Design, Manufacturing and Testing of a Two-Wheeled Self-Balancing Robot

Lucas Lins Souza[1*], **Matheus Henrique Nunes França**[1]
[1]*SENAI CIMATEC university Center; Salvador, Bahia, Brazil*

This work presents a self-balancing robot's modeling, manufacturing, and testing. This study aimed to design and construct a robot capable of maintaining balance while stationary and in motion, employing an effective control strategy. The robot's design incorporates sensors, microcontrollers, actuators, and control algorithms. The control strategy involved implementing an LQR (Linear Quadratic Regulator) controller and a Kalman filter for state estimation. The results demonstrate the robot's ability to effectively maintain balance and navigate flat terrain while controlled by a joystick. This study provides valuable insights into the design and control of self-balancing robots.

Keywords: Self-Balancing Robot. LQR. Kalman Filter. Dynamic Model.

Mobile robots have garnered significant attention and importance in today's technological landscape. Among the diverse range of mobile and self-balancing robots stand out due to their myriad applications [1,2], particularly in transportation and logistics.

The choice of focusing on self-balancing robots for this project arises from their exceptional mobility on flat terrains and ability to navigate common indoor obstacles like stairs and ramps. However, achieving this mobility requires tackling the challenging task of maintaining balance, given the strongly nonlinear behavior of these robots [3].

This paper presents the balancing and teleoperation performance achieved by the implemented control system of an original two-wheeled self-balancing robot. A model-based approach was adopted for the controller, with system modeling considering the robot's dynamics and the wheel actuators, thus eliminating the need for dedicated joint controllers.

This project is an open-source research platform, providing a resource for students and researchers to delve into robotics. All software was developed using ROS (Robot Operating System) [4], a widely used open-source middleware for robotics to facilitate our study.

Received on 19 October 2023; revised 28 December 2023.
Address for correspondence: Lucas Lins Souza. Rua Embira n° 149, Cond. Platno, ap 2801, T. Titânio, Salvador, Bahia, Brazil. Zipcode: 41680-113. E-mail: lucaslinssouza@gmail.com.

Materials and Methods

Robot Prototype

The design of the robot drew inspiration from the Ascento robot [5] and the work of Kollarčík [6]. These robots feature a substantial base that houses most of the robot's mass and two articulated legs with two joints each, enabling complex maneuvers such as navigating obstacles, climbing stairs, and jumping; however, unlike these robots that use a single servomotor per leg along with an additional mechanism to ensure linear up-and-down movement, a simplified design with two servomotors was adopted for this project, similar to the robot MABEL [7].

The Figures 1 and 2 illustrate the robot's 3D model and genuine model. The upper part of the robot serves as the base, housing its sensors and electronics. An MPU6050 Inertial Measurement Unit (IMU) was utilized in this model, directly connected to a Raspberry Pi 4. The U2D2 board facilitates communication via USB between the Dynamixels and the Raspberry Pi 4. The wheels were equipped with two Dynamixels of the XM430-W210 model, while four MX-106 Dynamixels were used for the joints in the legs. Powering the entire system is a 14.8-volt lithium polymer battery (LiPO).

Figure 1. The robot's designed and manufactured model.

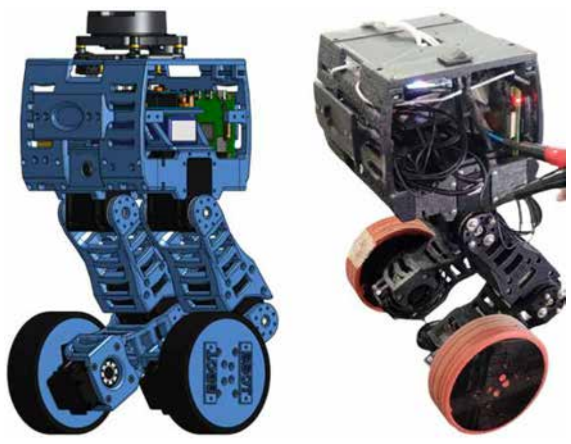
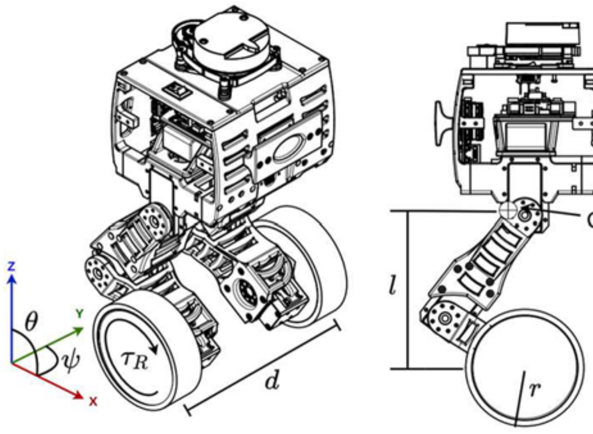


