Received 13 July 2025, accepted 28 July 2025, date of publication 11 August 2025, date of current version 15 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3597797

RESEARCH ARTICLE

Modeling and Control of a Ballbot: A Systematic Approach

MAHMOUD ABDELRAHIM[1,2], **MAHMOUD A. THABET**[3], **HOSSAM S. ABBAS**[3,4], **MOHAMED M. M. HASSAN**[3], **MOHAMED H. AMIN**[3], **AND ABDELRAHMAN MORSI**[3]

[1]Renewable Energy Laboratory, Prince Sultan University, Riyadh 12435, Saudi Arabia
[2]Mechatronics Engineering Department, Assiut University, Assiut 71515, Egypt
[3]Electrical Engineering Department, Assiut University, Assiut 71515, Egypt
[4]Institute for Electrical Engineering in Medicine, University of Lübeck, 23558 Lübeck, Germany

Corresponding author: Mahmoud A. Thabet (mahmoud.alyousify@eng.aun.edu.eg)

This article is derived from a research grant funded by the Research, Development, and Innovation Authority (RDIA), Saudi Arabia, with grant number (13354-PSU-2023-PSNU-R-3-1-EI). The authors would also like to acknowledge the technical and support received from Renewable Energy Lab, Prince Sultan University, Saudi Arabia.

ABSTRACT The ballbot is a type of mobile robot that achieves dynamic stability by maintaining balance on a spherical ball. Its control poses significant challenges due to its nonlinear and unstable nature, as well as its five degrees of freedom and underactuated system dynamics with dynamic constraints. To the authors' knowledge, there is a lack of a systematic approach in the literature that addresses these considerations, particularly when it comes to the parameter identification problem. This paper aims to provide a complete and straightforward approach for developing a model-based control of the ballbot, including hardware design and fabrication, detailed system modeling, advanced parameter identification, and control algorithm design. A planar dynamic model is adopted in this work to reduce system complexity and enable real-time control feasibility, serving as a tractable foundation for future extension to a fully coupled 3D dynamic model. The proposed approach is validated through several real-time experiments, demonstrating its robustness and validity. For instance, the identified model achieved a root mean square error (RMSE) of 0.0091 for the yz-plane and 0.0495 for the xy-plane, demonstrating accurate predictive performance, and the controller successfully maintained a tilt angle within $\pm 1^\circ$ under disturbances. The approach demonstrates a modular, low-complexity framework suitable for both research and educational platforms. Future work may explore extending this approach to more complex dynamic environments, advanced hardware, and nonlinear intelligent control strategies such as fuzzy logic and model predictive control.

INDEX TERMS Ballbot, model-based control, LQR, PID, modeling, parameter identification.

I. INTRODUCTION

In recent times, there has been a growing utilization of service robots in various aspects of life to perform interactive tasks with humans. The COVID-19 pandemic has further accelerated the use of service robots for tasks such as contactless delivery operations, disinfection, and medication delivery. According to [1], there exist 203 robot prototypes grouped into six classes that have been used in reality during the COVID-19 pandemic.

Different designs have been presented for service robots. In [2], a service robot was designed to be statically stable by using more than two wheels on a base as shown in Fig. 1a. Such a design typically features a broad base with added weights to lower its center of gravity. Additionally, its acceleration and deceleration capabilities are deliberately kept low to minimize the potential tipping. The low acceleration/deceleration and wide wheelbase restrict its mobility, especially in cluttered human environments.

An alternative design that can solve many fundamental mobility problems of the statically stable mobile robots is the so-called ball balancing robot (ballbot) [3], [4], [5], The ballbot is a type of mobile robot that achieves dynamic stability by maintaining balance on a spherical wheel with a single point of contact on the ground, see Fig. 1b–d. This unique design allows the ballbot to move in any direction

The associate editor coordinating the review of this manuscript and approving it for publication was Hiram Ponce.

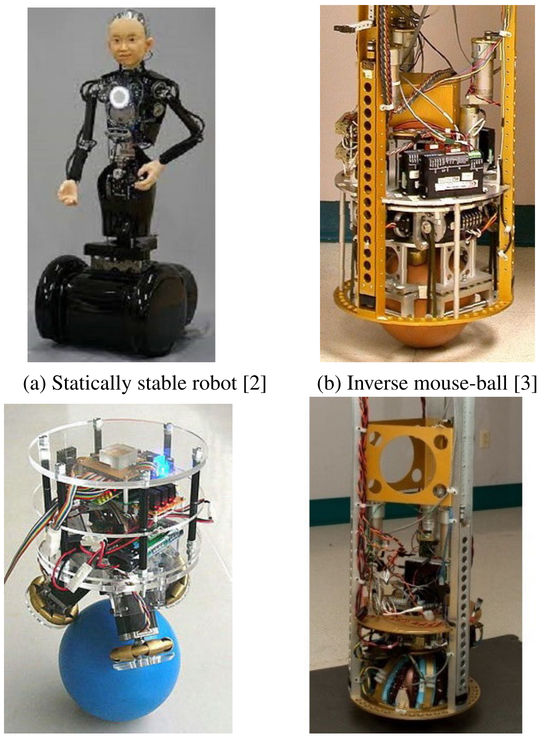

(a) Statically stable robot [2] (b) Inverse mouse-ball [3]
(c) Omni-wheels [4] (d) Spherical induction motor [5]

FIGURE 1. Statically and dynamically stable mobile robots.

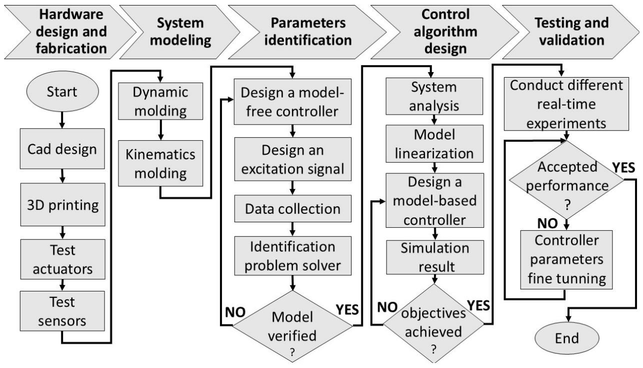


FIGURE 2. Systematic procedure for model-based control of the ballbot.

with a zero turning radius, eliminating the need to roll to change direction. Additionally, its agile movements enhance navigability in confined, crowded, and dynamic environments. Despite its many mobility features, the ballbot is considered a challenging control problem due to its nonlinear and unstable nature, as well as its five degrees of freedom and underactuated system dynamics with dynamic constraints, where a single control input is used to control both the ball position and the body tilt angle for each plane. Although the ballbot has gained significant interest from many researchers, the literature is lacking in a systematic procedure that addresses all of these considerations, particularly concerning the parameter identification issue, which is a crucial step to obtain an accurate model that can be used for designing high-performance controllers.

In our previous conference paper [6], we presented a practical parameter identification procedure for the ballbot. The current extended work builds upon those findings and introduces a significantly expanded framework that offers a comprehensive, systematic approach for model-based control of the ballbot, as summarized in Fig. 2. In addition to parameter identification, this work covers the full development pipeline, including hardware design, system modeling, controller design, and experimental validation. Specifically, the process begins with the design, fabrication, and testing of the ballbot's mechanical and electronic subsystems (ball, actuators, sensors, and control hardware). It then proceeds with the derivation of the kinematic and dynamic models, followed by experimental parameter identification using real-world data. The control design phase includes linearization, controller synthesis, simulation studies, and real-time implementation. Finally, a series of experiments are conducted to validate the system under various conditions and identify practical challenges.

This extended contribution not only expands the scope of the original work but also emphasizes a structured, analytically grounded control methodology that can serve as a foundation for future extensions. These may include the integration of advanced nonlinear or intelligent control strategies, such as fuzzy logic, robust control, or model predictive control. We believe this framework can be generalized to other dynamically stable robotic systems, such as two-wheeled balancing robots [7] and unicycle robots [8]. Furthermore, the developed ballbot prototype serves as a valuable benchmark platform for evaluating a wide range of control techniques and as a teaching aid in robotics and control systems education.

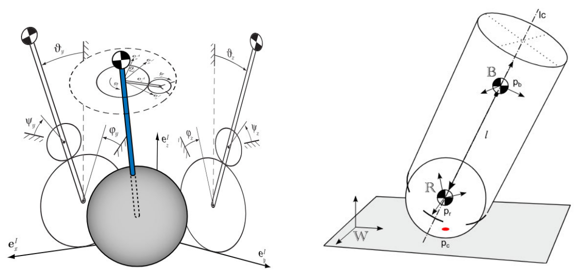
A. LITERATURE REVIEW

In this section, the previous and state-of-the-art works on the ballbot are briefly reviewed. Firstly, an overview of the ball-driving mechanism is discussed, followed by the modeling techniques, and finally, the control approaches.

1) BALL DRIVE MECHANISM

The concept of a ballbot was initially introduced in 2006 [3], where an inverse mouse-ball drive mechanism is used as shown in Fig. 1b. This mechanism consists of four timing belts connecting motors to rollers to balance the robot, while a fifth motor is utilized to provide yaw motion. However, the drawback of such a design is the mechanical complexity and excessive friction between the ball and the rollers.

In 2008, a novel drive mechanism was developed that utilizes three omni-wheels [4], see Fig. 1c. This drive mechanism eliminates the need for a separate mechanism for yaw rotation, while it requires three motors only to drive the ball in all directions. Additionally, it has only three force transmission points compared to the four points on the inverse mouse-ball design. The well-known ballbot robots Kugle [9], Rezero [10], and Pham [11] use the omni-wheel drive mechanism as well with more flexible designs.



(a) Planar 2D model [10] (b) Coupled 3D model [12]

FIGURE 3. **Ballbot modeling sketches.**

Another drive mechanism that was proposed for the ballbot is the spherical induction motor [5], which consists of a hollow steel ball with an outer copper shell and six stators positioned around the circumference of the ball as shown in Fig. 1d. Although this mechanism reduces the number of moving parts in the robot to the ball and the body parts only, it is considered to be electrically too complex.

2) BALLBOT MODELING

The ballbot dynamics have been modeled in various ways, including a simple 2D planar [3], [9], [10], [11], or a more complex coupled 3D model [9], [10], [12], [13]. In the planar model, as shown in Fig. 3a, the ballbot dynamics are separated into three 2D models, one for each plane (xz, yz, and xy), each with two DOFs: one for the ball's translation and one for the body's rotation. Additionally, the three omni-wheels and motors are modeled as three virtual wheels, one for each plane for driving the ball. The coupling between the three 2D models is neglected, meaning the balancing controller consists of three independent controllers, one for each plane. Despite its simplicity, this model provides a suitable approximation to qualitatively understand the dynamics of the ballbot system.

On the other hand, the coupled 3D model, as shown in Fig. 3b, accounts for the coupling between the planes. It can be described by five DOFs, two for the ball's translation and three for the body's rotation. This model accurately captures all aspects of the ballbot dynamics, providing a reliable basis for designing controllers, which can handle fast and complex maneuvers. However, it is more complex and computationally demanding than the 2D planar model.

3) BALLBOT CONTROL

In the literature, several linear control techniques have been proposed for controlling the ballbot system. These include double loop controllers, which typically consist of an outer loop with a linear quadratic regulator (LQR) controller and an inner loop with a PI controller [3], [11]. Other schemes use two PD controllers [4] or a single LQR to simultaneously control the orientation and position of the ball [14], [15]. Additionally, a fuzzy-based intelligent controller [16], [17], [18], [19], [20] and PD controller with inverse dynamics [12] have been proposed.

Moreover, nonlinear control techniques have been used to control the ballbot system. These include sliding mode control [9], [21], [22], gain scheduling control, which consists of a combination of various LQR controllers based on various linearization points [10], a nonlinear model predictive control using iterative linear quadratic Gaussian (ILQG) algorithms [23], and robust nonlinear integral backstepping HSMC (IBHSMC) [24].

Furthermore, with combined control with sliding mode control (SMC) and partial feedback linearization (PFL), the tilt angles of the body are managed through PFL, while the ball's movement on the floor is controlled using the SMC technique [25]. Additionally, a Reinforcement Learning algorithm is superposed to the conventional controller, making up a compound controller that stabilizes the ballbot system with a larger recovery area [26].

