Gazi Üniversitesi
Fen Bilimleri Dergisi
PART C: TASARIM VE TEKNOLOJİ

Gazi University
Journal of Science
PART C: DESIGN AND TECHNOLOGY

GU J Sci, Part C, 13(4): 1680-1698 (2025)

Simulation of Evolutionary Reinforcement Learning-Based Self-Balancing Throwable One-Legged Robot with a Reaction Wheel

Halit HÜLAKO[1*]

[1]*Hakkari University, Engineering Faculty, Mechanical Engineering Department, Hakkari, Turkey*

Article Info

Research article
Received: 07/03/2025
Revision: 21/08/2025
Accepted: 23/08/2025

Keywords

Self balance
ERL Learning
Simscape Multibody M.

Makale Bilgisi

Araştırma makalesi
Başvuru: 07/03/2025
Düzeltme: 21/08/2025
Kabul: 23/08/2025

Anahtar Kelimeler

Kendi kendine dengeleme
ERL Öğrenme
Simscape Multibody M.

Graphical/Tabular Abstract (Grafik Özet)

In this paper, the One-Legged robot is designed to stabilize itself and stand upright at the desired location after being thrown from a different heights. The 5-DOF planar underactuated main body is driven by Reaction wheels, and adaptive Cartesian impedance control has been implemented to effectively manage hard impacts. Evolutionary Reinforcement Learning based AI Agent has been used to adapt to different launch conditions, such as varying speed and altitude. / Bu makalede, Tek Bacaklı robot farklı yüksekliklerden fırlatıldıktan sonra kendini dengeleyip istenilen konumda dik duracak şekilde tasarlanmıştır. 5 serbestlik dereceli düzlemsel ve eksik tahrikli ana gövde, reaksiyon tekerlekleriyle sürülmekte olup sert darbelere karşı etkin bir şekilde başa çıkmak için adaptif kartezyen empedans kontrolü uygulanmıştır. Farklı hız ve irtifa gibi fırlatma koşullarına uyum sağlamak için Evrimsel Pekiştirmeli Öğrenme tabanlı bir Yapay Zekâ Ajanı kullanılmıştır.

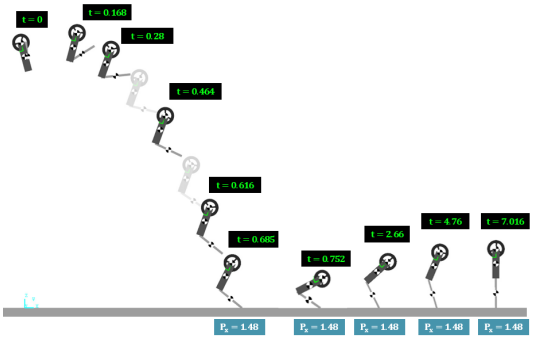

Figure A*: Screenshots of the robot's launch and upright standing for Test 1 were taken using MSM. /**Şekil A:** Test 1 için robotun fırlatılması ve dik durma anına ait ekran görüntüleri MSM kullanılarak alınmıştır.*

Highlights (Önemli noktalar)

- *Tek bacaklı robot, fırlatıldıktan sonra Reaksiyon tekerleği ve uyarlamalı Kartezyen empedans kontrolü sayesinde dengede kalabilmektedir. / The one-legged robot can maintain balance after being thrown using the Reaction Wheel and adaptive Cartesian impedance control.*
- *Evrimsel Pekiştirmeli Öğrenme (ERL) ile robot, farklı hız ve yüksekliklerdeki fırlatma testleri üzerinden kendi kendine dengelemeyi öğrenebilmektedir. / With Evolutionary Reinforcement Learning (ERL), the robot can learn self-balancing through launch tests at different speeds and heights.)*

Aim (Amaç): *It addresses the control of a one-legged robot that can learn to balance itself after being thrown. / Fırlatıldıktan sonra dengesini sağlamayı öğrenebilen tek bacaklı bir robotun kontrolünü ele alıyor.*

Originality (Özgünlük): *An agent was created using the ERL method, the robot was subjected to training to achieve self-balance, and learning was accomplished through this process. / ERL yöntemi kullanılarak bir ajan oluşturulmuştur, robotun kendi kendini dengeleyebilmesi için eğitime tabi tutulmuş ve bu süreçte öğrenme sağlanmıştır.*

Results (Bulgular): *Atıştan sonra robot dengelenmeyi sağlamış ve π/2 konumunda sabitlenmiştir. / After the launch, the robot achieved balance and stabilized at π/2.*

Conclusion (Sonuç): *After training and learning the control process, the robot has successfully maintained its balance in the MSM environment. While the planar motion has been effectively controlled in the simulation, there will be numerous parameters to consider when transitioning to a real-world application. / Robot, eğitim ve kontrol sürecini öğrendikten sonra, MSM ortamında dengesini başarıyla korumuştur. Düzlemsel hareket simülasyonda etkili bir şekilde kontrol edilmiş olsa da, gerçek dünya uygulamasına geçişte dikkate alınması gereken birçok parametre olacaktır.*

*Corresponding author, e-mail: halithulako@hakkari.edu.tr
DOI: 10.29109/gujsc.1653403

GU J Sci, Part C, 13(4): 1680-1698 (2025)

Gazi Üniversitesi
Fen Bilimleri Dergisi
PART C: TASARIM VE TEKNOLOJİ

Gazi University
Journal of Science
PART C: DESIGN AND TECHNOLOGY

http://dergipark.gov.tr/gujsc

Simulation of Evolutionary Reinforcement Learning-Based Self-Balancing Throwable One-Legged Robot with a Reaction Wheel

Halit HÜLAKO[1*]

[1]*Hakkari University, Engineering Faculty, Mechanical Engineering Department, Hakkari, Turkey*

Article Info

Research article
Received: 07/03/2025
Revision: 21/08/2025
Accepted: 23/08/2025

Keywords

Self balance
ERL Learning
Simscape Multibody M.

Abstract

In this paper, the One-Legged robot is designed to stabilize itself and stand upright at the desired location after being thrown from a different heights. The 5-DOF planar underactuated main body is driven by Reaction wheels, and adaptive Cartesian impedance control has been implemented to effectively manage hard impacts. Evolutionary Reinforcement Learning based AI Agent has been used to adapt to different launch conditions, such as varying speed and altitude. The learning process was performed as online learning within the Matlab simulation environment, which models the system dynamics of the robot. The graphical results of the simulation confirm that, with the assistance of the AI agent, the dynamic robot has successfully maintained its stability without tipping over after the launch and has been able to make the desired correction.

Evrimsel Pekiştirmeli Öğrenme Tabanlı Kendini Dengeleyebilen, Fırlatılabilir Reaksiyon Tekerli Tek Bacaklı Robotun Simülasyonu

Makale Bilgisi

Araştırma makalesi
Başvuru: 07/03/2025
Düzeltme: 21/08/2025
Kabul: 23/08/2025

Anahtar Kelimeler

Kendi kendine dengeleme
ERL Öğrenme
Simscape Multibody M.

Öz

Bu makalede, Tek Bacaklı robot farklı yüksekliklerden fırlatıldıktan sonra kendini dengeleyip istenilen konumda dik duracak şekilde tasarlanmıştır. 5 serbestlik dereceli düzlemsel ve eksik tahrikli ana gövde, reaksiyon tekerlekleriyle sürülmekte olup sert darbelere karşı etkin bir şekilde başa çıkmak için adaptif kartezyen empedans kontrolü uygulanmıştır. Farklı hız ve irtifa gibi fırlatma koşullarına uyum sağlamak için Evrimsel Pekiştirmeli Öğrenme tabanlı bir Yapay Zekâ Ajanı kullanılmıştır. Öğrenme süreci, robotun sistem dinamiklerini modelleyen Matlab simülasyon ortamında eşzamanlı olarak gerçekleştirilmiştir. Simülasyonun grafiksel sonuçları, Yapay Zekâ ajanının yardımıyla dinamik robotun fırlatma sonrası devrilmeden dengesini başarıyla koruduğunu ve istenilen düzeltmeyi yapabildiğini doğrulamaktadır.

1. INTRODUCTION (GİRİŞ)

In the modern world, people expect the availability of machines and devices that can be quickly used or set up without much effort. This is primarily driven by the need for simplification and time-saving, particularly with the advancement of technology. This can be an example of how a tripod or a monopod can be effortlessly prepared and deployed by simply throwing it, making it immediately ready for use.

In static stability criteria, if the projection of the robot's center of mass falls within the support polygon, the robot can maintain a balanced stance without tipping. However, in dynamic systems where no support polygon is defined, the robot must continuously move to maintain balance. In a dynamic system, especially for one-legged robot, where the leg contacts the ground at a single point, the robot must continuously perform hopping motions to maintain stability. When the robot is stationary, it tends to tip over. A planar robot has been developed to achieve and control balance after being thrown to a desired location. The robot is composed of a body and a single leg, and is capable of performing underactuated main body movements through the Reaction Wheel placed on it. The robot controls the impact effect using the Reaction Wheel and the leg actuator to maintain balance.

In general, Reaction wheels are an important method preferred in axis sets where direct connection and actuation are not possible. While robots in continuous contact with their environment can control their orientation, Reaction wheels are the preferred method in places with insufficient

*Corresponding author, e-mail: halithulako@hakkari.edu.tr
DOI: 10.29109/gujsc.1653403

actuation. Today, reaction wheels are widely used in space technology to control the orientation of free-floating satellites [1], [2]. Yang used magnetic torque coils, which are also used as actuators in satellites, to manage the amount of momentum that reaction wheels need to apply (saturation control) [3]. Zhang et al. attempted to control the roll angle of a robotic fish using a reaction wheel [4]. Gajamohan et al. managed to control a three-axis cube (Cubli), which can change direction, jump and stand at precise angles on its corners, with reaction wheels [5], [6]. Due to their common use in the control of underactuated systems, reaction wheels can be used to achieve balance control of single-wheeled mobile vehicles, referred to in the literature as unicycles [7], [8]. Additionally, Trentin et al. modelled and performed balance control of an inverted pendulum system using reaction wheels on both sides [9].

Kim et al. introduced a compact and lightweight Air Reaction Wheel (ARW) for small-scale legged jumping robots. The ARW generates high torque through air push and motor angular acceleration. Simulations and experiments validate its superior torque performance and stability, making it ideal for maneuvering rough terrains[10]. Zabihi and Alasty introduced a novel one-legged handspringing robot capable of hopping with both springy sides. This robot, featuring a single rotary actuator and two reaction wheels, considers slipping phases and demonstrates superior obstacle-clearing abilities compared to traditional hopping robots. The reaction wheels contribute to the dynamic stability by providing necessary torque to control the robot's orientation during flight and stance phases[11]. Haoran et al. presented Marsbot, a monopod robot designed for stable and precise 3D jumping. Utilizing the SLIP dynamics model for take-off control and the RWP model for attitude control, Marsbot employs three inertial tails as reaction wheels to achieve dynamic balance and accurate 3D positioning. These reaction wheels provide necessary torque adjustments to maintain stability and control during flight. Simulations confirm Marsbot's ability for continuous jumping and stable perching, validating the control algorithms and models[12]. Anzai et al. developed the MH-1, a monopod robot equipped with a reaction wheel to achieve hopping and posture stabilization. The reaction wheel provides necessary torque for upright posture control and recovery from falls. Experimental results demonstrate the robot's ability to hop in a constrained vertical direction and stabilize its posture using the reaction wheel. This study highlights the effectiveness of integrating reaction wheels in legged robots for dynamic balance and control [13]. In some studies of legged robots, dynamic models with reaction wheels were created to achieve the desired orientations on the underactuated main body and support balanced walking [14]. Roscia et al. presented an Orientation Control System (OCS) for quadruped robots, designed to improve aerial maneuvers during jumps. The system utilizes two rotating and actuated flywheels to control the robot's orientation by adjusting its angular momentum, addressing challenges in maintaining stability during the flight phase. Simulations on the Solo12 robot demonstrate the OCS's effectiveness in controlling roll and pitch angles, rejecting disturbances, and stabilizing post-landing. This compact OCS enhances the robot's maneuverability in complex environments [15].