Figure 2. Robot axis and state variables.



The Dynamic Model

A two-wheeled self-balanced robot's design, modeling, and control have been extensively researched and implemented. Klemm and colleagues [5,8] constructed a self-balancing robot that combines the efficiency of wheels with the mobility of legs to navigate uneven terrain and obstacles. Kim and Kwon [9] investigated and presented the dynamic equations of the wheeled inverted pendulum, while Kollarčík [6] designed a wheeled leg system based on these equations. The model utilized in this paper (1) is the one developed in [9] using the Lagrangian method. It is important to note that this model does not account for the legs, so the robot was modeled as having a fixed rod.

$$\ddot{\boldsymbol{q}} = \boldsymbol{F}(\dot{\boldsymbol{q}},\boldsymbol{u}) \tag{1}$$

$$\dot{\boldsymbol{q}} = [\dot{x} \quad \dot{\theta} \quad \dot{\psi}]^{\top} \tag{2}$$

$$\boldsymbol{u} = [\tau_L \quad \tau_R]^{\top} \tag{3}$$

The state vector $\boldsymbol{q}^{\cdot}$ is composed of the linear velocity $\boldsymbol{x}^{\cdot}$, the pitch velocity $\boldsymbol{\theta}^{\cdot}$, and the yaw velocity $\boldsymbol{\psi}^{\cdot}$. Figure 2 illustrates these variables. The input vector $\boldsymbol{u}$ is composed of the left and the right wheel's motor torque $\boldsymbol{\tau_L}$ and $\boldsymbol{\tau_R}$, respectively. The linearization of the system is performed by calculating $\boldsymbol{A_r}$ and $\boldsymbol{B_r}$, which are the Jacobian matrix of the system (1) concerning the state vector $\boldsymbol{q_r}$ (4) and the input vector $\boldsymbol{u}$ (3), respectively, at the system's fixed point (zero for all states). The pitch angle is added in $\boldsymbol{q_r}$ (different from $\boldsymbol{q}^{\cdot}$) because it is also desirable to control the pitch angle of the robot. This model is also in continuous time, so it must be discretized to be implemented in a microcontroller. The discretization was performed by applying the zero-order hold [10] in $\boldsymbol{A_r}$ and $\boldsymbol{B_r}$ with a sampling period of 0.0125 seconds (80 Hz).

$$\boldsymbol{q_r} = [\dot{x} \quad \dot{\theta} \quad \dot{\psi} \quad \theta]^{\top}, \quad \boldsymbol{A_r} = \left.\frac{\partial \ddot{\boldsymbol{q}}}{\partial \boldsymbol{q_r}}\right|_0, \quad \boldsymbol{B_r} = \left.\frac{\partial \ddot{\boldsymbol{q}}}{\partial \boldsymbol{u}}\right|_0 \tag{4}$$

The mathematical model of the robot relies on wheel torques as inputs, while the actuators are controlled using Pulse Width Modulation (PWM). Therefore, it is necessary to establish a relationship between these two variables to send the correct command directly to the motors rather than to a low-level joint controller. Moreover, the accurate actuators exhibit dynamics that the robot model does not consider. By integrating the robot's dynamic model with the motor's dynamic model, we can more accurately represent the system's overall dynamics.