Recent trends in mobile robot control have increasingly focused on intelligent and fuzzy logic-based strategies. For instance, a fuzzy PID-based kinematic controller was proposed in [27] for trajectory tracking in non-holonomic mobile robots, demonstrating effective adaptability through gain tuning in virtual environments. In [28], a Type-3 fuzzy logic-based predictive control strategy was introduced to identifying mobile robot dynamics online and handle model uncertainties and control constraints. While these fuzzy logic systems show robust performance, especially under uncertainty, their implementations often rely on extensive simulation environments or require high computational effort. In contrast, our current work prioritizes analytical clarity, modularity, and real-time feasibility on embedded platforms, establishing a reliable foundation upon which such intelligent methods can later be benchmarked and integrated.

B. PAPER CONTRIBUTIONS

The main contribution of this paper is to provide a comprehensive systematic approach for model-based control of the ballbot, the goal being to develop a fully operational prototype. In that regard, the work presented in this paper comprises the following contributions:

- The robot's mechanical design is characterized by mechanical simplicity, easy upgradability, and lower-cost manufacturing.
- The parameter identification procedure using experimental data is a key issue and poses significant challenges in ballbot control.
- We consider the motor speeds as a control input, which can reduce jerky motion and produce smoother motion compared to using the torque as the control input, which is commonly considered in the literature.
- The model of the ballbot does not incorporate motor dynamics; instead, for each motor, a separate low-level controller is utilized to follow the commands of the high-level balance controller. This approach minimizes the overall dynamical order of the closed-loop system, simplifies real-time implementation, and enhances

the modularity and execution speed of the control system.
- The proposed approach and control algorithms are validated through real-time experiments under various situations, testing different control objectives, including balancing, reference tracking, and disturbance rejection.

C. PAPER STRUCTURE

The paper is organized as follows: Section II describes the construction of the ballbot system and the details of the implementation process. Section III introduces the ballbot mathematical model and the two-level controller approach as well as the kinematic modeling of the ballbot. Section IV presents a practical and straightforward procedure based on Simulink and the Optimization Toolbox of MATLAB R2018a for identifying the unknown physical parameters of the ballbot using experimental data. Section V presents the linearization of the nonlinear model and analyzes the system model, as well as describes the design of the model-based controllers. The simulation results using MATLAB/Simulink are also presented. Section VI presents the experimental results from various tests conducted under different conditions to verify the validation of the proposed model and to evaluate the reliability and robustness of the control algorithms. Finally, Section VII provides conclusions and potential future works.

II. HARDWARE DESIGN AND FABRICATION

This section presents a detailed discussion of the ballbot experimental setup, including the drive mechanism and the sensors used to measure the robot states, as well as the digital filters used to mitigate noise in sensor measurements. It also illustrates the hardware wiring diagram of the ballbot system.

A. SETUP OVERVIEW

The experimental setup of the ballbot system considered in this work is illustrated in Fig. 4a. It comprises two parts: the body and the drive mechanism. The body is fabricated using 3D printing and is divided into two levels. The lower level accommodates the drive mechanism, while the upper level houses the electrical components, such as sensors, embedded boards, and a 12 V, 10 Ah Li-Po battery. Three-dimensional printing is used to fabricate the ballbot chassis and omni-wheels, providing benefits such as mechanical simplicity, easy upgradability, and lower-cost manufacturing.

A computer-aided design (CAD) model, shown in Fig. 4b, is an essential step in the installation of the ballbot. After determining the appropriate dimensions and materials of the different ballbot components, they can be arranged using the CAD in the x and y directions, centralized around the origin of the robot. The CAD model can also be used to estimate some important physical parameters, such as the robot's center of mass and moment of inertia, which will be used to initialize the parameter identification process as described in Section IV. The standard triangle language (STL) files for the motors mounting and the omni-wheels are exported from the CAD model and printed using a homemade 3D printer.

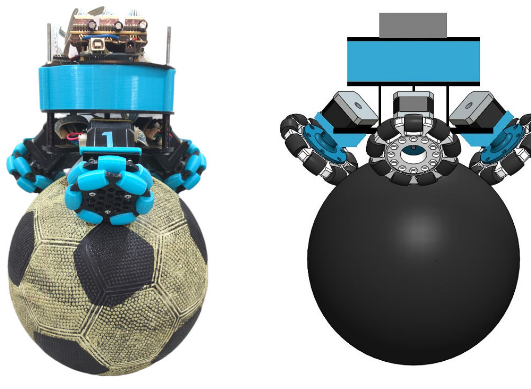
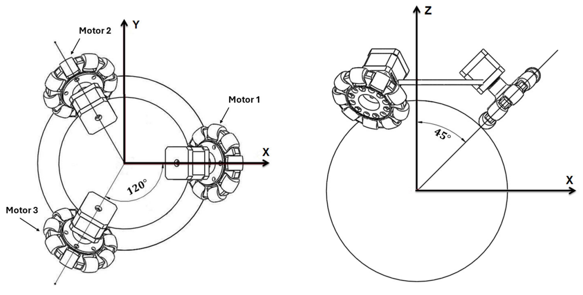

(a) Real ballbot (b) CAD model

FIGURE 4. Ballbot experimental setup.



(a) Top view (b) Side view

FIGURE 5. Ballbot omni-wheels configuration.

B. DRIVE MECHANISM

The ballbot's drive mechanism, as proposed, comprises a spherical ball (size 5 football), three stepper motors, and three omni-wheels. To provide the ballbot system with the required motion, the motors are positioned symmetrically at intervals of $\beta = 120°$, and the first motor is aligned with the x-axis as depicted in Fig. 5a. Additionally, each omni-wheel is situated at an angle of $\alpha = 45°$ as depicted in Fig. 5b.

In this work, the omni-wheels speeds are utilized as control inputs for the ballbot, calculated by a high-level controller as will be discussed in Section III. These speeds are then used as reference commands for low-level controllers (speed controller) to drive the stepper motors. The stepper motors provide precise positioning, repeatability of motion, fast reversals, and high torques. As depicted in Fig. 6, the low-level controller can accurately track a reference speed with a short settling time of 20 ms.

C. SENSORS

The angular velocities and tilt angles of the robot body are measured using an inertial measurement unit (IMU), while the velocity and position of the ball are measured using three quadrature encoders attached to the motors. These sensors provide full-state feedback for the control system, eliminating the need for observers.

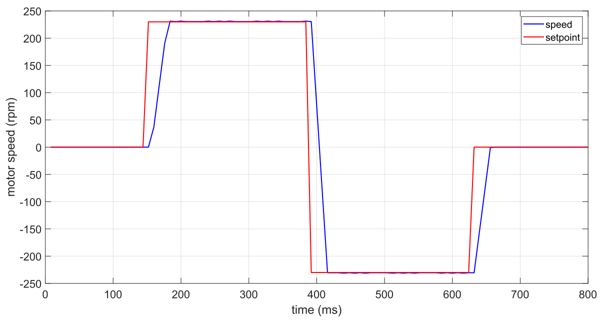


FIGURE 6. Low–level speed controller response.

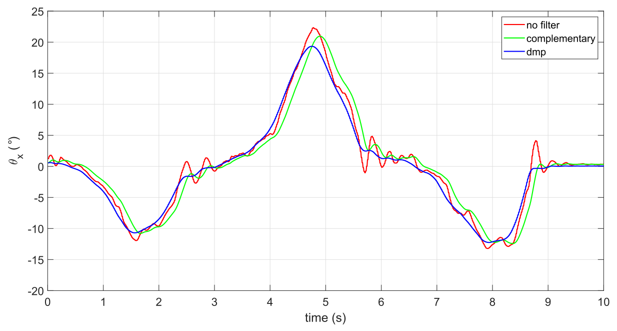


FIGURE 7. Tilt angle measurement.

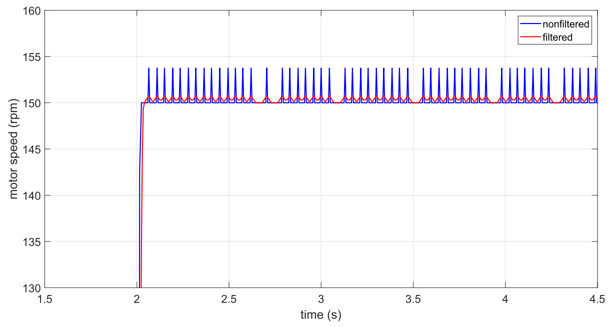


FIGURE 8. Motor speed measurement.

The IMU is the primary sensor in the ballbot, providing the angular velocities and tilt angles of the robot body. The MPU6050 sensor is employed, which combines a 3-axis gyroscope and a 3-axis accelerometer. The gyroscope measures angular velocities around the x-, y-, and z-axis with low-frequency noise, while the accelerometer measures linear accelerations along the x-, y-, and z-axis with high-frequency noise. One way to calculate the tilt angles (θ_x, θ_y, and θ_z) is to use a complementary filter [29] by filtering the accelerometer data with a low-pass filter and the gyroscope data with a high-pass filter, then combining the two signals to obtain a more accurate estimate. The filter is implemented by the following difference equation:

$$\theta_f[n] = \gamma \left(\theta_f[n-1] + T_s\, \dot{\theta}_g[n]\right) + (1-\gamma)\,\theta_a[n],$$
$$\gamma = \frac{N}{T_s + N}. \tag{1}$$

where θ_f is the filtered angle, $\dot{\theta}_g$ is the gyroscope angular velocity, θ_a is the tilt angle obtained from accelerometer data, T_s is a sampling period and N is a time constant. The filter response depends on the tuning of the filter time constant N, which significantly relies on experience.

Another approach to measure the tilt angles is to utilize a digital motion processor (DMP) on the MPU6050, which directly gives filtered data after fusing accelerometer and gyroscope data to reduce errors inherent in each sensor. This can decrease the workload on the peripheral microprocessors and avoid filtering and data fusion. Therefore, the DMP is considered in this work to obtain filtered tilt angles as shown in Fig. 7, which compares the filtered tilt angle obtained from the DMP and the complementary filter to the noisy tilt angle obtained from the accelerometer. The second sensor used in the ballbot is the quadrature encoder, which is attached to the driving motor's shaft to measure its position and velocity. These measurements are used to estimate the ball's velocity and position through kinematics equations as will be illustrated in Section III-B. However, as shown in Fig. 8 the revolutions per minute (RPM) measurement has spikes that may affect the performance of the controller. Therefore, a moving-average filter is applied with a window size of three to obtain a smooth measurement.

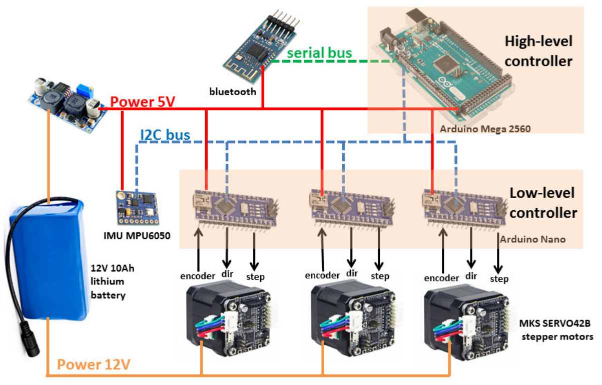
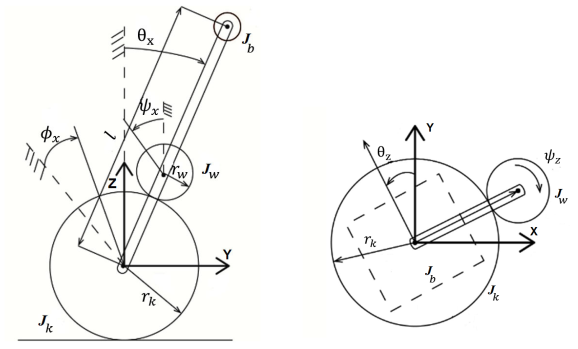


FIGURE 9. Hardware wiring diagram of the ballbot system.

D. PROCESSING AND COMMUNICATION

The hardware wiring diagram of the practical system is illustrated in Fig. 9. The software architecture of the closed-loop system is composed of two-level microcontrollers. The high-level one is an Arduino Mega. Its primary loop involves various function calls, such as reading measurements from sensors, logging data to the PC, executing the algorithm of the high-level controller, and sending commands to the low-level controllers. An Arduino Nano is utilized for each motor to handle the low-level controller responsible for speed control. Communication between the microcontrollers occurs via an I2C bus. Additionally, a Bluetooth joystick is used for sending reference commands and tuning controller parameters online.