Zhu et al. introduced TERL, which combines Evolutionary Algorithms (EA) and Reinforcement Learning (RL). TERL enhances exploration through RL and Particle Swarm Optimization (PSO), and focuses on the best individual for refinement. It outperforms existing RL and ERL methods in continuous control tasks [16]. Deng et al. presented QLJAYA, which integrates Q-learning and gradient search into the JAYA algorithm. QLJAYA improves convergence, local exploitation, and rotational invariance. Experiments show it outperforms standard JAYA and other metaheuristics [17]. The same group also introduced the Snow Ablation Optimizer (SAO), inspired by snow sublimation and melting. SAO balances exploration and exploitation, outperforming other metaheuristics on CEC2017 and CEC2020 benchmarks [18]. The use of Evolutionary Reinforcement Learning (ERL) in this work is motivated by its ability to optimize complex, high-dimensional problems where traditional methods struggle. ERL provides a robust framework that combines the exploratory power of evolutionary algorithms with the reinforcement learning. The decision to use a combination of genetic algorithm (GA) and artificial neural networks (ANN) is based on their complementary strengths. GA is effective in exploring the solution space and avoiding local optima, while ANN excels in modeling non-linear relationships and learning from large datasets. While many evolutionary algorithms are available, the combination of GA and ANN was chosen due to its demonstrated effectiveness in similar robotic control tasks.

The motion equations for dynamic systems are usually derived using the Lagrange or Newton-Euler methods. One approach to obtaining the dynamic model of complex systems with high degrees of freedom is through the use of 'Multibody Dynamics' simulations. To obtain a system's dynamic model, one can use block structures with Matlab Simscape MultiBody. Additionally, Matlab Simscape MultiBody provides a physical

environment and visual animations of the system's operation.

This article is divided into several sections, including the modeling of the robot and the successful implementation of the throw and balance mission. The second section covers the kinematic and dynamic modeling of the robot, as well as the construction of the corresponding Multi-Body System Model (MSM) structure. The ground has been modeled using spring and damper elements. The third section gives details of the procedure followed in the implementation of the throw and balance mission, including the impedance control applied. Additionally, it presents the specifics of the learning algorithm applied to maintain balance without tipping during the collision with the ground. Section four provides numerical simulations and discusses graphical results. In Chapter 5, conclusion and various recommendations for researchers are presented.

2. KINEMATIC AND DYNAMIC MODELING
(KİNEMATİK VE DİNAMİK MODELLEME)

Figure 1 shows the placements of the kinematic and dynamic representations of the Reaction Wheel Robot model. The robot is composed of three rigid bodies: the main body, the reaction wheel, and a leg with one degree of freedom (DOF), denoted by $i \in \{1,2,3\}$. The reaction wheel and the leg are connected to the two ends of the main body using revolute joints. Figure 1 shows that in the model, point A_0 represents the connection point of the main body and the reaction wheel, while A_1 represents the connection point of the main body and the leg. The planar dynamic robot model has a total of 5 DOF and moves freely in space. The constructed robot model uses the generalized coordinates $q \in \Re^{n+3}$, where the term 'n' represents the actuated joints. In its general form, the nonlinear equation of motion is as follows:

$$M(q)\ddot{q} + h(q,\dot{q}) = S^T\tau + \mathrm{J}^T f_c \quad (1)$$

The inertia matrix is represented by $M \in \Re^{(n+3)x(n+3)}$, and $h \in \Re^{n+3}$ includes gravity, Coriolis, and centripetal terms. The expression J provides the Jacobian matrix obtained for the contact point, while f_c represents the contact forces that arise when the robot's leg is in contact with the ground. The torque values in the actuated joints, excluding the underactuated base, are represented by $\tau \in \Re^n$. The selection matrix S^T is a boolean matrix used to choose n torque values.

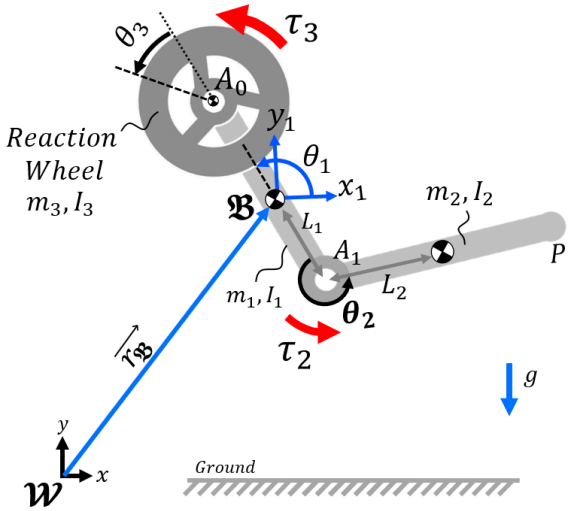

Figure 1. Kinematic and dynamic representations of the robot model. (Robot modelinin kinematik ve dinamik gösterimleri)

L_1 represents the distance from the center of the main body to joint A_1, while L_2 represents the distance from the center of the leg to end of the leg. Point P represents the point located on the lower part of the leg that makes contact with the ground. $\mathcal{W}(x,y)$ is defined as the inertial frame, while the floating base axis $\mathfrak{B}(x_1,y_1)$ positioned at the center of mass of the robot's main body. The position of the floating base $\mathfrak{B}$ with respect to the fixed reference frame is denoted by the expression $r_{\mathfrak{B}} \in \Re^2$. The angular displacement of the base axis is represented by θ_1. The angular displacements of the A_1 and A_0 joints are represented by $q \in \Re^n = (\theta_2, \theta_3)$, respectively. θ_1 is measured from the horizontal axis, while θ_2 and θ_3 are measured relative to θ_1. The mass and moments of inertia for centroids are respectively represented by $m_i \in \Re^3$ and $I_i \in \Re^3$. (x_1, y_1, θ_1) represents the underactuated position and orientation of the floating base. To indirectly control the

underactuated base, it is necessary to apply torques $\tau \epsilon \Re^n$ to the A_0 and A_1 joints.

The dynamic robot model shown in Figure 1 is obtained using MSM to produce the same results as the equations of motion given in Appendix. The MSM structure is modeled using Simulink and Simscape block diagrams as shown in Figure 2.

Solid objects can be created using solid modelling software, such as CAD programs or directly added from the ready library of MSM. The MSM simulation program is primarily composed of interconnected block diagrams that represent joint types, solid object models, and Rigid Transform blocks used to define axes to desired points. For the 3-DOF body "$\mathfrak{B}$" located at the center of the robot's body uses a planar joint named "Base".

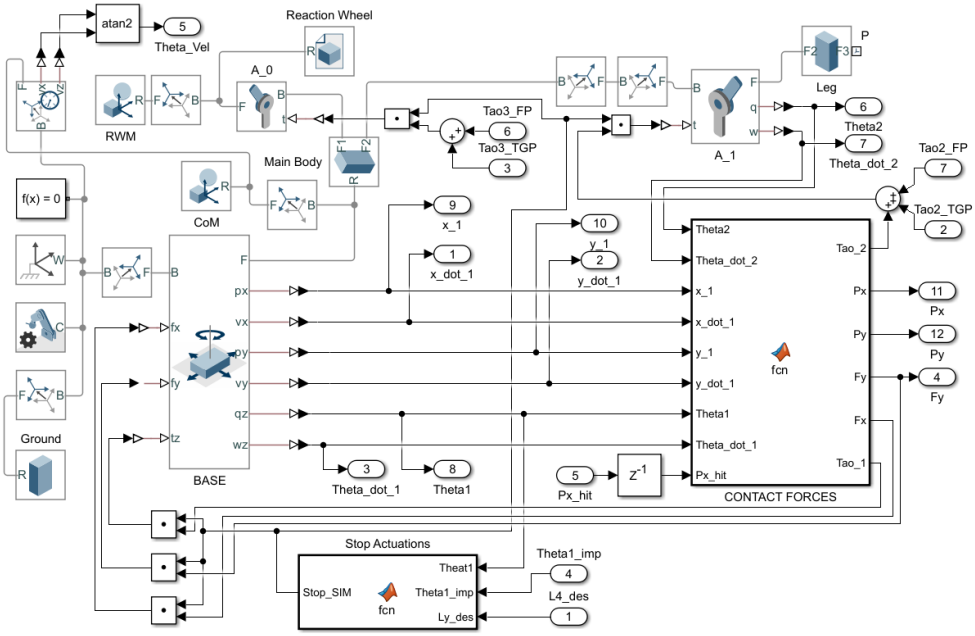

Figure 2. The block diagram illustrates the MSM dynamic model of the robot. (Blok diyagramı, robotun MSM dinamik modelini göstermektedir)

The upper end of main body A_0 is connected to the **Reaction Wheel** using a single DOF revolute joint, while the **Leg** block is connected to the lower end of the main body A_1 using a single degree-of-freedom revolute joint. In MSM, a structural damping amount of 0.005 $Nm/\frac{deg}{s}$ defined for the revolute joint **A_1**. The dynamic parameters for these components are entered in the **Main Body**, **Reaction Wheel**, and **Leg** blocks. The **Contact Forces** function block calculates the contact forces that arise when the robot's leg comes into contact with the ground. The resulting torque control signals are then sent to the **A_0** and **A_1** joint blocks, which are connected to the reaction wheel and the leg respectively. Once the main structure providing the dynamics is completed and the system analysis is initiated, animation of the mechanic model operating in the physical environment are provided. In MSM, it is possible to create a feedback system within the same environment by adding controller functions in addition to the main structure that gives the robot's dynamics. Torque signals obtained from all phases are collected and fed into the inputs of the **A_0** and **A_1** joints. The **Stop Actuation (SA)** block was added to terminate unsuccessful balancing attempts during real-time (in simulation) training.

2.1 Jacobian and Contact Modeling (Jakobyan ve Kontak Modelleme)

It is assumed that the contact surface of the robot leg with the ground is a point. The position of the point $P = [P_x\ P_y]^T \epsilon \Re^2$ relative to the inertial axis, P_x and P_y values are given in 2.a and 2.b, respectively.

$$P_x = x_1 - L_1\, cos(\theta_1) + 2L_2\, cos(\theta_1 + \theta_2) \quad (2.a)$$

$$P_y = y_1 - L_1\, sin(\theta_1) + 2L_2\, sin(\theta_1 + \theta_2) \quad (2.b)$$

The Jacobian expression J_P, which is obtained for the position vector P, is given below;

$$J_P = \begin{pmatrix} 1 & 0 & L_1\, sin(\theta_1) - 2L_2\, sin(\theta_1 + \theta_2) & -2L_2\, sin(\theta_1 + \theta_2) \\ 0 & 1 & -L_1\, cos(\theta_1) + 2L_2\, cos(\theta_1 + \theta_2) & 2L_2\, cos(\theta_1 + \theta_2) \end{pmatrix} \quad (3)$$

The vertical and horizontal contact forces acting at point *P* are given in equation (4).

$$f_c = \begin{cases} \left[k(c^x(P_x^{hit} - P_x) - b_c^x \dot{P_x}) \ \left(-k_c^y P_y - b_c^y \dot{P_y}\right)\right] & \text{if} \quad P_y < 0 \\ [0\ 0] & \text{if} \quad P_y \geq 0 \end{cases} \quad (4)$$

The contact forces occurring at point P are represented by $f_c = \left[f_c^x \ f_c^y\right] \epsilon \, \Re^2$. The mechanics of the ground in the horizontal and vertical directions are modelled as damping and spring elements. The position of the foot tip on the x-axis relative to the inertial axis at the moment of contact with the ground is denoted by P_x^{hit}. The stiffness of the ground is represented by the term $k_c \epsilon \, \Re^2 = \left(k_c^x \ k_c^y\right)$, while the damping coefficient of the ground is represented by $b_c \epsilon \, \Re^2 = \left(b_c^x, b_c^y\right)$. The values of k_c and b_c were determined through drop tests. $\dot{P} \epsilon \, \Re^2$, the first derivative of the toe point *P* with respect to time, is given in (5).

$$\dot{P} = \mathrm{J}_P[x_1 \quad y_1 \quad \theta_1 \quad \theta_2]^T \quad (5)$$

If point *P* touchs the ground, it is assumed that there is no slip and $\dot{P}(t) = 0$.