The dynamics of the wheel motors can be represented as a first-order system, with the PWM duty cycle as input and torque as output. It is important to note that the wheels' torque serves as both the input for the robot model and the output for the motor model. The state vector of the final model comprises the states of both the robot and the motor model. However, the output vector only includes the outputs of the robot model because the motor model's outputs are not directly measurable. The final robot model is depicted in Equation (5),

$$\begin{gathered} \boldsymbol{q_F}[\boldsymbol{n}+\mathbf{1}] = \boldsymbol{A_F q_F}[\boldsymbol{n}] + \boldsymbol{B_F u_F}[\boldsymbol{n}], \\ \boldsymbol{y_F}[\boldsymbol{n}] = \boldsymbol{C_F q_F}[\boldsymbol{n}], \\ \boldsymbol{q_F}[\boldsymbol{n}] = \begin{bmatrix} \boldsymbol{q_r}[\boldsymbol{n}] \\ \boldsymbol{q_m}[\boldsymbol{n}] \end{bmatrix}, \quad \boldsymbol{u_F}[\boldsymbol{n}] = \boldsymbol{u_m}[\boldsymbol{n}], \\ \boldsymbol{A_F} = \begin{bmatrix} \boldsymbol{A_r} & \boldsymbol{B_r C_m} \\ \mathbf{0}_{2\times4} & \boldsymbol{A_m} \end{bmatrix}, \quad \boldsymbol{B_F} = \begin{bmatrix} \mathbf{0}_{4\times2} \\ \boldsymbol{B_m} \end{bmatrix}, \\ \boldsymbol{y_F}[\boldsymbol{n}] = \boldsymbol{q_r}[\boldsymbol{n}], \quad \boldsymbol{C_F} = [\boldsymbol{I}_{4\times4} \quad \mathbf{0}_{4\times2}] \end{gathered} \tag{5}$$

where $\mathbf{0}_{ixj}$ is a zero matrix and $\boldsymbol{I}_{ixj}$ is an identity matrix of $\boldsymbol{i}$ rows and $\boldsymbol{j}$ columns and $\boldsymbol{q_m}$, $\boldsymbol{u_m}$, $\boldsymbol{A_m}$, $\boldsymbol{B_m}$, and $\boldsymbol{C_m}$ are the states, inputs, system matrix, input matrix, and output matrix of the motor model, respectively.

Control System Design

The final robot (5) describes the robot's dynamics, which means the system could be stabilized if an LQR controller was designed based on those matrices. To teleoperate the robot, however, the robot must receive and follow linear and yaw speed setpoints. Therefore, the error between the robot's current linear and yaw speed and their respective setpoints must be calculated, and the integrals of those errors must be sent to the controller as additional states. Including this integral action makes it possible to ensure that the errors approach zero; that is, the speed equals the setpoint. Therefore, an augmented model of the system (6) was built, including the integral of the linear and yaw velocity errors as two additional states and the vector $\boldsymbol{r}$ for the linear velocity $\dot{x}_{ref}$ and yaw velocity $\dot{\psi}_{ref}$ setpoints.

To design the gain matrix $\boldsymbol{K_{LQR}}$ of the controller, the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are required. The LQR is an optimal controller that minimizes a quadratic cost function [11], and these two matrices define the priorities in this optimization process. $\boldsymbol{Q}$, in this case, is an 8×8 diagonal matrix, where each value in its diagonal represents a weight that the controller should consider for stabilizing each system state, whereas $\boldsymbol{R}$ is a 2×2 diagonal matrix containing the weights for how much energy the controller can demand from each of the system's inputs. The control law can be defined as

$$\boldsymbol{u_F}[\boldsymbol{n}+\mathbf{1}] = -\boldsymbol{K_{LQR} q_{aug}}[\boldsymbol{n}] \tag{7}$$

As previously mentioned, the $\boldsymbol{y_{aug}}$ does not include the outputs of the motor model because the robot prototype cannot measure the current torque of the wheel motors. Therefore, a Kalman filter was designed to address that the LQR controller needs all current state values to calculate the control effort. The Kalman Filter designed for this work was based on the final robot model, not the augmented model. The design procedure was similar to the LQR controller's [11]: by creating two matrices, namely the disturbance covariance matrix $\boldsymbol{V_d}$ and the noise covariance matrix $\boldsymbol{V_n}$, a gain matrix $\boldsymbol{K_f}$ was calculated (also by optimization), which makes the filter stable. Once stability is reached, the filter outputs converge toward the fundamental values of the final robot model's states. It is essential to highlight that the Kalman filter designed in