(a) $yz-$plane (b) $xy-$plane

FIGURE 10. Planar model sketches for planes *yz* and *xy* (the *xz*-plane is similar to the *yz*-plane).

III. SYSTEM MODELING

The ballbot system is inherently unstable; therefore, high-performance controllers are crucial to stabilize the robot and achieve the desired control objectives. In this work, a model-based control is considered to stabilize the ballbot system as will be illustrated in Section V. Therefore, an accurate model of the system is required. In the sequel, the mathematical model of the ballbot system will be discussed, followed by kinematic modeling, which describes the relationships between the virtual and actual wheels.

A. THE DYNAMIC MODEL OF THE BALLBOT

In this work, we consider the planar modeling [3] to describe the dynamic model of the ballbot system. In such a modeling approach, the system dynamics are separated into three 2D models, one for each plane (xz, yz, and xy) as shown in Fig. 10. To develop a valid dynamic model for the ballbot system, several assumptions have been made, including the following:

1) The equation of motion of the yz- and xz-plane are symmetric and decoupled.
2) The contacts between the ball and the ground and between the wheels and the ball are point contacts.
3) No slipping occurs between the ball and the ground or between the wheels and the ball.
4) The model assumes rigid body dynamics without any deformation.
5) Only viscous friction is considered.
6) A flat floor surface is assumed.

The Euler-Lagrange approach is utilized to derive the dynamic equations of the planar model. A detailed derivation can be found in [9]. The matrix equation of motion in the yz-plane can be expressed as follows:

$$M(q_x)\ddot{q}_x + C(q_x,\dot{q}_x)\dot{q}_x + D(\dot{q}_x) + G(q_x) = Q_x\tau_x \tag{2}$$

where $q_x = \begin{bmatrix} y_k & \theta_x \end{bmatrix}^\top$ with y_k the ball's position in the y-direction, θ_x the body's tilt angle around the x-axis, and τ_x is the virtual wheel torque acting on the ball around the x-axis. Q_x, $D(\dot{q}_x)$, $G(q_x)$, $C(q_x,\dot{q}_x)$, and $M(q_x)$ are the input vector, friction vector, gravity vector, Coriolis matrix, and mass matrix, respectively. These are expressed as follows:

$$Q_x = \begin{bmatrix} \frac{1}{r_w} \\ -\frac{r_k}{r_w} \end{bmatrix},$$

$$D(\dot{q}_x) = \begin{bmatrix} B_{vk}\dot{y}_k \\ 0 \end{bmatrix},$$

$$G(q_x) = \begin{bmatrix} 0 \\ -M_b g l \sin\theta_x \end{bmatrix},$$

$$C(q_x,\dot{q}_x) = \begin{bmatrix} 0 & -M_b l \sin\theta_x\dot{\theta}_x \\ 0 & 0 \end{bmatrix},$$

$$M(q_x) = \begin{bmatrix} \frac{J_k}{r_k^2} + \frac{J_w}{r_w^2} + M_k + M_b & M_b l\cos\theta_x - \frac{J_w r_k}{r_w^2} \\ M_b l\cos\theta_x - \frac{J_w r_k}{r_w^2} & J_b + M_b l^2 + \frac{J_w r_k^2}{r_w^2} \end{bmatrix}.$$

The equation of motion in the xy-plane can be expressed as follows:

$$\ddot{\theta}_z = Q_z\tau_z,$$
$$Q_z = \frac{J_k r_k r_w}{J_k J_z r_w^2 + 3(J_k + J_z)J_w r_k^2 \sin^2\alpha}. \tag{3}$$

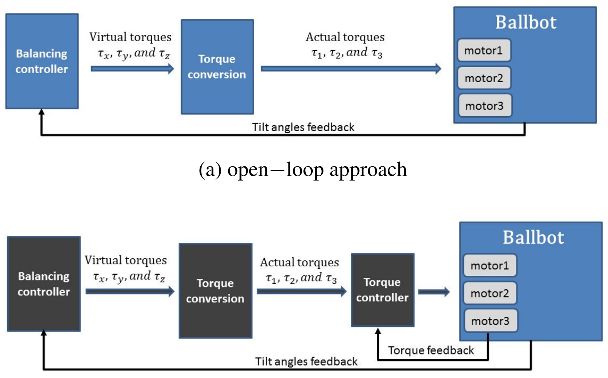
The virtual wheel applies a virtual torque τ_z to the ball around the z-axis, where J_z represents the moment of inertia of the robot body about the z-axis. A description of the ballbot variables and parameters, as depicted in Fig. 10, are presented in Tables 1 and 2, respectively. It should be noted that the dynamics of the xz-plane are the same as those of the yz-plane, with the only difference being that θ_x, y_k, and τ_x are replaced by θ_y, x_k, and τ_y, respectively. In the literature, the motors' torque are commonly used as control inputs. To achieve this, the virtual wheel torque, such as τ_x in (2), is converted to the corresponding torques of the actual wheels. Then, two different approaches can be implemented: in the first approach, shown in Fig. 11a, the actual torques are directly applied onto the motors in an open-loop manner, as demonstrated in [11], which may result in increased jerk and reduced control performance. Alternatively, the second

TABLE 1. Description of ballbot planar model variables.

Variable	Description
j	System coordinates x or y or z axis
i	Motor / omni-wheel 1 or 2 or 3
θ_j	Tilt angle of the robot body around the j-axis
$\dot{\theta}_j$	Angular velocity of the robot body around the j-axis
$\ddot{\theta}_j$	Angular acceleration of the robot body around the j-axis
ϕ_j	Angle of the ball around the j-axis
$\dot{\phi}_j$	Angular velocity of the ball around the j-axis
$\ddot{\phi}_j$	Angular acceleration of the ball around the j-axis
ψ_j	Angle of the virtual wheel around the j-axis
$\dot{\psi}_j$	Angular velocity of the virtual wheel around the j-axis
$\ddot{\psi}_j$	Angular acceleration of the virtual wheel around the j-axis
ψ_i	Angle of the actual wheel (omni-wheel) around the i-axis
$\dot{\psi}_i$	Angular velocity of the actual wheel around the i-axis
$\ddot{\psi}_i$	Angular acceleration of the actual wheel around the i-axis
τ_j	Torque of the virtual wheel around the j-axis
τ_i	Torque of the actual wheel (omni-wheel) around the i-axis
x_k, y_k	Translation of the ball along the x-axis and y-axis

TABLE 2. Description of ballbot planar model parameters.

Symbol	Unit	Description
α	[°]	Angle of the omni-wheel contact point
β	[°]	Motor separation angle
g	[m/s^2]	The acceleration of the gravity
M_k	[kg]	The mass of the ball
J_k	[kg·m^2]	The inertia of the ball
r_k	[m]	The radius of the ball
J_b	[kg·m^2]	The inertia of the body
M_b	[kg]	The mass of the body
r_w	[m]	The radius of the omni-wheel
M_w	[kg]	The mass of the omni-wheel
J_w	[kg·m^2]	The inertia of the omni-wheel
B_{vk}	[N/(m/s)]	The viscous friction of the ball to the ground
l	[m]	Distance from body center of mass to ball center
G_y	[kg·m^2]	Angular acceleration to torque constant in xz-plane
G_x	[kg·m^2]	Angular acceleration to torque constant in yz-plane
G_z	[kg·m^2]	Angular acceleration to torque constant in xy-plane



(a) open−loop approach

(b) closed−loop approach

FIGURE 11. Torque as a control input.

approach, as shown in Fig. 11b, utilizes torque control for each motor; therefore, additional sensors are required to implement the torque control. This can result in a high noise in the measurements, which may considerably influence the efficiency of the closed-loop system. In contrast to the aforementioned approaches, this study considers the virtual wheel's angular acceleration as a control input, which can be approximately related to the torque using the subsequent expression:

$$\tau_i = G_i \ddot{\psi}_i, \quad i = \{x, y, z\} \tag{4}$$

where G_i is a constant to be estimated and $\ddot{\psi}_i$ represents the virtual wheel angular acceleration in the x, y, or z direction. The resulting dynamic equations of the yz-, xz-, and xy-plane are written as follows:

$$M(q_x)\ddot{q}_x + C(q_x, \dot{q}_x)\dot{q}_x + D(\dot{q}_x) + G(q_x) = Q_x G_x \ddot{\psi}_x \tag{5}$$

$$M(q_y)\ddot{q}_y + C(q_y, \dot{q}_y)\dot{q}_y + D(\dot{q}_y) + G(q_y) = Q_y G_y \ddot{\psi}_y \tag{6}$$

$$\ddot{\theta}_z = Q_z G_z \ddot{\psi}_z \tag{7}$$

In this way, a low-level controller is implemented for each motor to track the reference speeds computed by the high-level balance controller. This approach allows setting a speed profile, such as a sigmoid function, which effectively minimizes jerk and results in smoother motion of the ballbot. Moreover, the equations of motion do not incorporate stepper motor dynamics. As a result, this approach minimizes the overall dynamical order of the closed-loop system, simplifies real-time implementation, and enhances the modularity and execution speed of the control system. The low-level controllers were implemented for each motor and validated in practice, as shown in Fig. 6. These controllers successfully ensured that the reference velocities generated by the high-level controller were accurately tracked during the experiments.

While excluding motor dynamics simplifies the system and facilitates embedded implementation, we acknowledge that modeling motor dynamics could improve accuracy in high-speed or high-load scenarios and enhance adaptability to different actuator types. This will be considered in future extensions.

B. THE KINEMATIC MODEL OF THE BALLBOT

This section discusses the method used to measure the velocity and position of the ball using the encoders fixed to the motor's shaft. It also explains how the balancing acceleration commands specified in (4) are transformed into angular velocity references for the low-level controller.

The relation between the angular velocities of the omni-wheels ($\dot{\psi}_1$, $\dot{\psi}_2$, $\dot{\psi}_3$) and the angular velocities of the virtual wheel in the x, y and z directions ($\dot{\psi}_x$, $\dot{\psi}_y$, $\dot{\psi}_z$) can be mathematically represented using the Jacobian matrix [30] as follows:

$$\begin{bmatrix} \dot{\psi}_x \\ \dot{\psi}_y \\ \dot{\psi}_z \end{bmatrix} = \begin{bmatrix} \frac{2}{3\cos\alpha} & \frac{-1}{3\cos\alpha} & \frac{-1}{3\cos\alpha} \\ 0 & \frac{1}{\sqrt{3}\cos\alpha} & \frac{-1}{\sqrt{3}\cos\alpha} \\ \frac{-1}{3\sin\alpha} & \frac{-1}{3\sin\alpha} & \frac{-1}{3\sin\alpha} \end{bmatrix} \begin{bmatrix} \dot{\psi}_1 \\ \dot{\psi}_2 \\ \dot{\psi}_3 \end{bmatrix}. \tag{8}$$

The relation between the ball angular velocities in the x, y, and z directions ($\dot{\phi}_x, \dot{\phi}_y, \dot{\phi}_z$) and the virtual wheel angular velocities in the x, y, and z directions ($\dot{\psi}_x$, $\dot{\psi}_y$, $\dot{\psi}_z$) is expressed as:

$$\begin{bmatrix} \dot{\phi}_x \\ \dot{\phi}_y \\ \dot{\phi}_z \end{bmatrix} = -\frac{r_w}{r_k} \begin{bmatrix} \dot{\psi}_x \\ \dot{\psi}_y \\ \dot{\psi}_z \end{bmatrix}. \tag{9}$$

Substituting (9) into (8) yields

$$\begin{bmatrix} \dot{\phi}_x \\ \dot{\phi}_y \\ \dot{\phi}_z \end{bmatrix} = \frac{r_w}{r_k} \begin{bmatrix} \frac{-2}{3\cos\alpha} & \frac{1}{3\cos\alpha} & \frac{1}{3\cos\alpha} \\ 0 & \frac{-1}{\sqrt{3}\cos\alpha} & \frac{1}{\sqrt{3}\cos\alpha} \\ \frac{1}{3\sin\alpha} & \frac{1}{3\sin\alpha} & \frac{1}{3\sin\alpha} \end{bmatrix} \begin{bmatrix} \dot{\psi}_1 \\ \dot{\psi}_2 \\ \dot{\psi}_3 \end{bmatrix}, \tag{10}$$

which can be used to transform the motor encoder measurements into ball angular velocities and determine its linear displacement in both the x and y directions as follows:

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} r_k \phi_y \\ -r_k \phi_x \end{bmatrix} \tag{11}$$

As illustrated in Fig. 12, the balancing high-level controller calculates acceleration commands in the x, y and z directions. These acceleration commands are integrated to obtain the virtual speed commands. Finally, the virtual speed commands

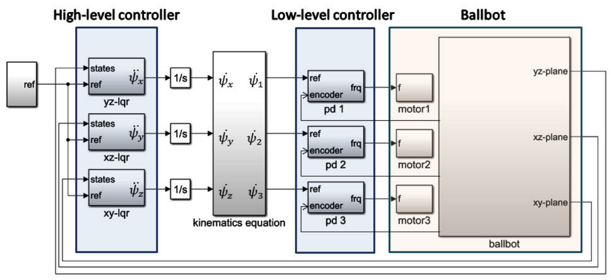


FIGURE 12. Two-level control approach. $\ddot{\psi}_{x,y,z}$ are the angular acceleration of the virtual, $\dot{\psi}_{x,y,z}$ are the angular velocity of the virtual wheel, and $\dot{\psi}_{1,2,3}$ are the angular velocity of the actual wheel.

