becomes more negative ( $\theta = -21^\circ$  at 2.4 s). Then, the dominant negative pitch angle leads to a large negative control law, so the robot moves towards negative  $x$ -direction ( $t = 2.4 \sim 4$  s). Then, similar to the normal case, the ADP controller would regulate  $\dot{x}$ ,  $\theta$  and  $\dot{\theta}$  to zero before the next kick ( $t = 4.7$  s). In this experiment, as the robot is kicked powerfully, the ADP control is more aggressive, and the robustness is sound.



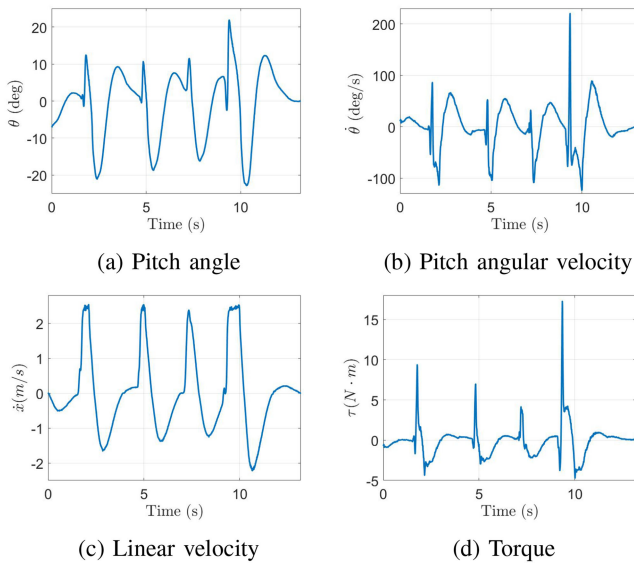


Fig. 12. Experimental data for robustness to the kick.

These training and experimental results demonstrate that:

- The proposed data-driven VI algorithm can generate a controller to balance the wheel-legged robot in the absence of the accurate dynamic model using a small amount of training data.
- The ADP controller can control the robot to track the desired linear velocity and yaw angular velocity given by the remote controller.
- The ADP controller shows sound robustness to external disturbances applied to the bottom and top of the robot.

## V. CONCLUSION

Based on recent RL and ADP techniques, this letter proposes a VI algorithm to generate a data-driven adaptive optimal balance controller that is responsive to parameter variations in the model of a wheel-legged robot. Since a systematic theory for adaptive optimal control of nonlinear systems is lacking, we have focused on a linearized model for the wheel-legged robot. A novel learning-based VI algorithm is developed for the dynamics of the wheel-legged robot evolving in continuous time, state and input spaces. Theoretic studies in terms of convergence proof and stability analysis are carried out. Experimental studies are used to validate the efficacy of the proposed learning-based adaptive optimal controllers. Our future work will be directed at developing advanced learning control algorithms for the nonlinear model of the wheel-legged robot.

## ACKNOWLEDGMENT

The authors would like to acknowledge Ke Chen, Xiangyu Chen, and Jin Liu for their support in the robotics system integration.

## REFERENCES

- [1] B. Dynamics, "Handle<sup>TM</sup>," [Online]. Available: <https://www.youtube.com/watch?v=7xvqQeoA8c>
- [2] V. Klemm *et al.*, "LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3745–3752, Apr. 2020, doi: [10.1109/LRA.2020.2979625](https://doi.org/10.1109/LRA.2020.2979625).
- [3] S. Wang *et al.*, "Balance control of a novel wheel-legged robot: Design and experiments," in *Proc. Int. Conf. Robot. Automat.*, 2021, pp. 6782–6788.
- [4] V. Klemm *et al.*, "Ascento: A two-wheeled jumping robot," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 7515–7521.
- [5] S. Xin and S. Vijayakumar, "Online dynamic motion planning and control for wheeled biped robots," in *Proc. IEEE/RSJ Inter. Conf. Intell. Robots Syst.*, 2020, pp. 3892–3899.
- [6] H. Chen, B. Wang, Z. Hong, C. Shen, P. M. Wensing, and W. Zhang, "Underactuated motion planning and control for jumping with wheeled-bipedal robots," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 747–754, Apr. 2021.
- [7] Z. P. Jiang, T. Bian, and W. Gao, "Learning-based control: A tutorial and some recent results," *Found. Trends Syst. Control*, vol. 8, no. 3, pp. 176–284, 2020.
- [8] Y. Jiang and Z. P. Jiang, *Robust Adaptive Dynamic Programming*. NJ, USA: Wiley-IEEE Press, 2017.
- [9] T. Bian and Z. P. Jiang, "Value iteration and adaptive dynamic programming for data-driven adaptive optimal control design," *Automatica*, vol. 71, pp. 348–360, 2016.
- [10] T. Bian and Z. P. Jiang, "Continuous-time robust dynamic programming," *SIAM J. Control Optim.*, vol. 57, no. 6, pp. 4150–4174, 2019.
- [11] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2018.
- [12] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–14.
- [13] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
- [14] Y. Jiang and Z. P. Jiang, "Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics," *Automatica*, vol. 48, pp. 2699–2704, Oct. 2012.
- [15] G. Zambella *et al.*, "Dynamic whole-body control of unstable wheeled humanoid robots," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3489–3496, Oct. 2019.
- [16] Y. Wen, J. Si, A. Brandt, X. Gao, and H. H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2346–2356, Jun. 2020.
- [17] L. Cui, W. Chen, H. Wang, and J. Wang, "Control of flexible manipulator based on reinforcement learning," in *Proc. Chin. Automat. Congr.*, 2018, pp. 2744–2749.
- [18] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6129–6136, Oct. 2020.
- [19] Y. Zhou, J. Lin, S. Wang, and C. Zhang, "Learning ball-balancing robot through deep reinforcement learning," in *Proc. Int. Conf. Comput., Control Robot.*, 2021, pp. 1–8.
- [20] M. A. Lee *et al.*, "Making sense of vision and touch: Learning multimodal representations for contact-rich tasks," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 582–596, Jun. 2020.
- [21] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, pp. 5872–5884, 2019.
- [22] J. Huang, Z. Guan, T. Matsuno, T. Fukuda, and K. Sekiyama, "Sliding-mode velocity control of mobile-wheeled inverted-pendulum systems," *IEEE Trans. Robot.*, vol. 26, no. 4, pp. 750–758, Aug. 2010.
- [23] V. Kučera, "A review of the matrix Riccati equation," *Kybernetika*, vol. 09, no. 1, pp. 42–61, 1973.
- [24] K. J. Åström and B. Wittenmark, *Adaptive Control, 2nd Edition*. MA, USA: Addison-Wesley, 1997.
- [25] K. H. Khalil, *Nonlinear Systems, 3rd ed.* Upper Saddle River, NJ: Prentice Hall, 2002.
- [26] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, pp. 2149–2154.
- [27] B. Pang and Z. P. Jiang, "Robust reinforcement learning: A case study in linear quadratic regulation," in *Proc. 35th AAAI Conf. Artif. Intell.*, vol. 35, 2021, pp. 9303–9311.
- [28] B. Pang, T. Bian, and Z. P. Jiang, "Robust policy iteration for continuous-time linear quadratic regulation," *IEEE Trans. Automat. Control*, Jun. 2021, doi: [10.1109/TAC.2021.3085510](https://doi.org/10.1109/TAC.2021.3085510).
- [29] Matlab, "Train DDPG agent to swing up and balance pendulum," [Online]. Available: <https://www.mathworks.com/help/reinforcement-learning/ug/train-ddpg-agent-to-swing-up-and-balance-pendulum.html>