International Journal on Advanced Science Engineering Information Technology

Vol.13 (2023) No. 3
ISSN: 2088-5334

Design an Intelligent Balanced Control of Quadruped Legs Based on Adaptive Neuro-Fuzzy Inference System (ANFIS)

Sigit Wasista [a,b], Handayani Tjandrasa [a,*], Supeno Djanali [a]

[a] *Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia*
[b] *Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Indonesia*
*Corresponding author: *handatj@its.ac.id*

***Abstract*— This study discusses the 12 DoF stability control algorithm for Quadruped robot legs to adjust the balance on irregular terrain. The source of the instability is the irregularity of the ground surface and external forces. Therefore, dynamic stability criteria are needed to plan the robot's movement and restore balance for the movement of a four-legged robot with a dynamic gait over an irregular terrain. The novelty of this study is the use of 12 ANFIS at once to manage the 12 DoF of each leg, which are grouped into four sections, and each section consists of 3 ANFIS. The ANFIS method is used as an algorithm to move the 12-DoF robot legs by training some robot leg movement data based on the slope angle of the surface. The results of training with the ANFIS method can be optimal if the number of rules is close to the given training data. From 29 body tilt angle position data and 12-DOF robot legs, good results will be obtained if the 5x5 number membership function is used for each input which will produce 25 ANFIS rules and combined using the Gaussian type so that it can produce RMSE = 0.068233. The next research is to develop reliable methods such as Zero Moment Point (ZMP) combined with the BPNN or ANFIS methods so that it is expected to get a reliable robot body balance.**

***Keywords*— Quadruped robot; balanced control; stabilize movement; robotic simulation.**

Manuscript received 19 Jan. 2023; revised 2 Feb. 2023; accepted 1 Mar. 2023. Date of publication 30 Jun. 2023.

I. INTRODUCTION

Several medium-sized quadruped robots have been developed to investigate the swing of the robot's legs for effect on balancing its gait. For example, some mid-sized quadruped robots, such as Kim et al. [1] developed an open-source dynamic quadruped robot, PADWQ.

When considering the dynamic motion of a legged robot, one must pay attention to its stability. The robot needs to complete the given task while maintaining balance and moving smoothly. Shi et al. [2] studied the walking gait of a four-legged animal with a bionic structure capable of performing smooth movement in all directions, inverse kinematic solution using the Denavit-Hartenberg (D-H) method was explored moving the legs of a quadruped robot. Sun et al. [3] studied a learning-based control architecture for quadrupedal self-balancing by adapting to some unexpected scenes from continuous external interference. Şen et al. [4] simulated a 3-DoF linear leg model with the PIλDμ position control system, a control designed by selecting various fraction order parameters as a comparison with the classic PID control. Wasista et al. [5] studied models of robotic leg swing trajectories as a periodic balanced motion test on the new design of a Quadruped Robot called "Kancil-V2". Lee et al. [6] developed robust AiDIN-VI control based on precise torque measurement. Researchers have proposed various static and dynamic stability analysis methods to evaluate the stability of legged robots. Jia et al. [7] presented a new stability criterion for the motion of quadruped robots with dynamic gaits running over irregular terrain. Neuro-fuzzy is a combination of two systems: fuzzy logic systems and artificial neural networks. System neuro-fuzzy based on the fuzzy inference system trained using a learning algorithm derived from the artificial neural network system. With this, the neuro-fuzzy system has all the advantages possessed by fuzzy inference systems and artificial neural network systems[8]. Qin et al. [9] Examining Stable balance adjustment structure on quadrupedal robotic legs based on the bionic lateral swing posture designed, and the leg kinematics model. Cui et al. [10] discuss the virtual suspension model control to solve the robot problem with a line or fulcrum to maintain a standing balance. Bakırcıoğlu et al. [11] used Adaptive Neuro-Fuzzy Inference System (ANFIS) to obtain position control data (trajectory)

on one leg of a four-legged robot, which was previously controlled by the PID method so that position control can be replaced with ANFIS. Zhu et al. [12] studied the walking stability of quadrupedal robots with a stabilization reactive control strategy against lateral disturbances based on the theory of catch point stability. Sun et al. [13] have researched adaptive robotic quadruped motor control for fast postural adaptation in various fields through new distributed feedback-based reflexes with online learning. Gonzalez-Luchena et al. [14] present a new controller for a quadrupedal robot to walk using asymmetrical gait patterns in various scenarios, with simple and fast operation. Biswal et al. [15] surveyed various design and development approaches to quadrupedal robots, and environmental perception techniques. Tripathy et al. [16] build fast browsing Random Tree-based move organizer that completes jumps in reproduction with Little Dog Robot over a very ominous landscape. Gonzalez et al. [17] studied running quadruped robots on narrow paths or walking in an area of almost zero support. Hwangbo et al. [18] have researched a new method for training neural network weights in simulations and used it in a quadruped robotic leg control system. Sun et al. [19] have presented a dynamic balance control method to increase quadrupedal robot's stability by adjusting its legs' position. Of all the references mentioned above, no one has used balance control using the 12-ANFIS method to adjust the balance of the robot's body so it does not fall at any time on an inclined surface.