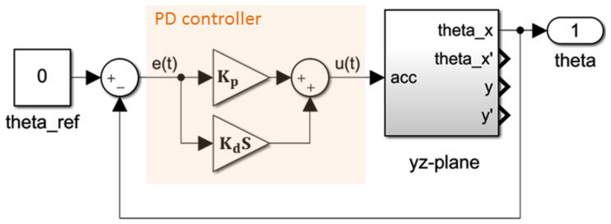


FIGURE 13. Balancing PD controller.

TABLE 3. Ballbot's known physical parameter values.

α	β	g	r_k	r_w	M_k	M_b	M_w
45	120	9.82	0.115	0.050	0.426	2.160	0.127

are transformed using (8) to obtain the corresponding references for the low-level speed controllers.

IV. PARAMETERS IDENTIFICATION

The accuracy of the dynamic model of the ballbot and its physical parameters are pivotal to the design and performance of the controller. As a result, correctly identifying the physical parameters of the ballbot is a crucial and challenging task in its development. While a CAD model can provide some of the ballbot's unknown parameters, such as inertia and center of mass, the real ballbot's values may not match entirely due to assembly and manufacturing errors. Moreover, essential unknown parameters such as friction factors cannot be obtained from the CAD model.

To overcome the difficulties of dedicated experiments for measuring the unknown parameters, we propose a practical and straightforward parameter identification procedure [6] based on Simulink and the Optimization Toolbox of MATLAB. The unknown parameters of the ballbot system which are required to be identified include the inertia, torque to acceleration constant, friction, and center of mass, i.e., $J_k, J_b, J_w, G_x, G_y, G_z, B_{vk}$, and l, see Table 2. The values of the measurable parameters are shown in Table 3, where the masses (M_k, M_b, *and* M_w) and the radii (r_k, r_b, *and* r_w) were determined using standard measurement tools, while (α, β) are the mechanical design parameters.

A. DESIGN A MODEL-FREE CONTROLLER

For parameter identification problems, there exist many techniques [31], [32], [33] for systems operating in open-loop or closed-loop conditions. Due to the inherent instability of the ballbot's dynamics, the experimental data can only be collected in a closed loop with a stabilizing controller [34]. Thus, a dedicated PD controller is designed, which can deal with the transient behavior of the system. Since it does not make explicit use of the system dynamics, its parameters are adjusted through manual tuning using a trial-and-error approach to achieve stability while gathering the necessary data for the identification process. The structure of the PD controller is shown in Fig.13. For a continuous-time PD controller, we consider the parallel form

$$u(t) = K_p e(t) + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{12}$$

For practical implementation, the derivative term is modified to be a first-order low-pass filter to reduce the noise in the derivative error and smooth out the control signal. The resulting Laplace transform is

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{NK_d}{1+\frac{N}{s}} \tag{13}$$

The resulting difference equation after discretizing the equation (13) using the Backward Euler method is

$$u[k] = \frac{1}{a_0}u[k-1] + \frac{b_0}{a_0}e[k] + \frac{b_1}{a_0}e[k-1] \tag{14}$$

where

$$\begin{aligned} b_0 &= K_p(1+NT_S) + NK_d \\ b_1 &= -(K_P + NK_d) \\ a_0 &= 1 + NT_s \end{aligned} \tag{15}$$

where N is the low-pass filter coefficient and T_s is the sampling time. Equation (14) is implemented on the ballbot high-level controller, and the PD controller gains are adjusted through manual tuning using a trial-and-error approach. The resulting gains are $K_p = 23$, $K_d = 4$, and $N = 300$, which are used to stabilize both the yz-plane and the xz-plane while collecting the required experimental data. The effectiveness of the designed PD controller is validated, as depicted in Fig.14, where the PD controller demonstrates successful balancing of the ballbot. The tilt angles θ_x and θ_y remain within $\pm 1.25°$ and are effectively maintained close to zero.

B. EXCITATION SIGNAL DESIGN

According to [34], the typical problem in closed-loop system identification is that the feedback causes correlations between the input and output variables, which may lead to biased parameter estimation. Therefore, an external excitation signal is inserted to reduce such signal correlations and to improve the estimation of the parameters. To ensure that all important modes of the ballbot system are excited and the collected data accurately reflect the system behavior and capture its important dynamics, it is necessary to use a suitably rich excitation signal. In this regard, a multi-sine signal [35]

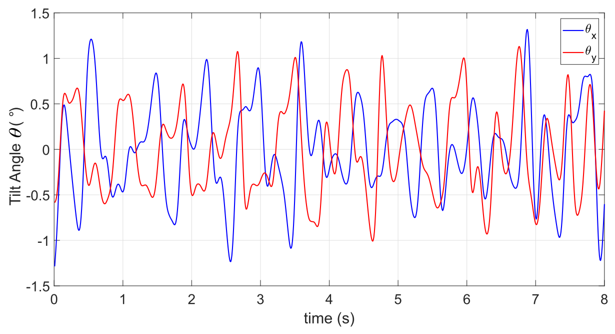


FIGURE 14. Balancing PD controller response during collecting the experimental data.

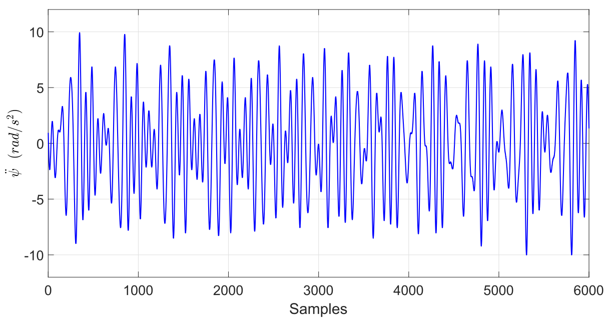


FIGURE 15. Multi−sine excitation signal to excite the *yz*−plane.

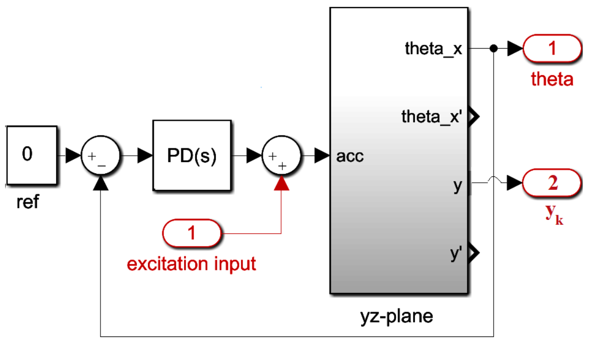


FIGURE 16. Experimental setup for parameter identification in the *yz*-plane.

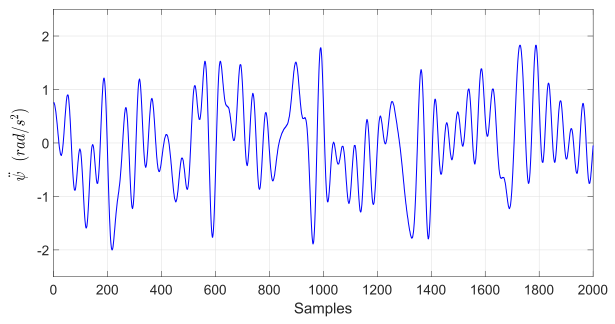


FIGURE 17. Multi−sine excitation signal to excite the *xy*−plane.

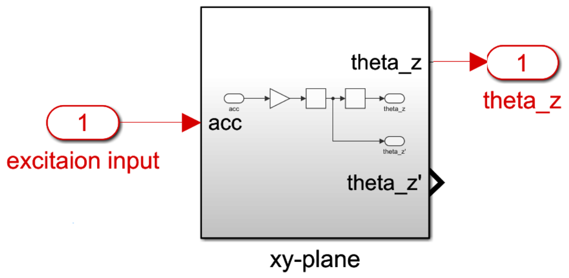


FIGURE 18. Experimental setup for parameter identification in the *xy*-plane.

is an appropriate choice, which consists of a combination of sinusoidal signals with varying frequencies, phases, and amplitudes as expressed in the following equation:

$$D[n] = \sum_{k=1}^{F} A_k \cos\left(2\pi k n f_s/N + \phi_k\right),$$
$$n = 0, \ldots, N-1 \tag{16}$$

where F represents the number of sinusoidal components, A_K is the amplitude, ϕ_k the phase of the k^{th} component, f_s represents the sampling frequency, and N is the number of samples per period. To excite the ballbot around the zero tilt angle, a multi-sine signal consisting of four sines with a frequency range of [0.5–2] Hz and a sampling frequency of 125 Hz is utilized as depicted in Fig.15. The experimental setup for closed-loop system identification is illustrated in Fig.16, in which the *yz*-plane is perturbed in a closed loop.

The controller output is disturbed by the multi-sine signal, and the corresponding data from the IMU and encoders are gathered and filtered offline with a zero-phase moving-average filter with a window size of five using the MATLAB `filtfilt` command.

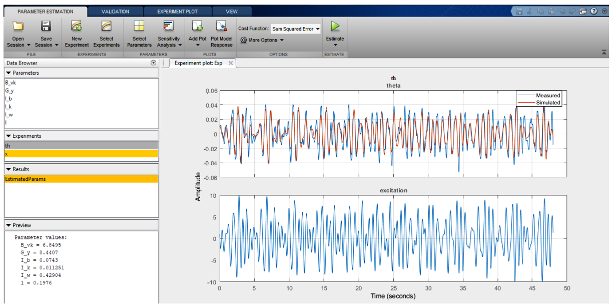
The experimental setup shown in Fig.16 is used to identify the unknown parameters of the *xz*- and *yz*-plane. Another experiment is conducted to identify the unknown parameters of the *xy*-plane in open-loop while the ballbot is balanced in closed-loop. To achieve that, the multi-sine signal shown in Fig.17 is used to disturb the control input as shown in Fig.18.

C. IDENTIFICATION PROBLEM SOLVER

The identification problem is solved by the parameter estimation tool of the Simulink Design Optimization toolbox, see Fig.19. The parameter estimation process using this tool consists of the following steps:

1) Collecting the test data from the system (experiment);
2) Specifying the parameters to be estimated (including initial guesses, parameter bounds, etc.);
3) Configuring the estimation algorithm;
4) Validating the results against different data sets.

This tool formulates the parameter estimation problem in the form of an optimization problem with constraints, and its solution is the estimated parameter values. This optimization problem consists of the following:



```
Parameter values:
  B_vk = 6.8495
  G_y = 8.4407
  I_b = 0.0743
  I_k = 0.011251
  I_w = 0.42904
  l = 0.1976
```

FIGURE 19. Parameter estimation tool.

TABLE 4. Ballbot's estimated parameters.

J_k	J_b	J_w	G_x	G_y	G_z	B_{vk}	l
0.01	0.07	0.43	8.44	8.44	2.12	6.85	0.19

- Design a vector x, the ballbot unknown parameters to be estimated;
- Objective function $F(x)$, which calculates a measure of the difference between simulated and measured responses;
- Set of constraints which specify restrictions on vector x.