3. ONE-LEGGED ROBOT BALANCE CONTROL PROCEDURE (TEK-BACAKLI ROBOT DENGE KONTROL PROSEDÜRÜ)

Balance control procedure is divided into three main phases. The first phase, known as the 'Flight Phase' covers the period from when the robot is launched until it makes contact with the ground. During this stage, the robot is in mid-air, guided towards its intended destination and preparing for landing. The second phase is the 'Transition Phase'. In this phase, we focus on the moment the robot touches the ground. During this phase, an online real-time learning in simulation structure is developed to prevent the robot from tipping over upon impact and to maintain a balanced posture. The structure is designed to effectively absorb the impact forces of a hard landing while simultaneously learning to maintain the robot's equilibrium, ensuring that it remains upright without falling. This stage is considered the most crucial component of the balance control procedure. The final phase is the 'Ground Phase'. During this stage, the robot completes its full upright position while maintaining control. The robot then proceeds to its final location while the complete control structure of the learning and phases created by Matlab Simulink, including the robot and its environment, is shown in Figure 3. This integrated structure manages an effective control and learning process during the flight, transition, and grounding stages.

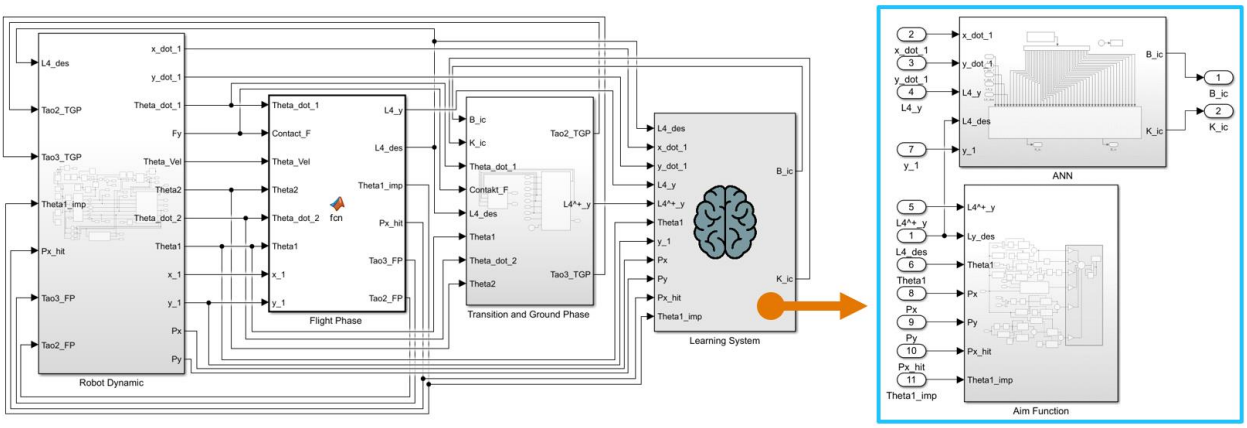

Figure 3. Main MS and MSM control architecture for Robot. (Robot için temel MS ve MSM kontrol mimarisi)

Figure 3 shows the **Robot Dynamic (RD)** block, which models the robot and the environment. The blocks for the **Flight Phase (FP)**, **Transition and Ground Phase (TGP)** are presented in that order and have been thoroughly discussed. The **Learning System (LS)** performs the learning and determines the control coefficients. The **LS** subblock contains block structures for the Neural Network and aim function, referred to as **ANN** and **Aim Function (AF)** respectively. State observations taken from the **RD** block feed the other block structures. While, control actions obtained from the **FP** and **TGP** blocks enter the **RD** block.

3.1. Flight Phase (Uçuş Fazı)

The Flight Phase covers the period during which the robot launches and travels towards the targeted landing area. During this stage, Flight phase controllers track the desired references and adjust the robot's orientation during the flight. It is proposed that, when the robot leg makes contact with the ground, the direction of the velocity vector aligns in parallel between the total center of mass and point *P*. Thus, the direction of the planar vector

belonging to the total center of mass of the robot is desired to pass through point P during flight. It will be aligned in the direction of the $\vec{L}_4$ vector, defined between point P and CoM, which is controlled by impedance control, starting when the robot touches the ground and throughout the transition phase. In other words, this desired situation is achieved by minimizing θ_{err}. P' is the desired reference endpoint of the robot leg. As seen in Figure 4, the angular difference between P and P' was modeled and controlled as a torsional spring and damper, with the balance point being P'.

Equation (6) provides the δ value obtained through geometric methods for generating the reference value θ_2^{ref}. During the flight phase, after launching the robot, a variable reference orbit θ_2^{ref} is produced depending on θ_{vel}, which is the angle of the velocity vector of the total center of mass. τ_2^{FP} controls the angle θ_2 to follow the variable reference trajectory during the flight phase. The equations are given in Equations 7 and 8.a, 8.b, respectively.

$$\delta = \sin^{-1}\left(\frac{L_G \sin(\theta_1 - \theta_{vel})}{2L_2}\right) \quad (6)$$

$$\theta_2^{ref} = (\delta - \theta_1 + \theta_{vel}) + 2\pi \quad (7)$$

$$\tau_2^{FP} = \left(\theta_2^{ref} - \theta_2\right)K_2^{fl} - \dot{\theta}_2 B_2^{fl}, \quad (8.a)$$

$$\tau_3^{FP} = (\theta_1^{FP} - \theta_1)K_3^{fl} - \dot{\theta}_1 B_3^{fl} \quad (8.b)$$

The calculation of the total center of mass excludes the robot leg due to its significantly lighter weight compared to other parts. The distance from the total center of mass to A_1 is denoted as L_G. The velocity vector of the total center of mass is represented by $\vec{V}_G$, and its angle is denoted as θ_{vel}. The transform sensor is used in the **RD** block to obtain the θ_{vel} value of the CoG. K_2^{fl} and B_2^{fl} represent the virtual spring and damper values used to control θ_2 during the flight phase. A control torque is applied to the Reaction Wheel to maintain the θ_1 angle of the Main Axis at the desired value throughout the flight duration. Therefore, equation (8) applies torsional impedance control between the desired reference angle θ_1^{FP} and θ_1. Applying torque to the reaction wheel creates a reverse moment that affects the orientation of the main axis, specifically θ_1.

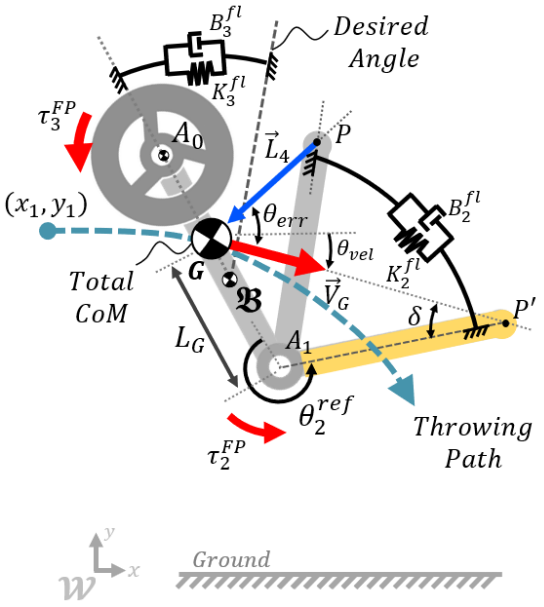

Figure 4. Flight Phase control. (Uçuş Fazı Kontrolü)

τ_3^{FP} is the control torque used to control the angle θ_1, K_3^{fl} and B_3^{fl} are the virtual spring damping coefficients used to control θ_1 in the flight phase, respectively. Virtual control coefficients are optimized by trial and error technique. τ_3^{FP} controls the angle θ_1, while K_3^{fl} and B_3^{fl} are virtual spring damping coefficients used for controlling θ_1 during the flight phase. The virtual control coefficients have been optimized using a trial and error technique.

3.2. Transition and Ground Phase (Geçiş ve Yer Fazı)

Adaptive impedance control has been implemented to enable the robot to absorb the effects of hard impacts, prevent tipping over, and recover balance. This approach offers a versatile method for controlling the robot's response to external forces, ensuring stability, and adjusting oscillation characteristics as needed. Figure 5 shows that the length and angle of vector L_4 are controlled by springs and dampers defined on the horizontal and vertical axes. These forces act on the robot, allowing it to imitate spring damping movement [19]. The stiffness value and damping coefficient of the spring can be adjusted to achieve the desired type of oscillation. The vector loop closure equation for the $L_4 \epsilon\, \Re^2$ vector shown in Figure 5 can be written with complex numbers in exponential form as in (9) and this yield two explicit equation. The length of the vector and its first derivative are provided in (10.a and 10.b) and (11.a and 11.b), respectively, while the angle θ_4 is given in equation (12).

$$\overrightarrow{GA_1} + \overrightarrow{A_1P} + \overrightarrow{PG} = 0 \quad (9)$$

$$\overrightarrow{PG} = -L_G e^{i(\theta_1+\pi)} - 2L_2 e^{i(\theta_1+\theta_2)} \quad (10)$$

$$L_4^x(\theta_1,\theta_2) = L_G\cos(\theta_1) - 2L_2\cos(\theta_1+\theta_2) \quad (10.a)$$

$$L_4^y(\theta_1,\theta_2) = L_G\, sin(\theta_1) - 2L_2\, sin(\theta_1+\theta_2) \quad (10.b)$$

$$V_4 = iL_G\dot{\theta}_1 e^{i(\theta_1)} + i2L_2(\dot{\theta}_1+\dot{\theta}_2)e^{i(\theta_1+\theta_2)} \quad (11)$$

$$\dot{L_4^y} = L_G\dot{\theta}_1\cos(\theta_1) - 2L_2(\dot{\theta}_1+\dot{\theta}_2)\cos(\theta_1+\theta_2) \quad (11.a)$$

$$\dot{L_4^x} = -L_G\dot{\theta}_1\, sin(\theta_1) + 2L_2(\dot{\theta}_1+\dot{\theta}_2)\, sin(\theta_1+\theta_2) \quad (11.b)$$

$$\theta_4 = \cos^{-1}\left(\frac{L_G\cos(\theta_1) - 2L_2\cos(\theta_1+\theta_2)}{L_4}\right) \quad (12)$$

L_4^x and L_4^y represent the components on the horizontal and vertical axes, respectively. During the transition phase, the objective is to stabilize the angle θ_4 at $\pi/2$ radians and maintain the length of L_4 at a predetermined value during impact. Control is achieved by learning the system through experimentation using a learning algorithm. Once a stable posture is achieved in the transition phase, the system moves to the ground phase and attempts to maintain the angle θ_4 at the same value. The L_4 value is controlled to follow a trajectory until the robot is fully upright. Equations 13 and 14 provide the force components applied in the horizontal and vertical axes by impedance control.