$$\begin{gathered} \boldsymbol{q_{aug}}[\boldsymbol{n}+\mathbf{1}] = \boldsymbol{A_{aug} q_{aug}}[\boldsymbol{n}] + \boldsymbol{B_{aug} u_F}[\boldsymbol{n}] + \boldsymbol{r} \\ \boldsymbol{y_{aug}}[\boldsymbol{n}] = \boldsymbol{C_{aug} q_{aug}}[\boldsymbol{n}] \end{gathered}, \qquad \boldsymbol{A_{aug}} = \begin{bmatrix} \boldsymbol{A_F} & \mathbf{0}_{6\times2} \\ \boldsymbol{G} & \boldsymbol{I}_{2\times2} \end{bmatrix},$$

$$\boldsymbol{B_{aug}} = \begin{bmatrix} \boldsymbol{B_F} \\ \mathbf{0}_{2\times2} \end{bmatrix}, \qquad \boldsymbol{G} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}, \tag{6}$$

$$\boldsymbol{q_{aug}} = [\boldsymbol{q_F} \quad \epsilon_{\dot{x}} \quad \epsilon_{\dot{\psi}}]^{\top}, \qquad \boldsymbol{y_{aug}} = [\boldsymbol{q_r} \quad \epsilon_{\dot{x}} \quad \epsilon_{\dot{\psi}}]^{\top},$$

$$\boldsymbol{C_{aug}} = \begin{bmatrix} \boldsymbol{C_F} & \mathbf{0}_{4\times2} \\ \mathbf{0}_{2\times6} & \boldsymbol{I}_{2\times2} \end{bmatrix} \qquad \boldsymbol{r} = [\mathbf{0}_{1\times6} \quad \dot{x}_{ref} \quad \dot{\psi}_{ref}]^{\top}$$

this paper is a linear estimator, which means it converges while the robot operates around the fixed point. Equation (8) shows the calculation of the estimated full state vector of the final robot model $\hat{q}_F$. It can, then, be used to form q_{aug} (6), along with the error as mentioned above integrals. The whole control system is illustrated in Figure 3.

$$\hat{q}_F[n+1] = (A_F - K_f C_F)\hat{q}_F[n] + [B_F \quad K_f][u_F[n] \quad y_F[n]]^T \tag{8}$$

Results

Both stability and teleoperation tests were conducted to evaluate the controller's performance. Throughout these tests, the robot was powered using a tether cable.

Stability Test

This test aimed to assess the system's response to external disturbances. Two tiny pushes were applied to the robot's base while it was balancing on flat terrain: the first from the front and the second from the back (Figures 4 and 5). The push from the back is depicted in Figure 4 (top middle frame), the Figure 5 shows the teleoperation test, and the results are presented in Figures 6 and 7.

It is evident that both pushes, occurring around seconds 6 and 11, led to an increase in the frequency of the system's oscillations, as shown in Figure 6. Eventually, the robot returned to its previous steady state. However, this steady state exhibited significant oscillations. These oscillations indicate that the controller is operating near the stability limit, suggesting that slightly larger disturbances could render the system marginally stable or even unstable.

Teleoperation Test

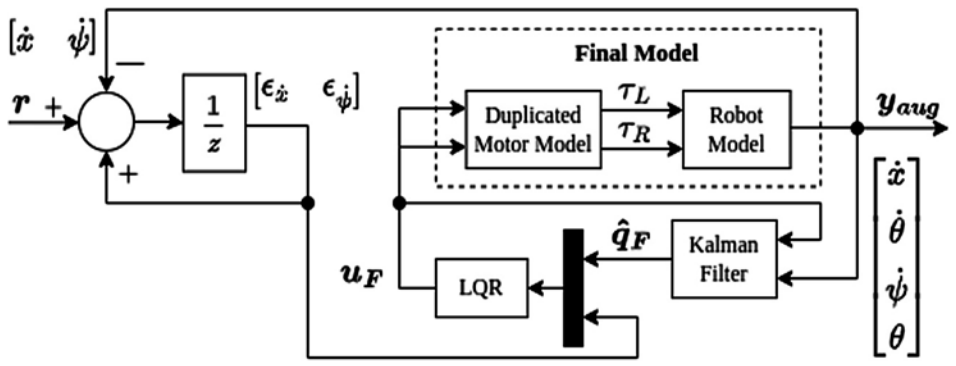
This test aimed to analyze whether the controller could enable the robot to follow linear and yaw velocity setpoints sent by the user via a joystick. Forward and backward motions, as well as turning right and left, were tested individually.The results are depicted in Figures 8 and 9. The robot maintained its balance throughout the test, oscillating below 0.1 radians. Figure 8 illustrates the system's response alongside the input signal, showcasing no delay, overshoot, and a settling time of approximately 3.5 seconds. Figure 9 indicates that the robot moved forward without delay but experienced a delay of around 5 seconds when moving backward.