The sum of square error is chosen as an objective function and is expressed as follows:

$$F(x) = \sum_{k=0}^{N} e^2(k),$$
$$e(k) = y(k) - \hat{y}(k) \tag{17}$$

where N is the number of samples, y is the measured response, and $\hat{y}$ is the simulated response. In every iteration of the optimization problem, a non-linear least squares solver tunes the values of the design variables x to minimize the specified objective function $F(x)$ subject to constraints. To ensure that the estimated values are within reasonable bounds, constraints were imposed on the parameter values within the optimization problem. The initial values of the parameters were obtained from the SOLIDWORKS 2020 CAD software and were used to initialize the optimization process. It is worth mentioning that improper initialization or bounds for the design vector x can result in unsuccessful convergence or imprecise results. Upon convergence, the ballbot's estimated parameters are presented in Table 4. Another data set was used to evaluate the accuracy of the estimated parameters, which was collected similarly to the identification data set. The validation results for the yz- and xy-plane are depicted in Figures 20 and 21, respectively. The best fit rate (BFR)% between the measured output y and simulated output $\hat{y}$ is calculated to verify these results using the following formula:

$$\text{BFR}(\%) = \max\left(0,\, 1 - \frac{\|y - \hat{y}\|}{\|y - \text{mean}(y)\|}\right) \cdot 100 \tag{18}$$

The values of BFR% between y and $\hat{y}$ of the yz- and xy-plane are 54.5% and 75.5%, respectively. The relatively lower BFR can be attributed to minor phase shifts, sensor noise, low amplitude of the signal, and the inherent simplifications of the planar model, which does not account for coupling between axes.

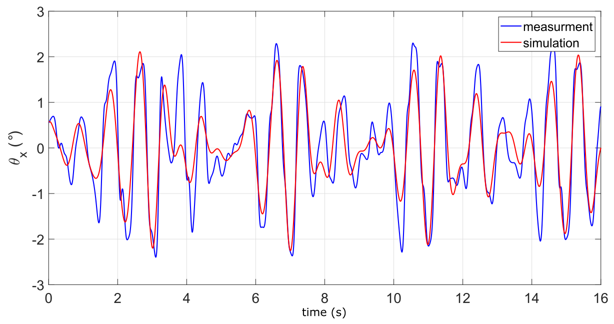


FIGURE 20. The $yz-$plane validation result with a BFR of 54.4% and RMSE of 0.0091.

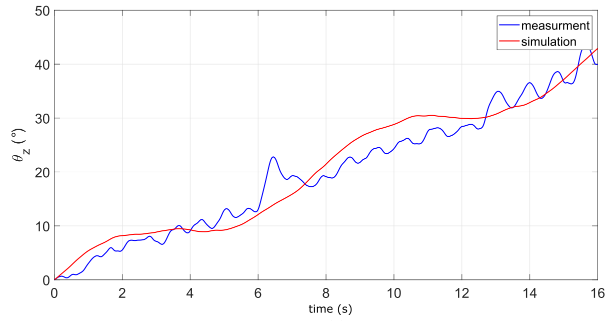


FIGURE 21. The xy-plane validation result with a BFR of 75.5% and RMSE of 0.0495.

To better assess the model accuracy, we also compute the Root Mean Square Error (RMSE), which provides a more direct measure of the absolute error in the same unit as the measured variable. The RMSE is given by:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\big(y(k) - \hat{y}(k)\big)^2} \tag{19}$$

The RMSE for the yz- and xy-plane are 0.0091 and 0.0495, respectively, indicating that the simulated output closely follows the measured response with minimal deviation. This low RMSE supports the adequacy of the estimated parameters.

Moreover, the effectiveness of the identified model is further validated through the experimental implementation of the model-based controller, as discussed in Section VI. The controller was able to maintain system stability, successfully reject external disturbances, and track reference commands in real-time. These experimental outcomes demonstrate that the identified model captures the essential dynamics required for high-performance control.

In conclusion, although the BFR is moderate for the yz-plane, the combination of low RMSE and strong experimental performance confirms the suitability and reliability of the identified model for practical control applications.

V. CONTROL ALGORITHM DESIGN

In the previous section, a PD controller is considered to balance the robot during the experiments for parameter identification; however, it does not explicitly use the model dynamics. Therefore, a more accurate model-based controller, which can represent the system's dynamics, will be developed to balance the robot and achieve the controller's goals.

A. LINEARIZATION

The use of linear approximations in this work, specifically the design of an LQR controller, is a deliberate choice that prioritizes analytical simplicity, modularity, and real-time feasibility on embedded hardware. While the ballbot system is inherently nonlinear, this linear control design enables rapid implementation and predictable stability behavior, making it a suitable baseline for systematic development and future integration of more advanced nonlinear or robust control strategies.

To be able to develop a linear controller for the ballbot, the nonlinear dynamic Equation (5) has to be linearized. The yz-plane is considered with the state vector x that is defined as

$$x = \begin{bmatrix}\theta_x & \dot{\theta}_x & y_k & \dot{y}_k\end{bmatrix}^\top \tag{20}$$

The system is linearized around the unstable equilibrium point, where all state variables and the input are zeros, i.e.,

$$\begin{aligned}\bar{x} &= \begin{bmatrix}\bar{\theta}_x & \bar{\dot{\theta}}_x & \bar{y}_k & \bar{\dot{y}}_k\end{bmatrix}^\top = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^\top \\ \bar{u} &= 0\end{aligned} \tag{21}$$

Based on the equilibrium point given in (21), the first-order Taylor expansion is used to compute the linear state–space representation of (5) as follows:

$$\begin{aligned}\dot{x} &= Ax + Bu, \\ y &= Cx + Du\end{aligned} \tag{22}$$

where the state–space matrices A, B, C, and D are computed as:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\partial \ddot{\theta}_x}{\partial \theta_x} & \frac{\partial \ddot{\theta}_x}{\partial \dot{\theta}_x} & \frac{\partial \ddot{\theta}_x}{\partial y_k} & \frac{\partial \ddot{\theta}_x}{\partial \dot{y}_k} \\ 0 & 0 & 0 & 1 \\ \frac{\partial \ddot{y}_k}{\partial \theta_x} & \frac{\partial \ddot{y}_k}{\partial \dot{\theta}_x} & \frac{\partial \ddot{y}_k}{\partial y_k} & \frac{\partial \ddot{y}_k}{\partial \dot{y}_k} \end{bmatrix}, B = \begin{bmatrix} 0 \\ \frac{\partial \ddot{\theta}_x}{\partial u} \\ 0 \\ \frac{\partial \ddot{y}_k}{\partial u} \end{bmatrix},$$

$$C = I_{4\times1}, \quad D = 0_{4\times1}, \quad \frac{\partial \ddot{\square}}{\partial \square}\Big|_{(x,u)=(\bar{x},\bar{u})}. \tag{23}$$

The expressions for $\ddot{\theta}_x$ and $\ddot{y}_k$ in (23) can be obtained from (5) as:

$$\begin{aligned}\ddot{q}_x &= \begin{bmatrix}\ddot{y}_k \\ \ddot{\theta}_x\end{bmatrix} \\ &= M(q_x)^{-1}\left(Q_x G_x \ddot{\psi}_x - C(q_x, \dot{q}_x)\dot{q}_x - D(\dot{q}_x) - G(q_x)\right)\end{aligned} \tag{24}$$

Substituting the state–space matrices A and B in (23) with the numerical values from Tables 3 and 4, we obtain the following state–space matrices for both the yz-plane and the xz-plane models:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 14.1 & 0 & 0 & -2.54 \\ 0 & 0 & 0 & 1 \\ 1.55 & 0 & 0 & -0.32 \end{bmatrix}, B = \begin{bmatrix} 0 \\ -2.66 \\ 0 \\ 0.67 \end{bmatrix}$$

$$C = I_4 \quad, D = 0_{4\times1}. \tag{25}$$

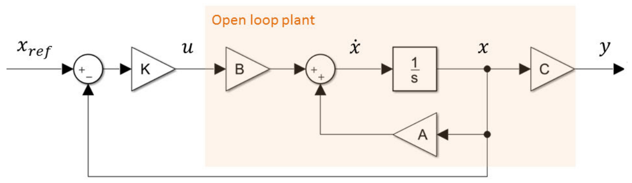


FIGURE 22. Full state feedback controller.

It is worth mentioning that the state−space matrices computed in (25) can also be obtained by using the MATLAB command `linmod`. In addition, regarding the xy-plane, it has a linear dynamic Equation (7) with a state vector $x = \begin{bmatrix}\theta_z & \dot{\theta}_z\end{bmatrix}^\top$ and the following state–space matrices:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 2.122 \end{bmatrix} C \qquad = I_2, D = 0_{2\times1}$$

To check the stability of the ballbot system, the eigenvalues of system matrix A in (25) for the yz-plane/xz-plane have to be determined as follows:

$$\det(A - \lambda I) = 0 \tag{26}$$

which yields $\lambda_1 = 0$, $\lambda_2 = 3.6197$, $\lambda_3 = -3.8981$, *and* $\lambda_4 = -0.0390$. This indicates that one of the poles is in the right-half plane, which makes the open-loop system unstable.

For designing an efficient controller for the ballbot system, the stabilizability and detectability of the system have to be examined. To achieve that, the ranks of controllability and observability matrices are computed using the MATLAB commands: `rank(ctrb(A,B))` and `rank(obsv(A,C))`, respectively. This yields a full rank in both cases, which indicates that the ballbot system is stabilizable and detectable.

B. LQR CONTROLLER AND ITS SIMULATION RESULTS

States of the ballbot can be measured directly from the sensors (IMU and encoders attached to the motors); thus, a full state feedback controller can be designed, as shown in Fig.22, where all states are used in the feedback path. The control input is implemented as:

$$u = -K(x - x_{\text{ref}}) \tag{27}$$

where K is the state feedback vector, and x_{ref} is the reference state vector. Substituting (27) in (22) yields the following state–space equations for the closed-loop feedback system:

$$\dot{x} = (A - BK)x + BKx_{\text{ref}}$$

$$= A_{cl}x + BKx_{\text{ref}}$$
$$y = Cx \tag{28}$$

The time domain performance and stability of the closed-loop system are determined by the system matrix A_{cl} and the location of its eigenvalues. These closed-loop poles can be placed anywhere to satisfy stability and a desired level of performance by choosing an appropriate K.

The state feedback vector K directly influences the stability of the system, which can be obtained by a manual pole-placement method to satisfy stability and desired performance. Alternatively, it can be optimized by minimizing a quadratic cost function that combines the state error and control effort; the resulting controller is known as the linear quadratic regulator (LQR).

The yz-plane is considered now with the state vector $x = \begin{bmatrix}\theta_x & \dot{\theta}_x & y_k & \dot{y}_k\end{bmatrix}^\top$. In LQR, the feedback control law is given by $u = -K(x - x_{\text{ref}})$, where u is the control input, x_{ref} represents the reference state vector, which is ideally zero for balancing and station-keeping objectives, and K denotes the optimal feedback gain vector for the LQR controller. The LQR controller seeks to minimize a quadratic cost function subject to the ballbot dynamics (25):

$$J(u) = \int_0^\infty \left((x - x_{\text{ref}})^\top Q (x - x_{\text{ref}}) + u^\top R\, u \right) dt \tag{29}$$

where Q is the state weighting matrix, and R is the control weighting matrix. The solution to the algebraic Riccati equation provides the optimal feedback gain vector for the LQR controller, which is defined as follows:

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0. \tag{30}$$

The solution to the Riccati equation is the positive definite matrix P. Once the P matrix is determined, the optimal feedback gain matrix K can be calculated as $K = PBR^{-1}$. By tuning the weighting matrices Q and R, we can adjust the controller's behavior to achieve the desired performance characteristics. In order to tune the Q and R matrices, assigning significant weights to the $(y_k, \dot{y}_k)$ states in the Q matrix limits the ballbot's ability to maintain balance. Conversely, assigning significant weights to $(\theta_x, \dot{\theta}_x)$ states enhances the balancing performance but compromises control over the robot's position. Additionally, assigning high weights to the derivative states $(\dot{\theta}_x, \dot{y}_k)$ results in jerky and aggressive motion due to the noise amplification. It is important to note that increasing the R matrix will decrease the control input and increase the steady-state time. Considering these issues, appropriate values of Q and R are obtained as $R = 0.1$ and Q is a matrix of ones with a main diagonal of [6.5, 1, 1.2, 1] for the yz-plane/xz-plane. Using the `lqr` MATLAB command yields the following optimal gain vector:

$$K = \begin{bmatrix}-18.07 & -5.20 & -3.53 & -2.39\end{bmatrix}$$

During the real-time experiments, we noticed that the robot was too aggressive with these gains, so we decreased the gain corresponding to the $\dot{\theta}$ state to -2.5.