$$F_{ic}^x = (L_4^{xeq} - L_4^x).K_{ic} - \dot{L_4^x}B_{ic} \quad (13)$$

$$F_{ic}^y = \{\ L_{err}^y K_{ic} - \dot{L_4^y}B_{ic} \quad if,\ L_4^y < (L_g + 2L_2\ ) \quad (14)$$

$$L_4^{des} = \{\ L_4^y(t_{imp})\ \ if,\ \ sign(f_c) > 0 \quad (15)$$

$$c_1^+ = \begin{cases} \sum_{\alpha=0}^{\frac{T}{t_{int}}-1} t_{int} & if, \quad (-\eta < L_4^x < \eta) \\ 0 & , \quad Otherwise \end{cases} \quad (16)$$

$$L_{err}^y = \begin{cases} \left(L_4^{des} + (T - t_{imp}).z\right) - L_4^y & if, \quad (-\eta < L_4^x < \eta)\ and\ \left(c_1^+ \geq t_{ref}\right) \\ \left(L_4^{des} - L_4^y\right) & , \quad Otherwise \end{cases} \quad (17)$$

The virtual impedance control coefficients, B_{ic} and K_{ic}, are obtained from artificial neural networks in both horizontal and vertical directions. These coefficients were trained and determined by ANN with random initial conditions, such as the robot's height from the ground and speeds in the horizontal and vertical axes. During the transition phase, the desired reference values for the x and y directions are L_4^{xeq} and L_4^{des}, respectively. L_4^{xeq} is determined in advance, and L_4^{des} is equal to the length L_4^y at the time of impact t_{imp} as stated in equation (15). The fixed solver interval is denoted as t_{int}. When the condition specified in (16) is met, control for the robot's straightening during the Ground phase is ensured with the time-dependent trajectory given in (17). T represents the total operational time until the process is completed, while z adjusts the rate of change in the velocity of the reference value. In the steady state, the goal is to achieve equilibrium within the $\pm\eta$ limits in the L_4^x direction, and at the same time, the c_1^+ value is expected to remain in equilibrium for the desired time t_{ref}. The representations of the Cartesian impedance control applied to the body center are shown in Figure 5. The resulting impedance forces are converted into torque action signals, which drive the reaction wheel and the second limb. These signals are derived from the equations presented in (18.a-b).

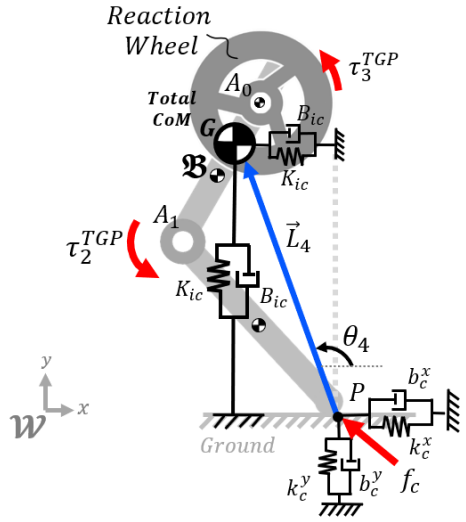

Figure 5. Cartesian Impedance Control used in Transition and Ground Phase. (Geçiş ve zemin fazlarında kullanılan Kartezyen empedans kontrolü.)

$$\tau_2^{TGP} = F_{ic}^x L_G\, sin(\theta_1) + F_{ic}^y L_G\, cos(\theta_1), \quad (18.a)$$

$$\tau_3^{TGP} = F_{ic}^x L_4^y + F_{ic}^y L_4^x \quad (18.b)$$

In the TG phases, τ_2^{TGP} and τ_3^{TGP} apply torques to A_1 and A_0, respectively. Figure 6 shows the sub-functions of **TGP**. The impedance control block receives the output variables K_{ic} and $B_{ic}$from the **ANN** block and converts them into control torques in the **Force-Torque Converter (FTC)** block.

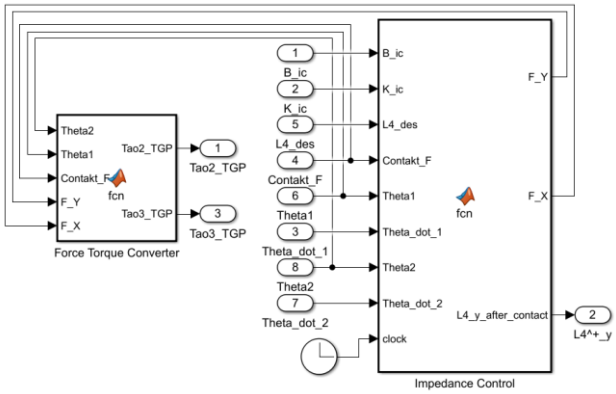

Figure 6. TGP subsystem block structure. (TGB Alt Bloğu)

3.3. Learning Balance Control During Impact
(Çarpma Anında Denge Kontrolünün Öğrenimi)

To effectively absorb the effects of high-energy collisions, the terms B_{ic} and K_{ic} are determined using Evolutionary Reinforcement Learning. In this system, the genetic algorithm and ANN work together to enable the agent's continuous learning and improvement of its performance. In model-free and continous system, the agent interacts with its environment to learn its task. The Artificial Neural Network or Policy generates actions based on the observation data obtained by the agent's interaction with the environment. The artificial neural network weight parameters are optimized by the genetic algorithm, evolving towards the best parameter set to increase the agent's performance. In contrast to Reinforcement Learning, where the agent tries to maximise the reward value, in our system the agent works towards minimising the Aim function, bringing it closer to zero. Furthermore, the agent is only provided with critical data rather than all observation data. The policy, acting as an actor, utilizes a neural network structure consisting of five inputs, five hidden layers, and two output cells. The neural network structure created for the Critic Network is disabled in this method. The neural network structure created for the actor has an input vector $Inp \in \Re^5$, a weight matrix for the first and second layers $w_{il} \in \Re^{5x5}$ and $w_{ol} \in \Re^{5x5}$, and a bias vector for the first and second layers $b_{il} \in \Re^5$ and $b_{ol} \in \Re^2$ as given Eq. 19.a-c

$$Inp = \begin{pmatrix} y_1 \\ \dot{y}_1 \\ \dot{x}_1 \\ L_4^y \\ L_4^{des} \end{pmatrix}, \quad w_{il} = \begin{bmatrix} w_1 & w_2 & \dots & w_5 \\ w_6 & w_7 & \dots & w_{10} \\ w_{11} & w_{12} & \dots & w_{15} \\ w_{16} & w_{17} & \dots & w_{20} \\ w_{21} & w_{22} & \dots & w_{25} \end{bmatrix}, \tag{19.a}$$

$$b_{il} = \begin{pmatrix} w_{36} \\ w_{37} \\ \vdots \\ w_{40} \end{pmatrix} \tag{19.b}$$

$$w_{ol} = \begin{bmatrix} w_{26} & w_{27} & \dots & w_{30} \\ w_{31} & w_{32} & \dots & w_{35} \end{bmatrix}, \; b_{ol} = \begin{pmatrix} w_{41} \\ w_{42} \end{pmatrix} \tag{19.c}$$

The ANN architecture used in this study (Eq. 19–20) was selected based on preliminary empirical trials, aiming to balance learning capacity with computational efficiency. The weight coefficients, represented by w_n where $n \in \{1,..,42\}$, are updated in real-time after each shot through the Genetic algorithm. The activation function used in the cells is 'tansig'. During the training phase, the ANN input receives the state variables $\dot{x}_1,\ y_1, \dot{y}_1, L_4^y$, and L_4^{des} of the dynamic model, and the output provides the necessary terms for B_{ic} and K_{ic} impedance control. The formulation is given Eq. 20.a and 20.b; Figure 7 shows the learning structure used for this purpose. The simulation uses the Matlab Genetic Algorithm Toolbox to determine the ANN weight coefficients. The objective function targeted for our system is given by equation (21). The term ψ is added during training to penalise falling due to collision effects in shooting trials. The term θ_1^h is equal to the value of $\theta_1(t_{imp})$ at the moment of the robot contact with the ground and L_4^{des}, L_4^{xeq} represents the vertical and horizontal clearance of the robot at that moment. Although the state variables given to the ANN input have changed, after the impact, the robot is required to balance L_4^y and L_4^x, around the value of L_4^{des}, L_4^{xeq} respectively.

$$f: \Re^5 \rightarrow \Re^2$$

$$f_{ol,k} = \sigma_{act}\left(\sum_{j=1}^{5} w_{ol,kj}.\sigma_{act}\left(\sum_{i=1}^{5} w_{il,ji}.Inp_i + b_{il,j}\right) + b_{ol,k}\right), \tag{20.a}$$

$$k = 1,2$$

Where;

$$\sigma_{act}(x) = 2.\left(\frac{1}{1+e^{-2.x}}\right) - 1 \tag{20.b}$$

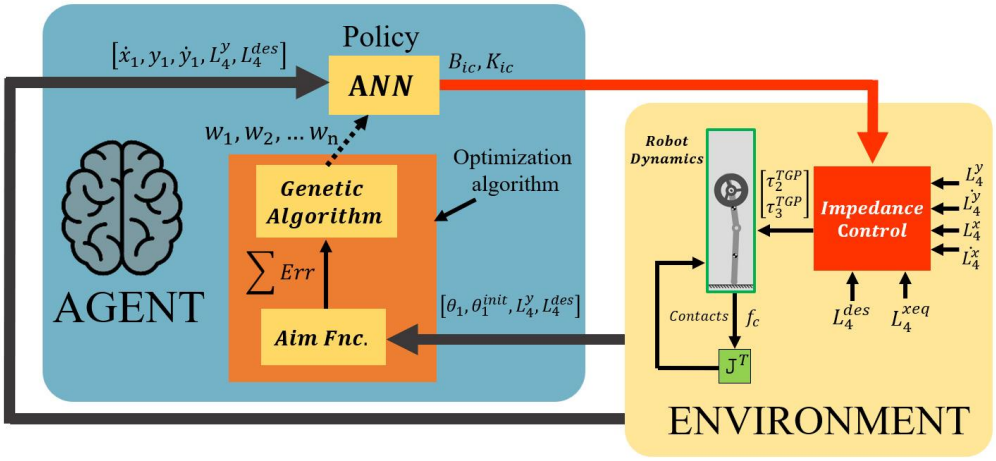

Figure 7. Real-time in simulation robot training cycle. (Simülasyonda gerçek zamanlı robot eğitim döngüsü)

Despite the large collision force, it is desired to allow a displacement in the horizontal direction up to a maximum of ξ and punishment starts when this limit is exceeded. The first term, $max\left(0, \left|L_4^{des} - \xi - L_4^y\right|^2\right)$, begins to receive high penalty points when it exceeds ξ, which is the maximum leg closing amount, due to the impact effect at the moment of contact with the ground. While the expression $\left|L_4^{des} - L_4^y\right|$ helps the robot returns to the balance point, $\left(L_4^{des} - L_4^y\right)^2$ minimises the difference between it and the target height. The terms $\left(P_x^{hit} - P_x\right)^2$ and $\lambda_6 {P_y}^2$ are added to prevent bouncing and displacement after collisions. λ_j $j \in \{1,..,6\}$ are the weighting factors and their values were determined by trial and error. The maximum control torque τ_k^{TGP}, $k \in \{2,3\}$ is limited to $\pm u_k^{max}$. Therefore, the conditions given in (23) have been applied. To ensure that the initial conditions are within controllable limits during robot launch, the initial values for θ_1 in the launch simulations are assumed to be in the range of $\frac{\pi}{3} < \theta_1 < \frac{13\pi}{18}$. The initial values for $\dot{y}_1$ are assumed to be in the range ${\dot{y}^{min}}_1 < \dot{y}_1 < {\dot{y}^{max}}_1 \frac{m}{s}$. For y_1 and $\dot{x}_1$ are $y_1^{min} < y_1 < y_1^{max}$ ${\dot{x}^{min}}_1 < \dot{x}_1 < {\dot{x}^{max}}_1 \frac{m}{s}$ respectively. In the initial state, link 2 is assumed to start almost parallel to the base, and θ_2 starts at angles close to zero radians.