Additionally, it was unable to reach the setpoint of 0.5 m/s. The controller executed numerous small backward and forward movements to stabilize the robot, resulting in oscillating linear velocity. This behavior also affected the settling time, exceeding 10 seconds, as shown in Figure 9 for the second setpoint.

Conclusion

In this study, we successfully designed, modeled, and implemented a self-balancing robot with an effective control strategy. We demonstrated the

Figure 3. Control block diagram.



$[\dot{x} \quad \dot{\psi}]$
$-$
r +
+
$\frac{1}{z}$
$[\epsilon_{\dot{x}} \quad \epsilon_{\dot{\psi}}]$
Final Model
Duplicated Motor Model
τ_L
τ_R
Robot Model
y_{aug}
$\hat{q}_F$
Kalman Filter
u_F
LQR
$[\dot{x} \; \dot{\theta} \; \dot{\psi} \; \theta]$

Figure 4. Robot during stability test.

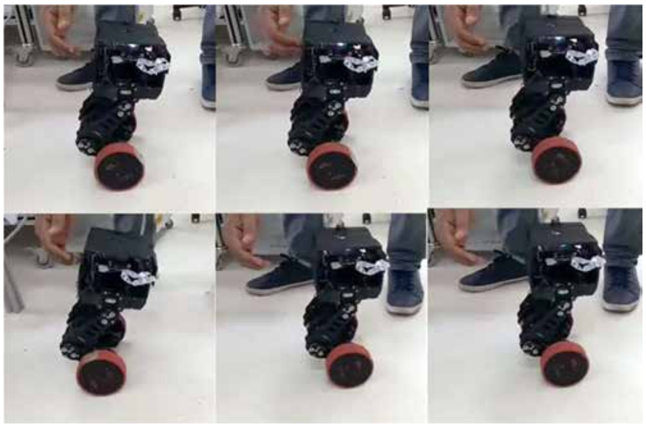


Figure 5. Robot during teleoperation test.

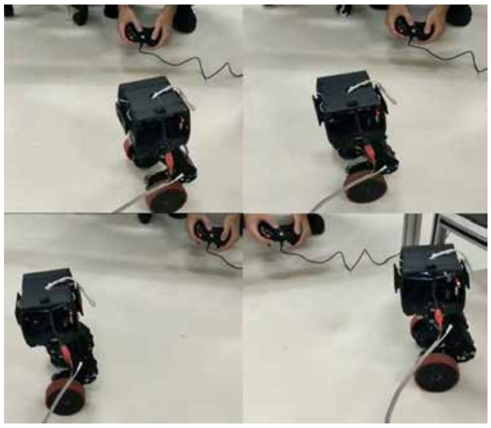


Figure 6. Linear velocity curve.

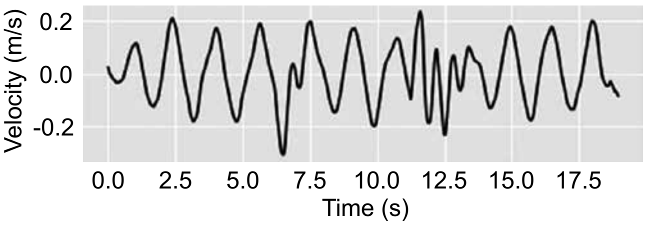
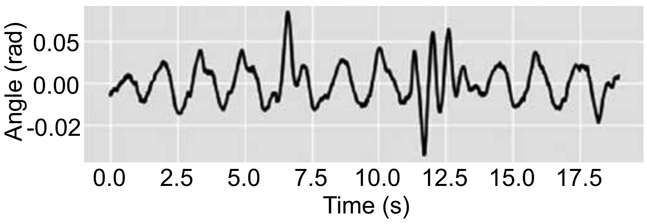


Figure 7. Pitch angle curve.



robot's ability to maintain balance through technical design, mathematical modeling, and real-world testing. The control strategy based on the Linear Quadratic Regulator (LQR) yielded promising results in a natural flat environment. The stability and teleoperation tests provided valuable insights into the controller's performance. The stability test highlighted the robot's resilience in recovering from external disturbances, albeit with noticeable linear and angular velocity curve oscillations.