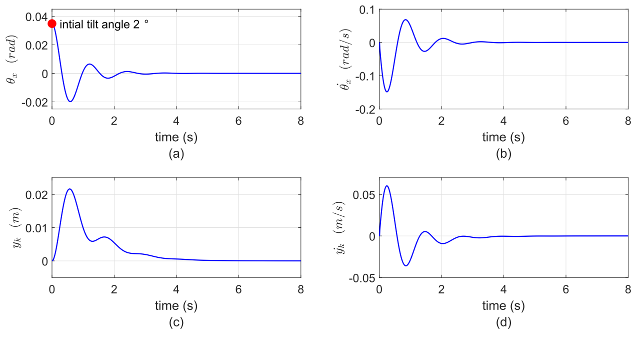


FIGURE 23. Simulation results for balancing and station−keeping control with an initial 2° tilt angle: (a) the robot's tilt angle around the x−axis, (b) the angular rate of the robot's tilt angle around the x−axis, (c) the robot's position in the y direction, and (d) the robot's velocity in the y direction.

To investigate the performance of the proposed LQR controller, different simulations are conducted using MATLAB/Simulink with the ballbot parameters listed in Tables 3 and 4. The first simulation investigates the capability of the robot to return to its zero tilt angle when starting from an initial tilt angle (2°). In such a scenario, zero references are selected for all the system states. Fig.23 shows that the proposed LQR controller can regulate all the system states within a short time and can maintain the ballbot balance.

The second simulation examines the positional transfer ability of the ballbot. In this case, the ballbot is commanded to move in a straight path from the origin to a target position ($x = 0$ m, $y = 0.5$ m). As illustrated in Fig.24c,d, there is an undershoot response observed in both the body position and velocity, primarily due to the non-minimum phase characteristics of the ballbot. However, Fig.24c demonstrates a steady-state error of 0.15 m in the robot's position in the y direction, indicating that the robot failed to reach its target position. This limitation arises from the under-actuated nature of the ballbot and its dynamic constraints, which necessitate maintaining a specific tilt angle while moving toward the target position. These results demonstrate that the proposed LQR controller regulates the robot tilt angle state to its zero reference before the robot reaches its target position. Consequently, the proposed LQR controller is unable to simultaneously achieve both balance and accurate position tracking.

C. TWO-LOOP CONTROL AND ITS SIMULATION RESULTS

By using a single LQR controller, the balancing, station keeping, and positional transfer are reduced to one regulation problem, where the LQR controls all states simultaneously. Therefore, the previous simulations indicate that the LQR controller can regulate all the states successfully and is sufficient for balancing and station-keeping goals but is inadequate for robot transfer. Therefore, it is essential to incorporate an additional controller to generate the required tilt angle reference based on the position reference. The LQR controller, which is designed in the previous section, is used

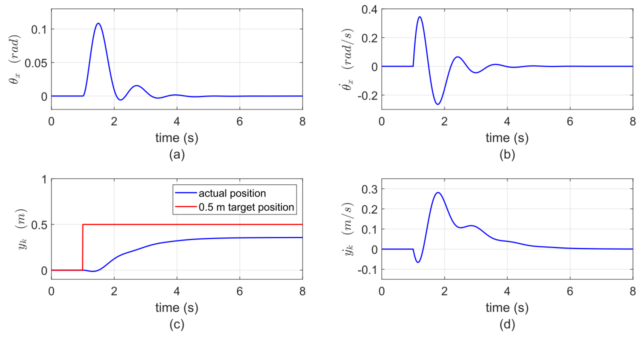


FIGURE 24. Simulation results for positional transfer control using the LQR controller: (a) the robot's tilt angle around the x−axis, (b) the angular rate of the robot's tilt angle around the x−axis, (c) the robot's position in the y direction, and (d) the robot's velocity in the y direction.

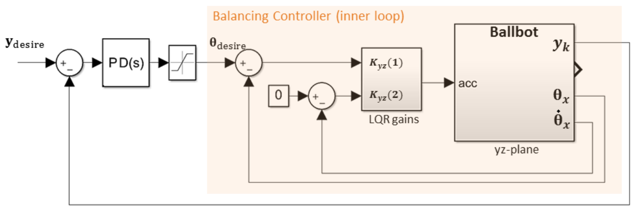


FIGURE 25. Structure of the two−loop controller, where the inner loop is an LQR controller and the outer loop is a PD controller.

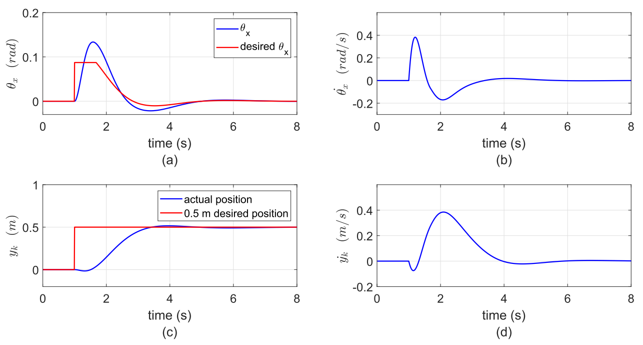


FIGURE 26. Simulation results for positional transfer using the two−loop controller: (a) the robot's tilt angle around the x−axis, (b) the angular rate of the robot's tilt angle around the x−axis, (c) the robot's position in the y direction, and (d) the robot's velocity in the y direction.

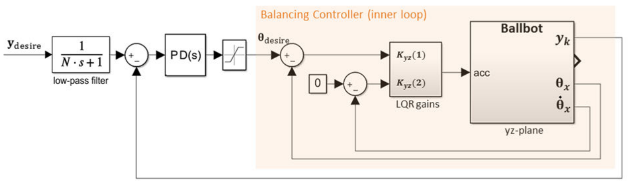


FIGURE 27. Structure of the two-loop controller with a low-pass-filter to smooth out the position reference.

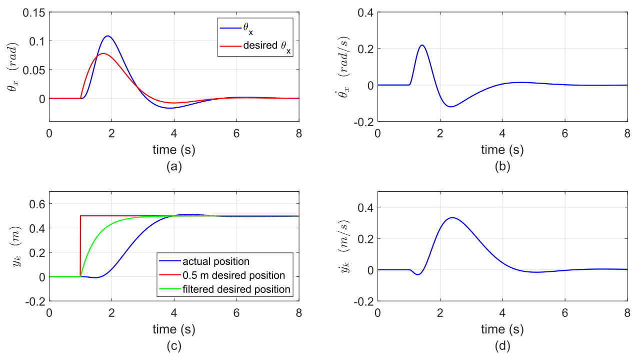


FIGURE 28. Simulation results for positional transfer using the two−loop controller with a low−pass filter: (a) the robot's tilt angle around the x−axis, (b) the angular rate of the robot's tilt angle around the x−axis, (c) the robot's position in the y direction, and (d) the robot's velocity in the y direction.

in the inner loop for balancing with state feedback gain $K = \begin{bmatrix}-18.07 & -2.5 & 0 & 0\end{bmatrix}$; however, a PD controller is used in the outer loop for positional transfer and station-keeping.

The PD controller tracks the desired robot position by computing the desired body tilt angle for the inner loop depending on the error between the desired position y_{desire} and the current position y_k. The output of the PD controller is saturated ($-6° < \theta < 6°$) to avoid a large tilt angle and ensure the robot balance. The Simulink control design toolbox is used to tune the PD gains based on the Simulink model of the ballbot system. The resulting controller gains are $K_p = 0.159$ and $K_d = 0.0628$. In this way, the control system consists of an inner loop for balancing control and an outer loop for positional transfer, as shown in Fig.25. Two simulations are conducted to assess the performance of the two-loop controller approach. In both simulations, it is desired to move the ballbot in a straight line from the origin to a target position ($x = 0$ m, $y = 0.5$ m). As illustrated in Fig.26a, the outer-loop controller (PD) produces a high desired tilt angle, which is saturated to 6°. This high tilt angle is mainly due to sudden and large changes in the position reference. Fig.26b shows that a two-loop controller can yield the position output on the desired position with a slight overshoot and without steady-state error. In the second simulation, a low-pass filter is used as shown in Fig.27 to smooth out the position reference, reduce the required tilt angle, and ensure the robot balance. Fig.28 demonstrates that the two-loop controller with the low-pass filter can track the smoothed position reference efficiently within a short settling time (3 s).

D. HEADING MOTION CONTROL AND ITS SIMULATION RESULTS

The xy-plane is considered now with the state vector $x = \begin{bmatrix}\theta_z & \dot{\theta}_z\end{bmatrix}^\top$, and the LQR controller is implemented as $u = -K(x - x_{\text{ref}})$, where x_{ref} is a reference state vector, which should be zero for the fixed heading angle, and K is the state feedback gain vector. In such a control problem, the tuning parameters Q and R are chosen as $R = 1.2$ and $Q = \text{diag}(2, 1)$, and the resulting optimal gain vector for the xy-plane is:

$$K = \begin{bmatrix}1.2910 & 1.4318\end{bmatrix}$$

Two different simulation scenarios are considered here. The first one assesses the ballbot's ability to return to its zero yaw angle from an initial yaw angle of 30°. As depicted

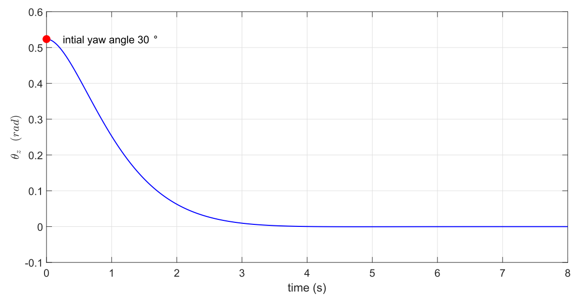


FIGURE 29. Simulation result for heading motion control in the *xy*−plane with an initial 30° yaw angle.

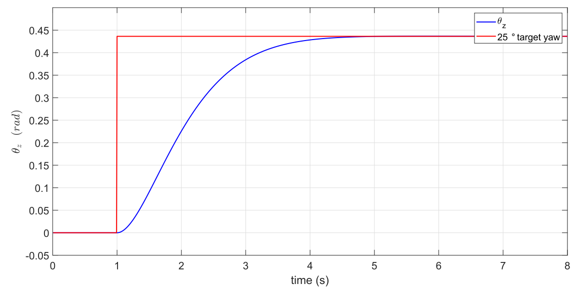


FIGURE 30. Simulation result for heading tracking control in the *xy*−plane.

in Fig.29, the ballbot can reach its zero yaw reference successfully. The second simulation assesses the ability of the robot to track a desired yaw angle. In such a case, the robot is commanded to rotate 25° around the *z*-axis. As depicted in Fig.30, the LQR controller can efficiently yield the orientation angle θ_z on the desired setpoint ($\theta_z = 25°$) with a slight overshoot and without steady-state error.

VI. TESTING AND VALIDATION

In this section, the robustness and reliability of the designed controllers in the previous section are verified experimentally under various conditions. This experimental validation also serves to assess the accuracy of the identified parameters of the ballbot model. The experiments are structured around five core control objectives:

- Maintaining tilt angles close to zero;
- Rejecting external disturbances;
- Compensating for variations in system parameters due to environmental changes;
- Achieving positional transfer;
- Controlling the heading motion.

These objectives were selected to evaluate the fundamental capabilities of the proposed systematic model-based approach, which integrates model identification, controller design, and hardware implementation. The results demonstrate the effectiveness of this modular and low-complexity design in real-world conditions and lay the groundwork for future extensions involving advanced control strategies.

Several experiments were conducted with real-time data collected from the robot depicted in Fig. 5a. The omni-wheel encoders were used to obtain the ball position and velocity data, while the IMU was used to gather body tilt angles and angular velocity data. All experimental data were collected through the PC serial port at a frequency of 125 Hz. It should be noted that unless otherwise stated, all the experiments were carried out on a surface covered with carpet.

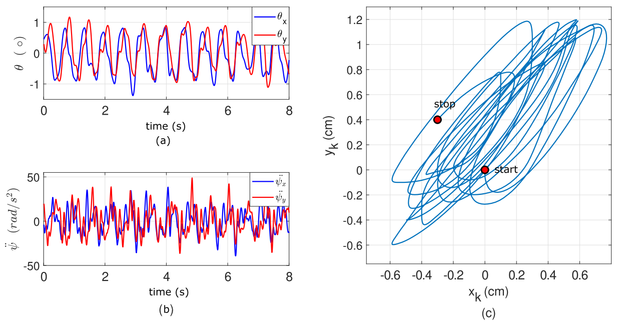


FIGURE 31. Station keeping and maintaining zero tilt angle controller performance: (a) the tilt angles of the robot, (b) the accelerations of the virtual wheel in the *x* and *y* directions, and (c) the position of the robot in the *xy*−plane.