$$\text{Err} = min \sum_{\vartheta=0}^{\frac{T_{train}}{t_{ln}}-1} \left(\lambda_1 . max\left(0, \left|L_4^{des} - \xi - L_4^y\right|^2\right) + \lambda_2 \left|L_4^{des} - L_4^y\right| + \lambda_3 \left(L_4^{des} - L_4^y\right)^2 + \lambda_4 \psi + \lambda_5 (P_x^{hit} - P_x)^2 + \lambda_6 {P_y}^2\right) \tag{21}$$

$$\psi = \begin{cases} (\theta_1 - \theta_1^h)^2, & if\ \left(\theta_1^h < \frac{\pi}{2}\right) and\ \left(-\frac{\pi}{5} > \theta_1 > \frac{\pi}{2}\right) and\ (L_4^{des} > 0) \\ & if\ \left(\theta_1^h > \frac{\pi}{2}\right) and\ \left(-\frac{5\pi}{6} > \theta_1 < \frac{\pi}{2}\right) and\ (L_4^{des} > 0) \end{cases} \tag{22}$$

$$s.t. \quad \begin{cases} y_1^{min} \le y_1 \le y_1^{max} \\ {\dot{y}^{min}}_1 \le \dot{y}_1 \le {\dot{y}^{max}}_1 \\ {\dot{x}^{min}}_1 \le \dot{x}_1 \le {\dot{x}^{max}}_1 \\ \tau_k = \begin{cases} u_k^{max} \frac{|\tau_k|}{\tau_k}, & \text{if } |\tau_k| > u_k^{max} \\ \tau_k, & \text{if } |\tau_k| \le u_k^{max} \end{cases} \end{cases} \tag{23}$$

The **AF** sub-block, modelled with Simulink blocks, is shown in Figure 8 (on the right). In each launch trial, the Genetic Algorithm, transfers new **ANN** weight coefficients to the MS environment for testing, as shown in Figure 8 (left). Throughout the simulation, the total error obtained from the **AF** is given to the Genetic Algorithm through the **Err** block. Therefore, as the objective function approaches zero during each simulation, the balance process is learned and the next simulation is tested with new coefficients to obtain better results. This real-time (in simulation) learning cycle continues until the number of iterations is reached or the desired performance range is achieved. The sampling time interval selected for training is entered in the **Err** block.

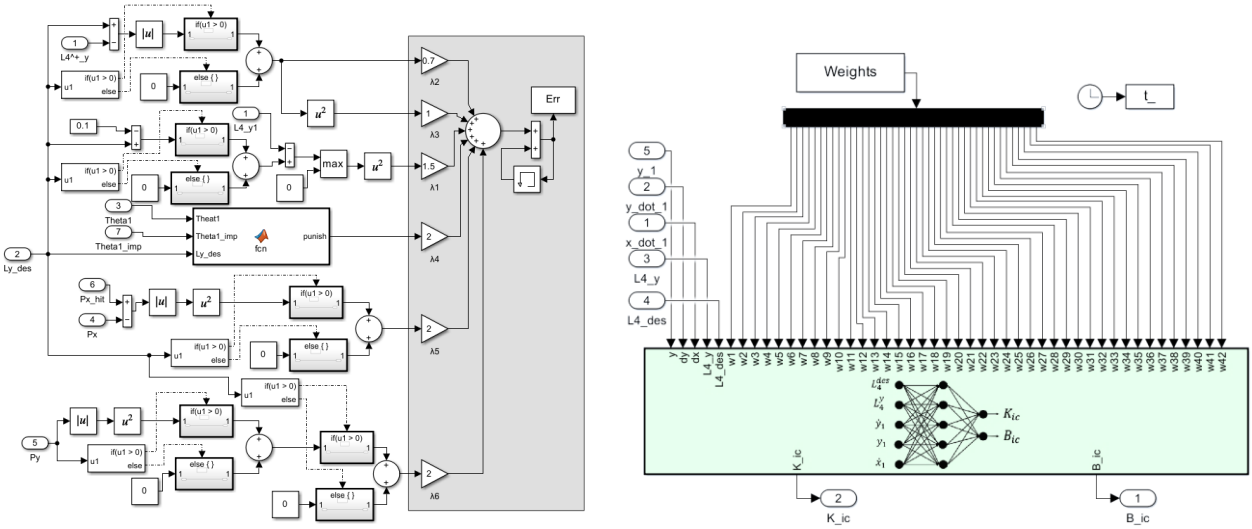

Figure 8. ANN (Left), **AF** (Right) subsystem block structures (YSA (Sol) ve AF (Sağ) alt sistem blok yapıları)

The genetic Algorithm is set up with 60 iterations and a population size of 10. The simulation initial conditions for y_1, $\dot{y}_1$, $\dot{x}_1$ are chosen randomly within the range specified in Eq. (23). The solver constant sampling time for the simulation is set to $t_{int} = 0.0001\ s$, and the solver is selected as "ode4 (Runge-Kutta) ". The sampling interval for training is chosen as $t_{ln} = 0.01\ s$ sec. The total simulation time for each training is $T_{train} = 1.40$ seconds.

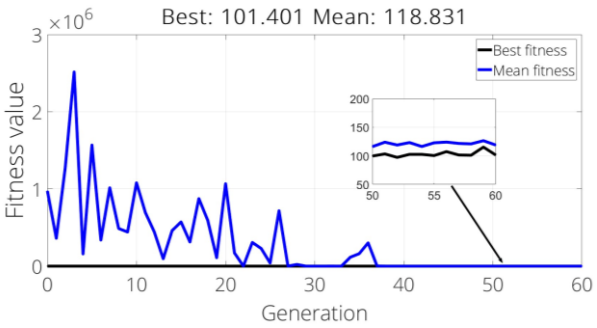

Figure 9. Learning State of Dynamic Systems During Iterations (Yinelemeler sırasında dinamik sistemlerin öğrenme durumu)

The training procedure was executed on a system equipped with an Intel Core i7 processor using Matlab 2022. A total of 600 launch trials were performed to complete the learning process. Figure 9 illustrates the Genetic Algorithm, listing the performance values while predicting the ANN parameters. Here, 'Best' refers to the best solution or the lowest objective function value found by the genetic algorithm at each iteration or generation. 'Mean' refers to the average value of all solutions in the population at each iteration or generation, which can be expressed as the average of the objective function or the fitness values. 'Fitness value' represents the fitness level of each individual or solution, typically measured based on an objective or fitness function. 'Iteration' can also be referred to as a generation. Each generation is a phase where individuals derived from the previous generation evolve and new individuals are created. Best and mean values are particularly important for evaluating the algorithm's performance, as they indicate the optimization progress and the quality of the solutions. It can be seen from the graph that the learning is completed after the 37th generation.

4. NUMERICAL SIMULATION (NÜMERİK SİMÜLASYON)

The results of the numerical simulations carried out using the Simulink model include all the control structures developed to test the launch and balance stages. The parameters required to control the robot and the initial values for the simulations performed are given in Tables 1 and 2, and these parameters are defined in the relevant blocks in the Simulink model.

Table 1. Parameters of Control and Robot dynamics (Kontrol ve robot dinamiklerinin parametreleri)

$Parameter$	$Value$	$Parameter$	$Value$	$Parameter$	$Value$
$L_1(m)$	0.2	λ_1	1.5	$y_1^{min}(m)$	1.2
$L_2(m)$	0.2	λ_2	0.7	$y_1^{max}(m)$	2.2
$L_G(m)$	0.142	λ_3	1	$\dot{y}^{min}{}_1(m/s)$	0
$m_1(kg)$	0.70	λ_4	2	$\dot{y}^{max}{}_1(m/s)$	2.5
$m_2(kg)$	$3x10^{-2}$	λ_5	2	$\dot{x}^{min}{}_1(m/s)$	0
$m_3(kg)$	$7x10^{-2}$	λ_6	2	$\dot{x}^{max}{}_1(m/s)$	2.7
$g(m/s^2)$	9.80665	$\xi(m)$	0.1	$K_2^{fl}(N/m)$	3.5
$I_1(kgm^2)$	$2.47917x10^{-3}$	$K_3^{fl}(N/m)$	1	$B_2^{fl}(N/m)$	$1x10^{-2}$
$I_2(kgm^2)$	$1.00563x10^{-4}$	$B_3^{fl}(N/m)$	0.1	$u_2^{max}(Nm)$	20
$I_3(kgm^2)$	$1.5931x10^{-4}$	$\eta(m)$	$2x10^{-2}$	$u_3^{max}(Nm)$	15
$k_c^x(N/m)$	10^3	$t_{ref}(sec.)$	2	$L_4^{xeq}(m)$	0
$b_c^x(N/ms^{-1})$	10^2	z	$1.5x10^{-2}$	$\theta_1^{FP}(rad)$	1.832
$k_c^y(N/m)$	10^5	$b_c^y(N/ms^{-1})$	$2x10^2$		

To ensure compatibility with a potential real-world application, the system parameters were selected within realistic and practical limits. As introduced in the context of a throwable tripod, the robot's total height was set to 0.4 meters, and the body weight was chosen as 700 grams, which is sufficient to accommodate brushless motors, motor driver boards, and a battery for three axes. Similarly, the mass values of the reaction wheel and leg segment were assigned based on physically feasible dimensions. The initial x and y positions and velocities were determined by referencing average human dimensions, ensuring that the launch heights and speeds reflect normal throwing conditions while excluding highly aggressive scenarios. Three launch tests were performed to evaluate the performance of the system on the robot. While random initial values were chosen for launch Test 1, the other two tests examined the pre-defined maximum and minimum starting conditions for the launch. The initial values of positions and velocities for 3 different launch tests are given in Table 2. At the beginning of the simulation (t = 0), while the reaction wheel was at rest, the robot was started at the initial angular and linear positions and velocities given below. After the robot is launched, it is expected to stabilize in both $[\theta_1, \dot{\theta}_1](t=\infty) = \left[\frac{\pi}{2}\ rad, 0\frac{rad}{s}\right]$ and $[\theta_2, \dot{\theta}_2](t=\infty) = \left[\frac{\pi}{2}\ rad, 0\frac{rad}{s}\right]$. During the simulations, the maximum control torques τ_3 and τ_2 were constrained, as given in equation (23). The simulation screenshot taken during the launch and installation of the robot of launch Test 1 is given in Figure 10. Again, for Test 1, the 8-second test graphs of the outputs of the robot's MSM model are presented in Figures 11 to 17.In the graphs, it can be seen that control is achieved with negligible differences in the MSMB model and the robot balances steadily during the verification process.