Similarly, the teleoperation test showed oscillatory behavior, but the robot maintained balance with minimal pitch angle oscillations. However, these oscillations and multiple backward and forward movements led to an extended settling time. Several critical areas of improvement need attention to enhance the self-balancing robot's capabilities and robustness. Firstly, optimizing the robot's weight is crucial for reducing energy consumption and improving maneuverability.

Secondly, optimizing weight distribution to align the center of mass with the wheel-ground contact point can ensure a more stable system. Lastly, upgrading the motor system, particularly by incorporating faster motors for the wheels, can significantly enhance dynamic response and agility. Integrating alternative motor types with higher rotational speeds enables rapid and precise movements, enabling the robot to respond promptly to disturbances and achieve superior performance across various environments.

Figure 8. Yaw angle response.

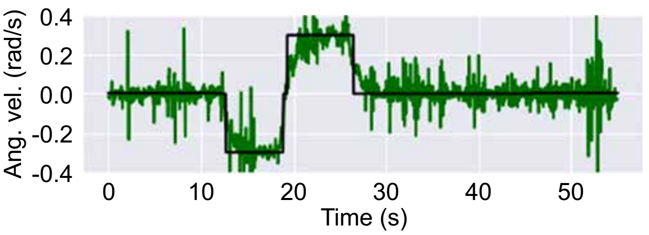
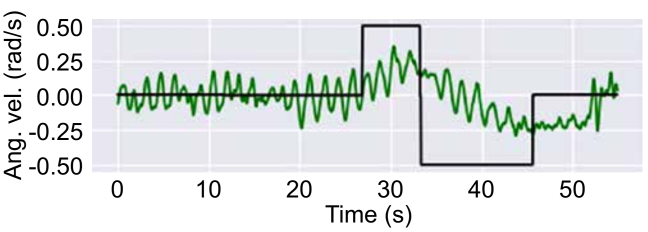


Figure 9. Linear velocity response.



Acknowledgments

We would like to express our sincere gratitude to MS. Marco Reis, Dr. Humberto Monteiro, and the Centro Universitario SENAI CIMATEC for their valuable support and contributions to this project. Their guidance, resources, and facilities have been instrumental in completing this work. We also thank all those who have assisted and supported us throughout this research endeavor.

References

1. DirectDriveTech. Diablo. DirectDrive.Available at: <https://en.directdrive.com product_diablo>.
2. Segway. Segway x2 SE. Available at: <https:/ www.segway.com/segway-x2-se/>.
3. Sun L, Gan J. Researching of two-wheeled self-balancing robot base on LQR combined with PID. 2010 2nd International Workshop on Intelligent Systems and Applications. IEEE 2010:1-5.
4. Stanford Artificial Intelligence Laboratory et al. Robotic operating system. Available at: <https://www.ros.org>.
5. Klemm V et al. LQR-assisted whole-body control of a wheeled bipedal robot with kinematic loops. IEEE Robotics and Automation Letters 2020;5(2):3745-3752.
6. Kollarcik A. Modeling and control of two-legged wheeled robot. 2021. Master thesis, Czech Technical University in Prague.
7. Raspibotics. MABEL (Multi Axis Balancer Electronically Levelled). Raspibotics. Available at: <https://raspibotics.github.io/MABEL/>. Accessed on: 16 Jun. 2023.
8. Klemm V et al. Ascento: A two-wheeled jumping robot. In: 2019 International Conference on Robotics and Automation (ICRA). IEEE 2019:7515-7521.
9. Kim S, Kwon S. Dynamic modeling of a two-wheeled inverted pendulum balancing mobile robot. International Journal of Control, Automation and Systems 2015;13:926-933.
10. Tóth R et al. Crucial aspects of zero-order hold LPV state-space system discretization. IFAC Proceedings Volumes 2008;41(2):4952-4957.
11. Brunton SL, Kutz JN. Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press,2022.