A. STATION KEEPING AND MAINTAINING ZERO TILT ANGLE

This experiment aims to utilize the proposed LQR controller to stabilize the ballbot at its initial position. The reference inputs for all states in equation (27) were set to zero. The body tilt angles θ_x and θ_y were kept within a range of $\pm 1°$ as illustrated in Fig. 31a. The virtual wheel accelerations in the *x* and *y*-directions are presented in Fig. 31b. Fig. 31c illustrates the ballbot's movement in the *xy*-plane, where the robot maintained its position within a circle of radius 1.2 cm around the origin throughout the entire balancing process.

B. REJECTING EXTERNAL DISTURBANCES

In this experiment, the ability of the ballbot system to reject disturbances using the LQR controller is evaluated. To generate a random disturbance, the robot was intentionally pushed, causing an immediate increase in the control effort to counteract the external forces and regain stability. As illustrated in Fig. 32a, the tilt angles increased rapidly and deviated up to $\pm 5°$. Fig. 32b illustrates the corresponding positional changes, showing drift towards a new position ($x = -8$ cm, $y = -8$ cm), and returning to the initial position within approximately 3 s with an error of less than 1 cm. These results demonstrate that the LQR controller can effectively reject external disturbances, enabling the ballbot to recover and return to its original position in a reasonable amount of time.

C. OVERCOMING VARIATIONS IN MODEL PARAMETERS DUE TO ENVIRONMENTAL CHANGES

A series of experiments were conducted to evaluate the robustness of the ballbot with the proposed LQR controller against uncertainties in model parameters. In the first experiment, the ground friction was changed, where the robot

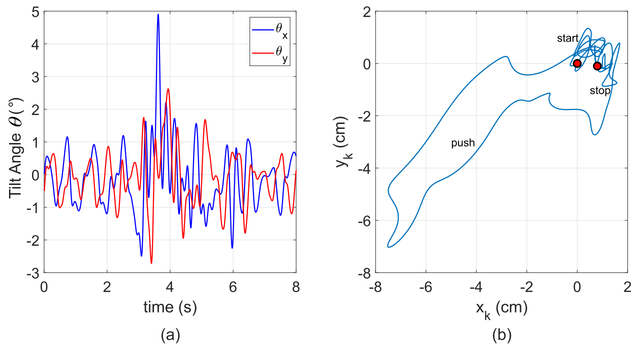


FIGURE 32. Rejecting external disturbances (push) controller performance: (a) the tilt angles of the robot, (b) the position of the robot in the *xy*−plane.

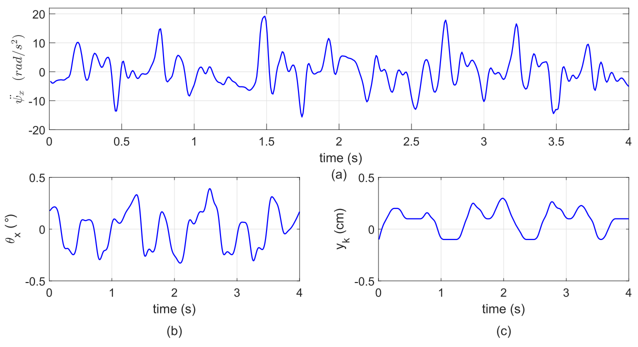


FIGURE 33. Overcoming variations in model parameters resulting from changes in the environment (ceramic surface): (a) the acceleration of the virtual wheel in the *x* direction, (b) the tilt angles θ_x of the robot, and (c) the position of the robot in the *y* direction.

was commanded to operate on a ceramic surface with a lower friction coefficient than the carpet. This resulted in reduced control effort needed to balance, resulting in a minimal tilt angle deviation of ±0.4°, and a small positional deviation of ±0.3 cm error as illustrated in Fig. 33a–c. In the second experiment, additional masses were placed on the robot's top to alter its center of mass. As depicted in Fig. 34, the robot maintained its balance for masses up to 0.5 kg. The third experiment evaluated the ability of the ballbot system to balance while using a new ball with different inertia, texture, and mass. As demonstrated in Fig. 35, the robot was able to maintain its balance despite the new ball's characteristics. Overall, these results demonstrate the robustness of the proposed controller in the presence of model uncertainties.

D. ACHIEVING POSITIONAL TRANSFER

In this experiment, the performance of the robot was evaluated for positional transfer. The objective was to move the robot in a straight path to a target position ($x = 0$ m, $y = 0.5$ m). As demonstrated in Fig. 36, the experimental results are consistent with the simulation results discussed in Section V-B (see Fig. 24). However, a steady-state error of 0.15 m was observed in the y direction, causing the robot to miss the target position. Such a response is attributed

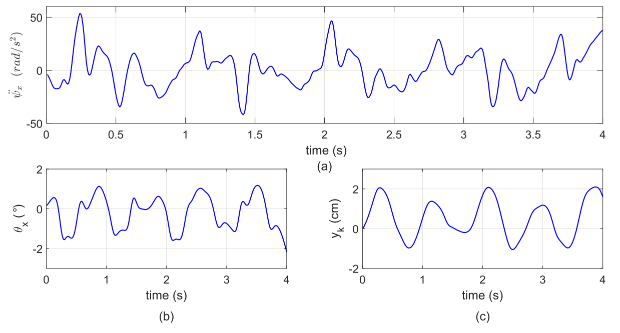


FIGURE 34. Overcoming variations in model parameters resulting from changes in the environment (loaded 0.5 kg mass): (a) the acceleration of the virtual wheel in the *x* direction, (b) the tilt angles θ_x of the robot, and (c) the position of the robot in the *y* direction.

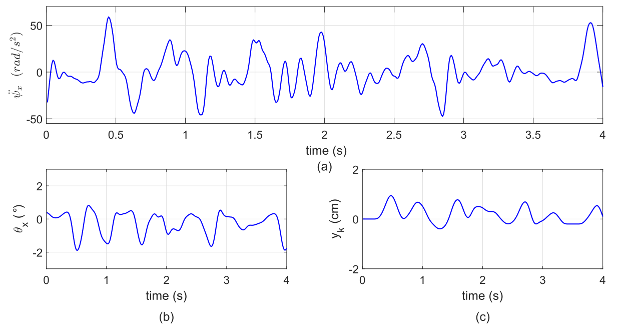


FIGURE 35. Overcoming variations in model parameters resulting from changes in the environment (new ball): (a) the acceleration of the virtual wheel in the *x* direction, (b) the tilt angles θ_x of the robot, and (c) the position of the robot in the *y* direction.

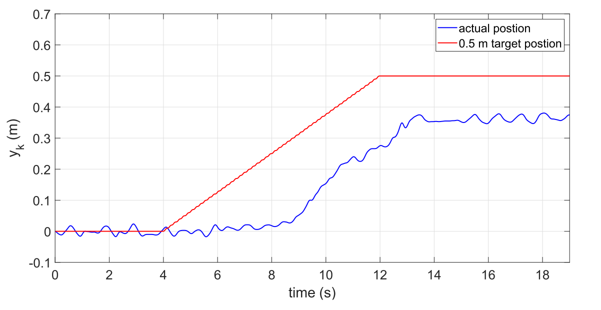


FIGURE 36. Achieving positional transfer using the LQR controller.

to the LQR controller regulating all states simultaneously, attempting to maintain its balance while moving, which is not a realistic behavior since the robot needs to maintain a certain tilt angle while moving. When trying to solve this problem by setting high weights on the $(y_k, \dot{y}_k)$ states in the Q matrix, the robot's ability to maintain balance was reduced.

The simulation and experimental results demonstrate that the proposed LQR controller is unable to achieve balancing and positional transfer simultaneously. To address this issue, the aforementioned experiment was repeated using the two-loop approach described in Section V-C, where the outer-loop controller (PD) tracks the desired robot position by generating a desired body tilt angle for the inner-loop

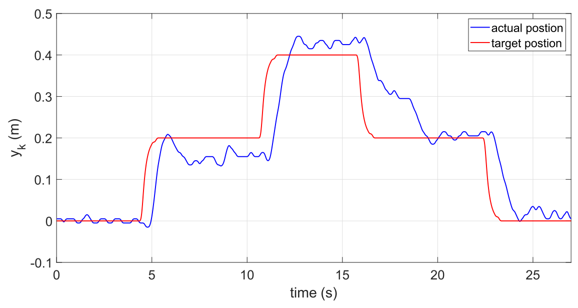


FIGURE 37. Achieving positional transfer using the two-loop controller.

controller (LQR) based on the error between the desired position y_{desire} and the current position y_k. As depicted in Fig. 37, the proposed controller is reasonably able to achieve balancing and positional transfer simultaneously.

The observed limitations in tracking precision are not solely due to the controller design but are also influenced by several mechanical factors inherent to the current hardware setup. First, slipping between the omni-wheels and the spherical ball occurs during rapid motor actuation. This slippage leads to loss of effective traction, causing discrepancies between intended and actual motion. Second, the use of a soft, deformable football as the balancing sphere introduces compliance and local deformation under load, resulting in uneven rolling and jerky movement during dynamic transitions. Third, the commercially available omni-wheels used in the prototype consist of discrete rollers with non-continuous contact surfaces. This segmented structure leads to vibrations and an unstable rolling response. Together, these mechanical imperfections introduce unmodeled dynamics that are not captured by the planar control model, thereby affecting the overall system performance.

E. CONTROLLING THE HEADING MOTION

The previous experiments focused on evaluating the performance of the closed-loop system in the xz- and yz-plane. This particular experiment, on the other hand, examines the heading motion control in the xy-plane using the proposed LQR controller. In such a case, it is desired to make the ballbot system track a specific yaw angle (θ_z). Fig. 38a shows that the tilt angles θ_x and θ_y are maintained around zero within $\pm 1.5°$ while the ballbot reaches its heading setpoint, Fig. 38b shows that the heading angle θ_z is successfully regulated to its setpoint ($\theta_z = 25°$) within a reasonable settling time and without steady-state error. This experiment demonstrates that the robot can efficiently balance itself and also rotate around the vertical z-axis simultaneously.

F. DISCUSSION

The numerical results of the experiments conducted in Section VI of this work provide strong validation for the correctness and validity of the derived dynamical model, which is based on the estimated parameters and also demonstrate the effectiveness and robustness of the proposed controller for various control objectives in the ballbot system. In the sequel, we synthesize the findings and discuss their implications:

- Maintaining zero tilt angle with station keeping:
  - The LQR controller successfully maintains the ballbot's stability around its origin position, with tilt angles maintained around zero within $\pm 1°$.
  - The control inputs effectively keep the ballbot within a circle of radius 1.2 cm around the origin throughout the balancing operation.
- External disturbance rejection:
  - The LQR controller shows the capability to reject external disturbances effectively.
  - After a random disturbance, the ballbot deviates up to $\pm 5°$ but returns to its zero initial position within about 3 s with an error in position less than 1 cm.
- Varying model parameters due to environmental changes:
  - The LQR controller demonstrates robustness against uncertainty in model parameters.
  - Experimentation with different friction levels and additional masses shows minimal deviation from the desired behavior, showcasing its reliability in real-world conditions.
- Point-to-point transfer:
  - The limitation observed in the point-to-point transfer experiment highlights the challenge of simultaneously balancing and achieving precise position tracking.
  - While the LQR controller fails to achieve precise point-to-point transfer due to its focus on maintaining a zero tilt angle during movement, a two-loop approach (outer PD loop and inner LQR loop) effectively balances the robot while tracking the desired position.
- Heading motion control:
  - The LQR controller successfully regulates the ballbot's heading motion while maintaining balance, allowing it to track a specific yaw angle efficiently.

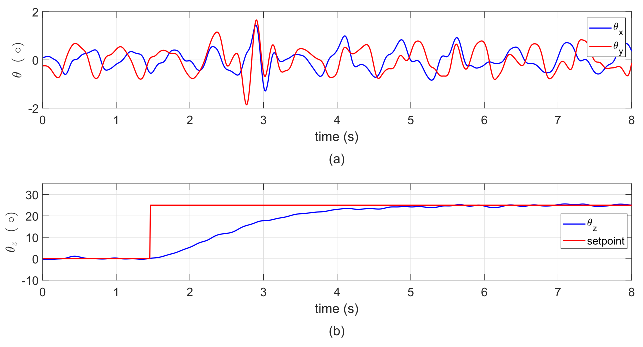


FIGURE 38. Controlling the heading motion controller performance: (a) the tilt angles of the robot, (b) the yaw angle θ_z of the robot.