Table 2. Initial values for simulations (Simülasyon için ilk Değerler)

$(t=0)$	*(Test 1)Random*	*(Test 2)Max*	*(Test 3)Min Initial*
x_1	$0\ m$	$0\ m$	$0\ m$
y_1	$1.7\ m$	$2.2\ m$	$1.2\ m$
$\dot{x}_1$	$2\ m/s$	2.7 m/s	0.0 m/s
$\dot{y}_1$	$1.2\ m/s$	2.5 m/s	0.0 m/s
θ_1	$1.832\ rad$	1.832 rad	1.832 rad
θ_2	$6.109\ rad$	6.109 rad	$6.109\ rad$
$\dot{\theta}_2, \theta_3, \dot{\theta}_3, \dot{\theta}_1$	[0,0,0,0]	[0,0,0,0]	[0,0,0,0]

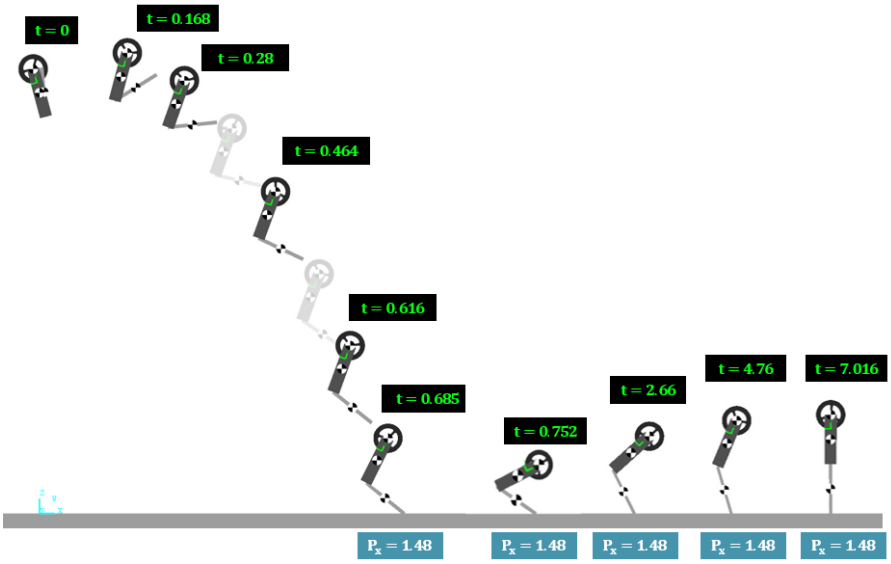

Figure 10. Screenshots of the robot's launch and upright standing for Test 1 were taken using MSM (Test 1 için robotun kalkış ve dik duruş anına ait ekran görüntüleri MSM kullanılarak alınmıştır.)

The linear position and velocity components of x_1 and y_1 when the robot falls during launch are given in Figure 11 and Figure 12, respectively. In Figure 12, the value of y_1 indicates that the robot stabilises at about 0.3 meters, which is its most upright position, after about 7 seconds. During the flight phase, until the moment of contact with the ground, in Figure 13 (Left), it can be observed that θ_1 closely follows the desired reference angle with negligible errors. As can be seen from the graph, within the 7-second duration, the value of θ_1 stabilises at the $\pi/2$ radians with negligible deviation. Similarly, $\dot{\theta}_1$ shows small oscillations around 0 rad/s, ensuring the successful stabilisation of the robot. In Figure 14, it is observed that θ_2 closely follows the desired variable trajectory (on the left) with negligible errors. In Figure 15 (on the left), it can be seen that the difference in the value of L_4^x quickly closes as the transition phase begins. In Figure 15 (on the right), the difference in the value of L_4^y due to the impact when the transition phase begins immediately closes and approaches the value of L_4^{des}. When the balance state is achieved for a duration of t_{ref}, the robot follows the variable trajectory L_4^{des} to straighten its posture.

After approximately 6 seconds, balance is fully achieved at $L_4^y = 0.342\ m$. As seen in Figure 16 (on the right), In order for the robot body to maintain balance in the desired position, the torque action signal applied to the reaction wheel joint is seen. As shown in Figure 16, despite the impact, the control torques τ_2 and τ_3 have managed to stabilise the robot within the desired range. Figure 16 shows that in the transition phase, within a period of 1 second, θ_4 is fixed at the desired reference value of $\pi/2$ radians. In the transition phase, within a 1-second duration, it can be observed in Figure 17 that θ_4 is stabilised at the desired reference value of $\pi/2$ radians. Figure 18 shows that the maximum and minimum initial values were tested for Test 2 and Test 3, for θ_1 (left), it follows the reference values θ_1^{FP} for both 2 test and is fixed at $\pi/2$. Similarly, in the flight phase, the variable reference trajectory θ_2^{ref} derived for θ_2 (Right) follows with negligible error, and the robot stabilises at $\pi/2$ in the balanced state. In Figure 19, more deflection occurred at the maximum initial values due to the effect of a larger impact. Here, the torque limits of the actuators driving A_0 and A_1 are reached, and the maximum value of ξ in the vertical direction is exceeded. Despite this, it is observed that the system has recovered and successfully reached balance. For both cases, after the ground contact occurs, it is observed that L_4^x is balanced around the reference point and L_4^y is balanced around the L_4^{des} value, which occurs gradually. Figure 19 shows that the main desired balanced upright posture was successfully achieved and maintained around the $\pi/2$ value throughout the ground phase.

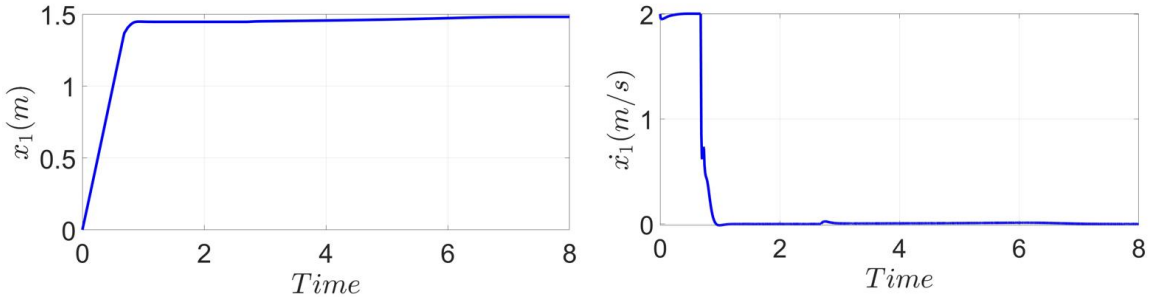

Figure 11. The graph shows the changes in the position of x_1 (left), its velocity $\dot{x}_1$ (right) during Test 1. (Grafik, Test 1 sırasında x_1'in konumundaki (sol) ve hızındaki (sağ) değişimleri göstermektedir)

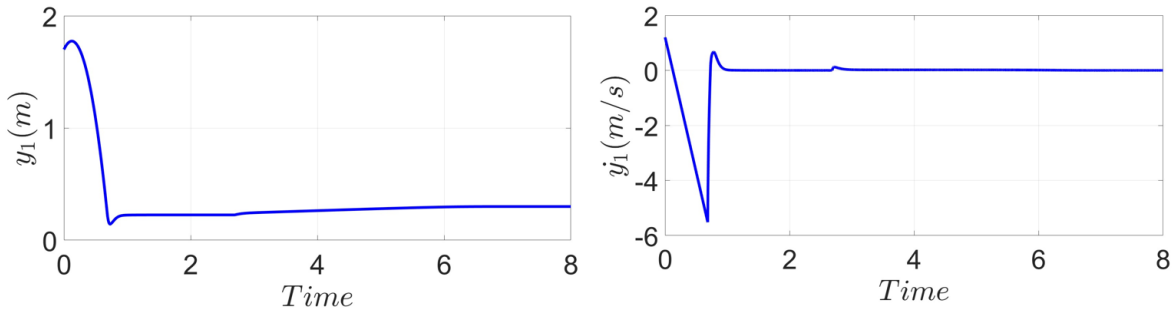

Figure 12. The graph shows the changes in the position of y_1(left), its velocity $\dot{y}_1$(right) during Test 1. (Grafik, Test 1 sırasında y_1'in konumundaki (sol) ve hızındaki (sağ) değişimleri göstermektedir)

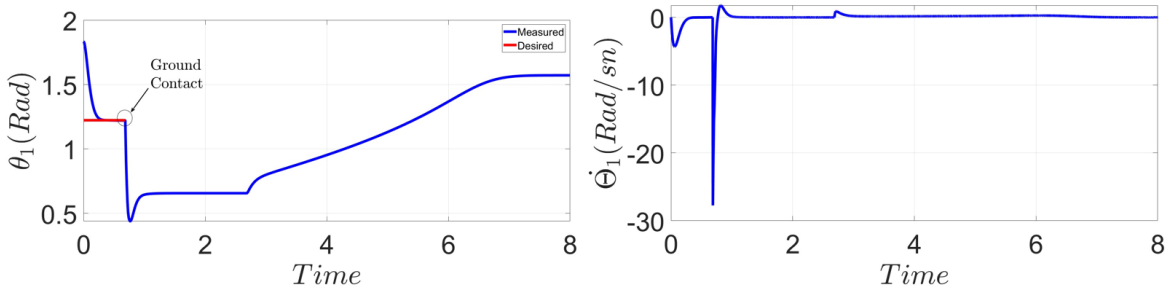

Figure 13. The graph shows the changes in the angular position of θ_1 (left) and angular velocity $\dot{\theta}_1$ (right) during Test 1. (Grafik, Test 1 sırasında θ_1'in açısal konumundaki (sol) ve açısal hızındaki (sağ) değişimleri göstermektedir)

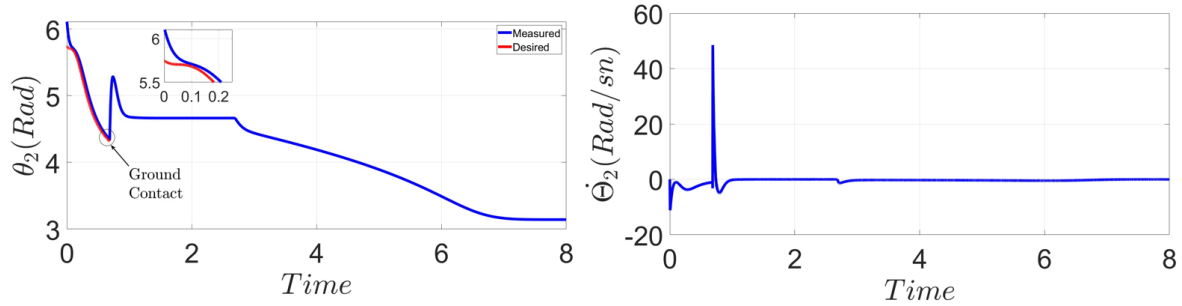

Figure 14. The graph shows the changes in the angular position of θ_2 (left) and angular velocity $\dot{\theta}_2$ (right) during Test 1. (Grafik, Test 1 sırasında θ_2'nin açısal konumundaki (sol) ve açısal hızındaki (sağ) değişimleri göstermektedir)

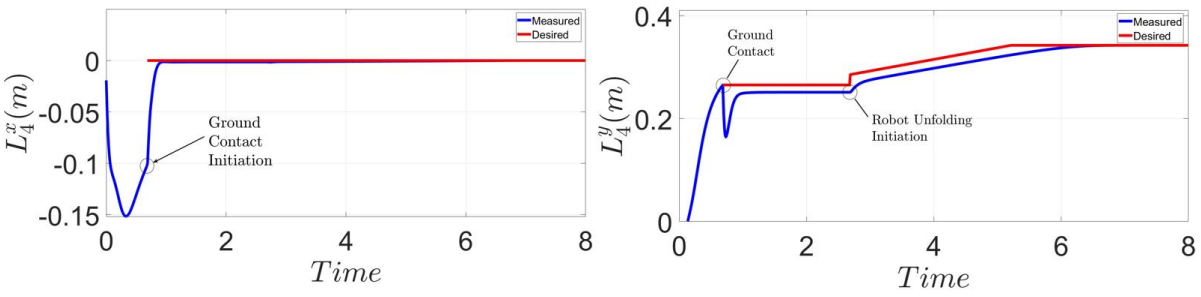

Figure 15. The graph shows the changes in the L_4^x (left) and L_4^y (right) during Test 1. (Grafik, Test 1 sırasında L_4^x'in (sol) ve L_4^y'in (sağ) değişimlerini göstermektedir)

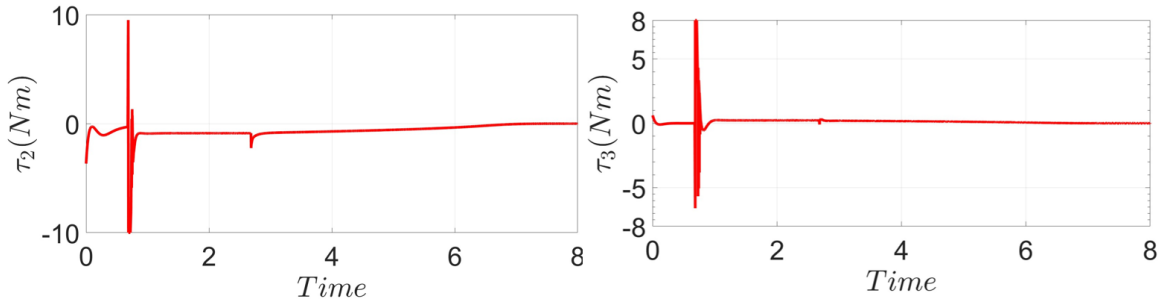

Figure 16. The graph shows the changes in the τ_2 and τ_3 during Test 1. (Grafik, Test 1 sırasında τ_2 ve τ_3'teki değişimleri göstermektedir)

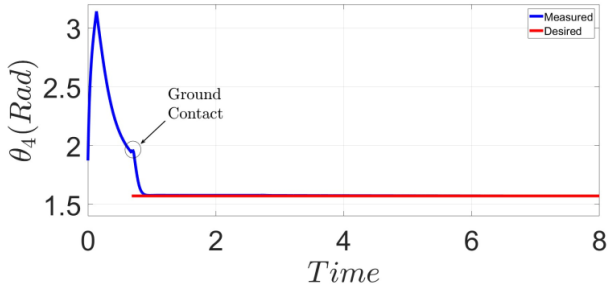

Figure 17. The graph shows the changes in the θ_4 for Test 1. (Grafik, Test 1 için θ_4'teki değişimleri göstermektedir)

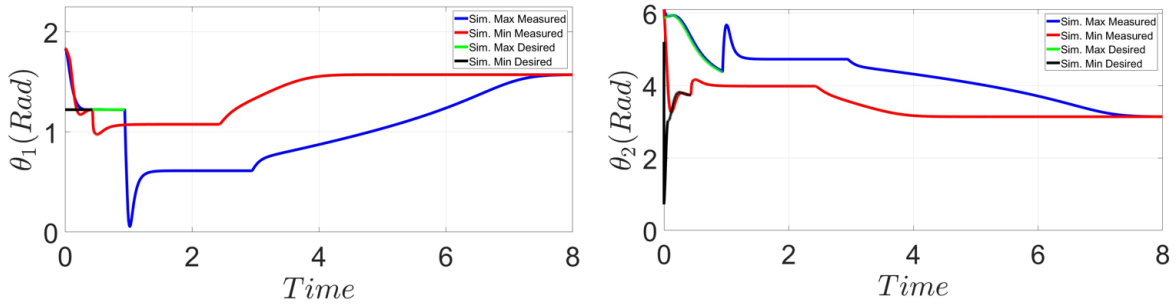

Figure 18. The graph shows the variation in the angular position of θ_1 (on the left) and θ_2 (on the right) during Test 2 and Test 3. (Grafik, Test 2 ve Test 3 sırasında θ_1'in açısal konumundaki (sol) ve θ_2'nin açısal konumundaki (sağ) değişimleri göstermektedir)

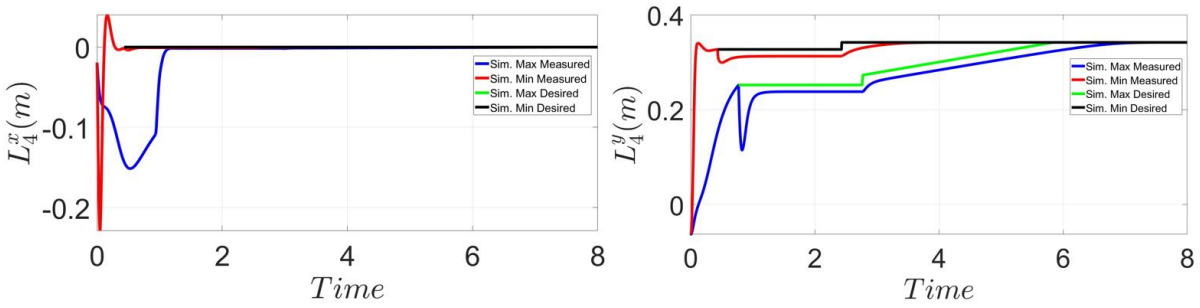

Figure 19. The graph shows the variation for L_4^x (left) and L_4^y (right) during Test 2 and Test 3 (Grafik, Test 2 ve Test 3 sırasında L_4^x'in (sol) ve L_4^y'in (sağ) değişimlerini göstermektedir)

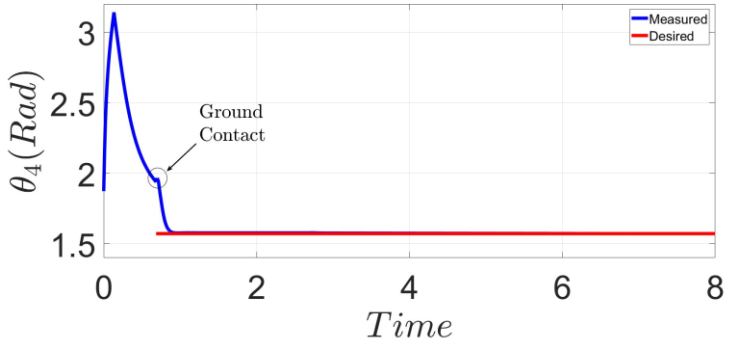

Figure 20. The graph shows the variation for θ_4 during Test 2 and Test 3. (Grafik, Test 2 ve Test 3 sırasında θ_4'teki değişimleri göstermektedir)

5. CONCLUSION AND RECOMMENDATIONS (SONUÇ VE ÖNERİLER)

After training and learning the control process, the robot has successfully maintained its balance in the MSM environment. While the planar motion has been effectively controlled in the simulation, there will be numerous parameters to consider when transitioning to a real-world application. Future work will therefore focus on the hardware implementation and experimental validation of the proposed control architecture. Our plan involves a Sim-to-Real transfer strategy. The ANN weights, optimized within the Simscape simulation, will be deployed onto a low-cost, powerful microcontroller such as an ESP32. This embedded controller will be responsible for real-time execution of the control loop. It will process data from onboard sensors for example Inertial Measurement Unit (IMU) for body orientation and motor encoders for joint angles and feed these states into the trained ANN. The network will then output the virtual impedance coefficients B_{ic} and K_{ic}, which the controller will use to calculate and command the necessary torques to the reaction wheel and leg motors. We anticipate challenges inherent to the sim-to-real gap, such as unmodeled dynamics, sensor noise, and actuator delays. To mitigate these issues, we plan to first refine the system identification of the physical prototype to improve the simulation's fidelity. Subsequently, if necessary, the pretrained ANN policy may undergo further online fine-tuning on the physical robot using reinforcement learning techniques to adapt to the nuances of the real world.

Additional weights that will come in 3D application will have costs in terms of controllability. This study was intentionally conducted in a 2D planar environment to reduce modeling complexity and computational cost during the development of the control and learning architecture. Extending the system to 3D presents additional challenges such as more degrees of freedom, increased dynamic instability, and complex interactions during landing impacts. Despite these difficulties, the proposed method was designed with extendibility in mind, and future work will focus on adapting the current structure to a 3D robotic system. As another potential solution, the use of brushless motors and propellers to actively drive the underactuated base axes for the mission can facilitate control. Since high thrust powers can be achieved in a short time, the desired robot balance will be easily allowed, control can be more easily achieved. However, we acknowledge

the importance of experimental validation and have outlined plans for comprehensive real-world testing in future work to confirm the practical applicability and effectiveness of the system.

DECLARATION OF ETHICAL STANDARDS (ETİK STANDARTLARIN BEYANI)

The author of this article declares that the materials and methods they use in their work do not require ethical committee approval and/or legal-specific permission.

Bu makalenin yazarı çalışmalarında kullandıkları materyal ve yöntemlerin etik kurul izni ve/veya yasal-özel bir izin gerektirmediğini beyan ederler.

AUTHORS' CONTRIBUTIONS (YAZARLARIN KATKILARI)

Halit HÜLAKO: He conducted the simulations, analyzed the results and performed the writing process.

Simülasyonları yapmış, sonuçlarını analiz etmiş ve maklenin yazım işlemini gerçekleştirmiştir.

CONFLICT OF INTEREST (ÇIKAR ÇATIŞMASI)

There is no conflict of interest in this study.

Bu çalışmada herhangi bir çıkar çatışması yoktur.

6. APPENDIX (EK)

The equations of motion are derived from Lagrange's equation, given in Eq. a1, where L is the Lagrangian function and τ_i are the input torques applied to the system. The dissipative energy in the robot mechanism can be disregarded, as there is no contribution from viscous torque.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = \tau_i$$

Positions and velocites of the center of mass of part 2 given in Eq. a2;

$$\begin{aligned} x_2 &= x_B - L_1\,\cos(\Theta_1) \\ &\quad + L_2\,\cos(\Theta_1+\Theta_2) \\ y_2 &= y_B + L_2\,\sin(\Theta_1+\Theta_2) - L_1\,\sin(\Theta_1) \\ \dot{x}_2 &= \frac{\partial}{\partial t}\,x_B + L_1\,\sin(\Theta_1)\,\frac{\partial}{\partial t}\,\Theta_1 \\ &\quad - L_2\,\sin(\Theta_1 \\ &\quad + \Theta_2)\left(\frac{\partial}{\partial t}\,\Theta_1 + \frac{\partial}{\partial t}\,\Theta_2\right) \\ \dot{y}_2 &= \frac{\partial}{\partial t}\,y_B - L_1\,\cos(\Theta_1)\,\frac{\partial}{\partial t}\,\Theta_1 \\ &\quad + L_2\,\cos(\Theta_1 \\ &\quad + \Theta_2)\left(\frac{\partial}{\partial t}\,\Theta_1 + \frac{\partial}{\partial t}\,\Theta_2\right) \end{aligned} \tag{a2}$$

Positions and velocites of the center of mass of disk given in Eq. a3;

$$\begin{aligned} x_3 &= x_B + L_1\,\cos(\Theta_1) \\ y_3 &= y_B + L_1\,\sin(\Theta_1) \\ \dot{x}_3 &= \frac{\partial}{\partial t}\,x_B - L_1\,\sin(\Theta_1)\,\frac{\partial}{\partial t}\,\Theta_1 \\ \dot{y}_3 &= \frac{\partial}{\partial t}\,y_B + L_1\,\cos(\Theta_1)\,\frac{\partial}{\partial t}\,\Theta_1 \end{aligned} \tag{a3}$$

The moments of inertia for all three limbs are given in Eq. a4. In Simscape Multibody, the inertia of a disk is calculated automatically. Here, the inner and outer dimensions of the disk are $R_i = 80\,mm$ and $R_o = 120\,mm$, respectively.

$$I_1 = \frac{{L_1}^2 m_1}{3},\; I_2 = \frac{{L_2}^2 m_2}{3},\; I_3 = \frac{m_3\left({R_i}^2 + {R_o}^2\right)}{2} \tag{a4}$$

The Lagrangian represents the difference between the kinetic energy and potential energy of the system and general formulation given in Eq. a5. The Lagrange function of the robotic system is given in the Eq. a6.