- The ability to simultaneously balance and rotate around the vertical axis expands the operational capabilities of the ballbot system.

In summary, the experiments collectively demonstrate the versatility and adaptability of the LQR controller in handling various control objectives and environmental conditions. The controller's ability to maintain stability, reject disturbances, and adapt to changing parameters highlights its robustness and effectiveness. The results also underscore that the estimated physical parameters adequately capture the essential dynamics of the system.

VII. CONCLUSION AND FUTURE WORK

This paper has introduced a complete and straightforward systematic approach for model-based control of the ballbot. The proposed approach was experimentally evaluated on a 3D printed ballbot system.

The closed-loop system is structured as two-level controllers, where a low-level speed controller is implemented for each motor to track the reference speeds computed by the high-level balance controller. This approach allows setting a speed profile to the motors, such as a sigmoid function, which effectively minimizes jerk and results in smoother motion of the ballbot. Moreover, the ballbot's equations of motion do not incorporate stepper motor dynamics. As a result, this approach minimizes the overall dynamical order of the closed-loop system, simplifies the real-time implementation, and enhances the modularity and execution speed of the control system. The Euler–Lagrange approach is utilized to derive the dynamic equations of the ballbot planar model, and the estimation of its unknown parameters $J_k, J_b, J_w, G_x, G_y, G_z, B_{vk}$, and l is successfully achieved using experimental data.

By using a single LQR controller, the balancing, station keeping, and positional transfer objectives are reduced to a single regulation problem, where the LQR controls all states simultaneously. Results from both simulations and experiments demonstrate that the proposed LQR controller effectively regulates all states and is adequate for accomplishing the balancing and station-keeping objectives. However, it is insufficient for achieving the positional transfer objective; hence, it is essential to incorporate an additional controller to generate the required tilt angle reference based on the position reference.

Various experiments have been conducted under different conditions to validate the proposed model and evaluate the reliability and robustness of the control algorithms. The results demonstrate that the ballbot system is adequately represented by the planar model and the proposed model-based controllers are sufficient to stabilize the ballbot and demonstrate their robustness against model uncertainties and disturbances.

In the conclusion of this work, the current state of our ballbot provides opportunities for further enhancements, particularly in addressing the issues identified during testing. These concerns are identified as potential areas for future research:

1) An issue of slipping between the ball and the omni-wheels arose during rapid motor rotations, causing the omni-wheels to spin without propelling the ball. To address this, a ball arrester could be employed to press the ball against the omni-wheels, thereby enhancing the frictional force between them and preventing the ball from being pushed out.
2) The available commercial omni-wheels have gaps between their smaller individual wheels, resulting in an unstable rolling motion that is unsuitable for ballbot development. Thus, there is a need for more advanced omni-wheels with a continuous circumferential contact line.
3) The deformation of the ball (football) caused by the body weight results in undesired jerky motion that could be solved by using a rigid ball, such as a bowling ball.
4) The proposed systematic procedure is currently applied to a planar model but is well-suited for extension to a fully coupled 3D dynamic framework. Future work will focus on this extension to improve performance in complex and coupled motion scenarios.
5) The current work adopts a planar model to facilitate analysis and real-time implementation. Future efforts will include more detailed motor modeling to improve tracking accuracy in high-speed or variable-load scenarios and increase adaptability to different actuation platforms.
6) The current control strategy provides a modular and reliable foundation for expansion. Future improvements will include benchmarking the current design against other advanced controllers such as robust, nonlinear, and fuzzy logic controllers, including fuzzy Type-2 and Type-3 systems, as well as model predictive control (MPC).
7) Additionally, the integration of high-level functions such as trajectory planning, terrain adaptation, and obstacle avoidance will be explored to extend the robot's capability to dynamic and unstructured environments.
8) Finally, the implementation and experimental validation of advanced nonlinear or hybrid control strategies will be investigated to expand the ballbot's performance in more demanding conditions.

REFERENCES

[1] R. R. Murphy, V. B. M. Gandudi, and J. Adams, "Applications of robots for COVID-19 response," 2020, *arXiv:2008.06976*.

[2] Y. Nakata, S. Yagi, S. Yu, Y. Wang, N. Ise, Y. Nakamura, and H. Ishiguro, "Development of 'ibuki' an electrically actuated childlike Android with mobility and its potential in the future society," *Robotica*, vol. 40, no. 4, pp. 933–950, Apr. 2022.

[3] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2006, pp. 2884–2889.

[4] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," in *Proc. Int. Conf. Control, Autom. Syst.*, Hong Kong, Oct. 2008, pp. 433–438.

[5] A. Bhatia, M. Kumagai, and R. Hollis, "Six-stator spherical induction motor for balancing mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 226–231.

[6] M. A. Alyousify, H. S. Abbas, M. M. M. Hassan, and M. H. Amin, "Parameter identification and control of a ball balancing robot," in *Proc. 8th Int. Conf. Mechatronics Robot. Eng. (ICMRE)*, Munich, Germany, Feb. 2022, pp. 91–97.

[7] D. Pratama, E. H. Binugroho, and F. Ardilla, "Movement control of two wheels balancing robot using cascaded PID controller," in *Proc. Int. Electron. Symp. (IES)*, Surabaya, Indonesia, Sep. 2015, pp. 94–99.

[8] M. A. Rosyidi, E. H. Binugroho, S. E. Radin Charel, R. S. Dewanto, and D. Pramadihanto, "Speed and balancing control for unicycle robot," in *Proc. Int. Electron. Symp. (IES)*, Sep. 2016, pp. 19–24.

[9] T. K. Jespersen, "Kugle-modelling and control of a ball-balancing robot," M.S. thesis, Dept. Control Automation, Aalborg Univ., Aalborg, Denmark, 2019.

[10] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot, "Modeling and control of a ballbot," B.S. thesis, Eidgenössische Technische Hochschule Zürich, 2010.

[11] D. B. Pham, H. Kim, J. Kim, and S.-G. Lee, "Balancing and transferring control of a ball segway using a double-loop approach [applications of control]," *IEEE Control Syst. Mag.*, vol. 38, no. 2, pp. 15–37, Apr. 2018.

[12] A. N. Inal, Ö. Morgül, and U. Saranli, "A 3D dynamic model of a spherical wheeled self-balancing robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Algarve, Portugal, Oct. 2012, pp. 5381–5386.

[13] A. Bonci, "New dynamic model for a ballbot system," in *Proc. 12th IEEE/ASME Int. Conf. Mech. Embedded Syst. Appl. (MESA)*, Auckland, New Zealand, Aug. 2016, pp. 1–6.

[14] C.-C. Tsai, C.-K. Chan, and L.-C. Kuo, "LQR motion control of a ball-riding robot," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Kaohsiung, Taiwan, Jul. 2012, pp. 861–866.

[15] I. Lal, M. Nicoara, A. Codrean, and L. Busoniu, "Hardware and control design of a ball balancing robot," in *Proc. IEEE 22nd Int. Symp. Design Diag. Electron. Circuits Syst. (DDECS)*, Apr. 2019, pp. 1–6.

[16] C.-H. Chiu and W.-R. Tsai, "Design and implementation of an omnidirectional spherical mobile platform," *IEEE Trans. Ind. Electron.*, vol. 62, no. 3, pp. 1619–1628, Mar. 2015.

[17] Y.-F. Peng, C.-H. Chiu, W.-R. Tsai, and M.-H. Chou, "Design of an omni-directional spherical robot: Using fuzzy control," in *Proc. Int. Multiconference Eng. Comput. Scientists*, vol. 1, 2009, pp. 18–20.

[18] H. Navabi, S. Sadeghnejad, S. Ramezani, and J. Baltes, "Position control of the single spherical wheel mobile robot by using the fuzzy sliding mode controller," *Adv. Fuzzy Syst.*, vol. 2017, pp. 1–10, Jan. 2017.

[19] C. Cai, J. Lu, and Z. Li, "Kinematic analysis and control algorithm for the ballbot," *IEEE Access*, vol. 7, pp. 38314–38321, 2019.

[20] S. M. Saidi, R. Mellah, A. Fekik, and A. T. Azar, "Real-time fuzzy-PID for mobile robot control and vision-based obstacle avoidance," *Int. J. Service Sci., Manage., Eng., Technol.*, vol. 13, no. 1, pp. 1–32, Sep. 2022.

[21] D. B. Pham and S.-G. Lee, "Hierarchical sliding mode control for a two-dimensional ball segway that is a class of a second-order underactuated system," *J. Vibrat. Control*, vol. 25, no. 1, pp. 72–83, Jan. 2019.

[22] Z. Anjum, H. Zhou, S. Ahmed, and Y. Guo, "Fixed time sliding mode control for disturbed robotic manipulator," *J. Vibrat. Control*, vol. 30, nos. 7–8, pp. 1580–1593, Apr. 2024.

[23] M. Neunert, F. Farshidian, and J. Buchli, "Adaptive real-time nonlinear model predictive motion control," in *Proc. IROS Workshop Mach. Learn. Planning Control Robot Motion*, 2014.

[24] V.-T. Do, S.-G. Lee, and J.-H. Kim, "Robust integral backstepping hierarchical sliding mode controller for a ballbot system," *Mech. Syst. Signal Process.*, vol. 144, Oct. 2020, Art. no. 106866.

[25] D. B. Pham, J. Kim, and S.-G. Lee, "Combined control with sliding mode and partial feedback linearization for a spatial ridable ballbot," *Mech. Syst. Signal Process.*, vol. 128, pp. 531–550, Aug. 2019.

[26] Y. Zhou, J. Lin, S. Wang, and C. Zhang, "Learning ball-balancing robot through deep reinforcement learning," in *Proc. Int. Conf. Comput., Control Robot. (ICCCR)*, 2022, pp. 1–8.

[27] J. G. Pérez-Juárez, J. R. García-Martínez, A. M. Santiago, E. E. Cruz-Miguel, L. F. Olmedo-García, O. A. Barra-Vázquez, and M. A. Rojas-Hernández, "Kinematic fuzzy logic-based controller for trajectory tracking of wheeled mobile robots in virtual environments," *Symmetry*, vol. 17, no. 2, p. 301, Feb. 2025. [Online]. Available: https://www.mdpi.com/2073-8994/17/2/301

[28] W. Xue, B. Zhou, F. Chen, H. Taghavifar, A. Mohammadzadeh, and E. Ghaderpour, "A constrained fuzzy control for robotic systems," *IEEE Access*, vol. 12, pp. 7298–7309, 2024.

[29] P. Gui, L. Tang, and S. Mukhopadhyay, "MEMS based IMU for tilting measurement: Comparison of complementary and Kalman filter based data fusion," in *Proc. IEEE 10th Conf. Ind. Electron. Appl. (ICIEA)*, Auckland, New Zealand, Jun. 2015, pp. 2004–2009.

[30] J. Blonk, "Modeling and control of a ball-balancing robot," M.S. thesis, Univ. Twente, 2014.

[31] J. R. Raol, G. Girija, and J. Singh, *Modelling and Parameter Estimation of Dynamic Systems*, vol. 65. Edison, NJ, USA: IET, 2004.

[32] J. Wu, J. Wang, and Z. You, "An overview of dynamic parameter identification of robots," *Robot. Comput.-Integrated Manuf.*, vol. 26, no. 5, pp. 414–419, Oct. 2010.

[33] L. Liu, Z. Long, A. T. Azar, Q. Zhu, I. K. Ibraheem, and A. J. Humaidi, "Least square algorithm based on bias compensated principle for parameter estimation of canonical state space model," *Meas. Control*, vol. 55, nos. 5–6, pp. 330–339, May 2022.

[34] P. Van den Hof, "Closed-loop issues in system identification," *IFAC Proc. Volumes*, vol. 30, no. 11, pp. 1547–1560, Jul. 1997.

[35] W. D. Widanage, A. Barai, G. H. Chouchelamane, K. Uddin, A. McGordon, J. Marco, and P. Jennings, "Design and use of multisine signals for Li-ion battery equivalent circuit modelling. Part 1: Signal design," *J. Power Sources*, vol. 324, pp. 70–78, Aug. 2016.