$$L = \sum_{i=1}^{3}\left[\frac{1}{2}m_i\left(\dot{x_i}^2 + \dot{y_i}^2\right) + \frac{1}{2}I_i\dot{\theta_i}^2\right] - \sum_{i=1}^{3}[m_i g h_i] \tag{a5}$$

$$\begin{aligned} L = {} & \frac{m_3\left((\dot{x}_B - \sigma_2)^2 + (\dot{y}_B + \sigma_3)^2\right)}{2} \\ & + \frac{m_2\left((\,\dot{y}_B - \sigma_3 + L_2\,\cos(\Theta_1+\Theta_2)\,\sigma_1)^2 + (\dot{x}_B + \sigma_2 - L_2\,\sin(\Theta_1+\Theta_2)\,\sigma_1)^2\right)}{2} \\ & + \frac{m_1\,{\dot{x}_B}^2}{2} + \frac{m_1\,{\dot{y}_B}^2}{2} - g\,m_1\,y_B - g\,m_3\,(y_B + L_1\,\sin(\Theta_1)) \\ & - g\,m_2\,(y_B + L_2\,\sin(\Theta_1+\Theta_2) - L_1\,\sin(\Theta_1)) + \frac{{L_1}^2 m_1\,{\dot{\Theta}_1}^2}{6} + \frac{{L_2}^2 m_2\,{\sigma_1}^2}{6} \\ & + \frac{m_3\left(\dot{\Theta}_1 + \dot{\Theta}_3\right)^2\left({R_i}^2 + {R_o}^2\right)}{4} \end{aligned} \tag{a6}$$

$$where$$
$$\sigma_1 = \dot{\Theta}_1 + \dot{\Theta}_2$$
$$\sigma_2 = L_1 \sin(\Theta_1)\ \dot{\Theta}_1$$
$$\sigma_3 = L_1 \cos(\Theta_1)\ \dot{\Theta}_1$$

Thus, the resulting equations of motion for generalized coordinates are listed in Eq. a7- a11. Applying the Lagrange equation for coordinate x_B:

$$(m_1 + m_2 + m_3)\,\ddot{x}_B - \frac{m_2\left(2\,L_2\,\sin(\sigma_3)\,\left(\ddot{\Theta}_1 + \ddot{\Theta}_2\right) - \sigma_1 - \sigma_2 + 2\,L_2\,\cos(\sigma_3)\,\left(\dot{\Theta}_1 + \dot{\Theta}_2\right)^2\right)}{2} - \frac{m_3\,(\sigma_2 + \sigma_1)}{2} = 0 \quad \text{(a7)}$$

$$where$$
$$\sigma_1 = 2\,L_1\,\cos(\Theta_1)\left(\frac{\partial}{\partial t}\,\Theta_1\right)^2$$
$$\sigma_2 = 2\,L_1\,\sin(\Theta_1)\,\frac{\partial^2}{\partial t^2}\,\Theta_1$$
$$\sigma_3 = \Theta_1 + \Theta_2$$

Applying the Lagrange equation for coordinate y_B:

$$(m_1 + m_2 + m_3)\,\ddot{y}_B + g\,m_1 + g\,m_2 + g\,m_3 - \frac{m_2\left(2\,L_2\,\sin(\Theta_1 + \Theta_2)\,\left(\dot{\Theta}_1 + \dot{\Theta}_2\right)^2 - 2\,L_2\,\cos(\Theta_1 + \Theta_2)\,\left(\ddot{\Theta}_1 + \ddot{\Theta}_2\right) - \sigma_1 + \sigma_2\right)}{2} - \frac{m_3\,(\sigma_1 - \sigma_2)}{2} = 0 \quad \text{(a8)}$$

$$where$$
$$\sigma_1 = 2\,L_1\,\sin(\Theta_1)\,{\dot{\Theta}_1}^2$$
$$\sigma_2 = 2\,L_1\,\cos(\Theta_1)\,\ddot{\Theta}_1$$

Applying the Lagrange equation for coordinate Θ_1:

$$\left(\frac{{L_1}^2\,m_1}{3} + {L_1}^2\,m_2 + {L_1}^2\,m_3 + \frac{4\,{L_2}^2\,m_2}{3} + \frac{{R_i}^2\,m_3}{2} + \frac{{R_o}^2\,m_3}{2} - 2\,L_1\,L_2\,m_2\,\cos(\Theta_2)\right)\ddot{\Theta}_1 + \frac{4\,{L_2}^2\,m_2\,\ddot{\Theta}_2}{3} + \frac{{R_i}^2\,m_3\,\ddot{\Theta}_3}{2} + \frac{{R_o}^2\,m_3\,\ddot{\Theta}_3}{2} - L_1\,m_2\,\cos(\Theta_1)\,\ddot{y}_B + L_1\,m_3\,\cos(\Theta_1)\,\ddot{y}_B + L_1\,m_2\,\sin(\Theta_1)\,\ddot{x}_B - L_1\,m_3\,\sin(\Theta_1)\,\ddot{x}_B + L_2\,m_2\,\cos(\Theta_1 + \Theta_2)\,\ddot{y}_B - L_1\,g\,m_2\,\cos(\Theta_1) + L_1\,g\,m_3\,\cos(\Theta_1) - L_2\,m_2\,\sin(\Theta_1 + \Theta_2)\,\ddot{x}_B + L_2\,g\,m_2\,\cos(\Theta_1 + \Theta_2) + L_1\,L_2\,m_2\,\sin(\Theta_2)\,{\dot{\Theta}_2}^2 - L_1\,L_2\,m_2\,\cos(\Theta_2)\,\ddot{\Theta}_2 + 2\,L_1\,L_2\,m_2\,\sin(\Theta_2)\,\dot{\Theta}_2\,\dot{\Theta}_1 = 0 \quad \text{(a9)}$$

Applying the Lagrange equation for coordinate Θ_2:

$$\frac{4\,{L_2}^2\,m_2}{3}\,\ddot{\Theta}_2+\frac{4\,{L_2}^2\,m_2\,\ddot{\Theta}_1}{3}+L_2\,m_2\,\cos(\Theta_1+\Theta_2)\,\ddot{y}_B-L_2\,m_2\,\sin(\Theta_1+\Theta_2)\,\ddot{x}_B+L_2\,g\,m_2\,\cos(\Theta_1+\Theta_2)-L_1\,L_2\,m_2\,\sin(\Theta_2)\,\left(\dot{\Theta}_1\right)^2-L_1\,L_2\,m_2\,\cos(\Theta_2)\,\ddot{\Theta}_1=\tau_2 \tag{a10}$$

Here, the actuated generalized coordinates are the 3rd and 2nd coordinates. In the equation of motion where the disk is present, it is clearly seen in Eq. a11 that the acceleration of $\ddot{\Theta}_1$ is directly affected by the actuation. Applying the Lagrange equation for coordinate Θ_3:

$$\frac{m_3\left(\ddot{\Theta}_1+\ddot{\Theta}_3\right)\left({R_{in}}^2+{R_{out}}^2\right)}{2}=\tau_3 \tag{a11}$$

REFERENCES

[1] C. Rui, I. V. Kolmanovsky, and N. H. McClamroch, "Nonlinear attitude and shape control of spacecraft with articulated appendages and reaction wheels," IEEE Trans Automat Contr, vol. 45, no. 8, pp. 1455–1469, Aug. 2000, doi: 10.1109/9.871754.

[2] G. Shengmin and C. Hao, "A comparative design of satellite attitude control system with reaction wheel," in Proceedings - First NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2006, 2006, pp. 359–362. doi: 10.1109/AHS.2006.2.

[3] Y. Yang, "Spacecraft Attitude and Reaction Wheel Desaturation Combined Control Method," IEEE Trans Aerosp Electron Syst, vol. 53, no. 1, pp. 286–295, Feb. 2017, doi: 10.1109/TAES.2017.2650158.

[4] P. Zhang, Z. Wu, H. Dong, M. Tan, and J. Yu, "Reaction-Wheel-Based Roll Stabilization for a Robotic Fish Using Neural Network Sliding Mode Control," IEEE/ASME Transactions on Mechatronics, vol. 25, no. 4, pp. 1904–1911, Aug. 2020, doi: 10.1109/TMECH.2020.2992038.

[5] M. Gajamohan, M. Merz, I. Thommen, and R. D'Andrea, "The Cubli: A cube that can jump up and balance," in IEEE International Conference on Intelligent Robots and Systems, 2012, pp. 3722–3727. doi: 10.1109/IROS.2012.6385896.

[6] M. Muehlebach, G. Mohanarajah, and R. D'Andrea, "Nonlinear analysis and control of a reaction wheel-based 3D inverted pendulum," in Proceedings of the IEEE Conference on Decision and Control, Institute of Electrical and Electronics Engineers Inc., 2013, pp. 1283–1288. doi: 10.1109/CDC.2013.6760059.

[7] T. L. Brown and J. P. Schmiedeler, "Reaction Wheel Actuation for Improving Planar Biped Walking Efficiency," IEEE Transactions on Robotics, vol. 32, no. 5, pp. 1290–1297, Oct. 2016, doi: 10.1109/TRO.2016.2593484.

[8] S. I. Han and J. M. Lee, "Balancing and velocity control of a unicycle robot based on the dynamic model," IEEE Transactions on Industrial Electronics, vol. 62, no. 1, pp. 405–413, Jan. 2015, doi: 10.1109/TIE.2014.2327562.

[9] J. Trentin, S. da Silva, J. Ribeiro, and H. Schaub, "Inverted Pendulum Nonlinear Controllers Using Two Reaction Wheels: Design and Implementation," IEEE Access, vol. PP, p. 1, May 2020, doi: 10.1109/ACCESS.2020.2988800.

[10] M. Kim, J. Kim, and D. Yun, "Compact Posture Control System for Jumping Robot Using an Air Reaction Wheel," IEEE/ASME Transactions on Mechatronics, 2024, doi: 10.1109/TMECH.2024.3398636.

[11] M. Zabihi and A. Alasty, "Dynamic stability and control of a novel handspringing robot," Mech Mach Theory, vol. 137, pp. 154–171, Jul. 2019, doi: 10.1016/j.mechmachtheory.2019.03.018.

[12] S. Haoran, X. Yong, L. Jiali, J. Xinyang, and Y. Jie, "Marsbot: A monopod robot capable of achieving three-dimensional dynamic and stable

jumping," Proc Inst Mech Eng C J Mech Eng Sci, vol. 236, no. 1, pp. 552–565, Jan. 2022, doi: 10.1177/09544062211024311.

[13] A. Anzai, T. Doi, K. Hashida, X. Chen, L. Han, and K. Hashimoto, "Monopod robot prototype with reaction wheel for hopping and posture stabilisation," 2021. [Online]. Available: http://creativecommons.org/licenses/by-nc-nd/4.0/

[14] T. L. Brown and J. P. Schmiedeler, "Reaction Wheel Actuation for Improving Planar Biped Walking Efficiency," IEEE Transactions on Robotics, vol. 32, no. 5, pp. 1290–1297, Oct. 2016, doi: 10.1109/TRO.2016.2593484.

[15] F. Roscia, A. Cumerlotti, A. Del Prete, C. Semini, and M. Focchi, "Orientation Control System: Enhancing Aerial Maneuvers for Quadruped Robots," Sensors, vol. 23, no. 3, Feb. 2023, doi: 10.3390/s23031234.

[16] Q. Zhu, X. Wu, Q. Lin, and W.-N. Chen, "Two-Stage Evolutionary Reinforcement Learning for Enhancing Exploration and Exploitation," 2024. [Online]. Available: www.aaai.org

[17] L. Deng and S. Liu, "Incorporating Q-learning and gradient search scheme into JAYA algorithm for global optimization," Artif Intell Rev, vol. 56, pp. 3705–3748, Dec. 2023, doi: 10.1007/s10462-023-10613-1.

[18] L. Deng and S. Liu, "Snow ablation optimizer: A novel metaheuristic technique for numerical optimization and engineering design," Expert Syst Appl, vol. 225, Sep. 2023, doi: 10.1016/j.eswa.2023.120069.

[19] S. Zhang, H. Zhang, and Y. Fu, "Leg Locomotion Adaption for Quadruped Robots with Ground Compliance Estimation," Appl Bionics Biomech, vol. 2020, 2020, doi: 10.1155/2020/8854411.