In this study, we fabricated an open-source dynamic quadruped robot that can be assembled from components available in the domestic market. The robot body consists of standard 3D-printed plastic so that assembling the robot can be easily fabricated, duplicated, and shared without using a CNC machine. Based on this, the robot can be produced with low-cost plastic 3D printing, with strength according to the material used, namely ABS, or more substantial than that. While the drive motor (actuator) uses a servo with a lifting force of +/-30-35 Kg/cm, which is controlled by an onboard computer, and the power supply/adapter is ready-to-use. It can be purchased from the domestic market. It tested the movement of the robot's legs using a simple Arduino-based drive controller to move the servo degrees for the balance controller according to the IMU sensor.

II. Materials and Method

The robot consists of one body and four limbs. Each consists of three parts: hips, thighs, and calves, so there are three Degrees of Freedom (DOF) in each limb. A DC motor actuator drives DOF. The main section dimensions of a Quadruped model are shown in Fig. 1. The proposed Quadruped robot has 12 (4x3) DOF and consists of 4 combinations of shoulders and legs with 3-DOF as shown in Fig. 2. The Main section sizes of a Quadruped Robot models are shown in Table 1. The kinematic design details of the leg are shown in Fig. 3, and D-H parameters are shown in Table 2.

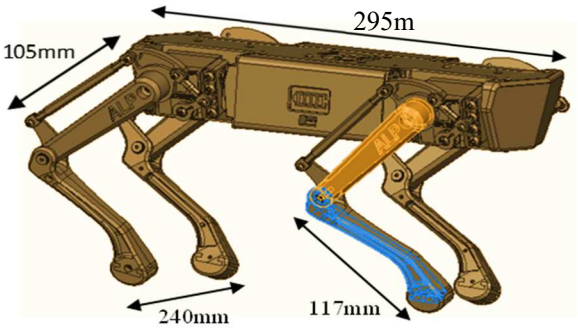


Fig. 1 The Main Body of Quadruped Robot

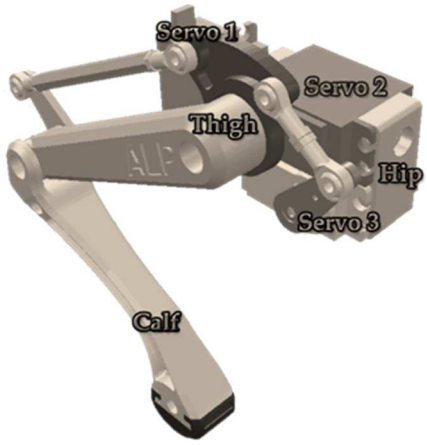


Fig. 2 The 3-DOF Hip, Thigh, and Calf

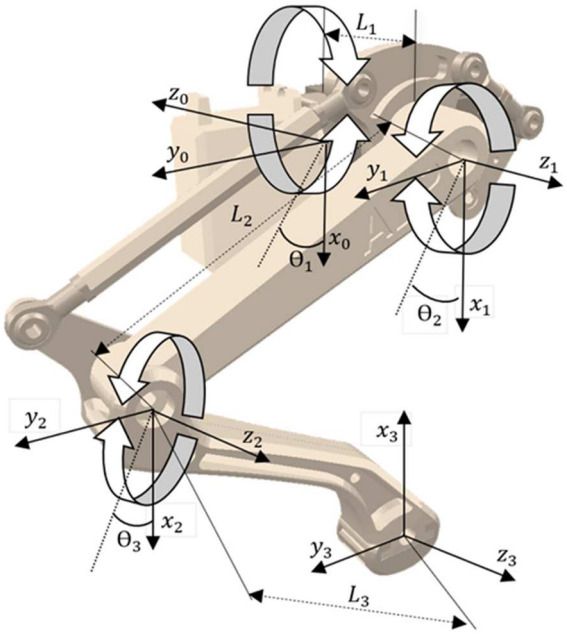


Fig. 3 The Kinematic Design and D-H Parameter

TABLE I
Main Section Size of the Robot

Part	Length(mm)
Body length	295
Body width	240
Thigh	105
Calf	117

TABLE II
D-H PARAMETERS OF QUADRUPED ROBOT

Link	a_i	d_i	α_i	θ_i
0-1	L_1	0	0	θ_1
1-2	L_2	0	0	θ_2
2-3	L_3	0	0	θ_3

Table 2 summarizes the explanation of the Denavit-Hartenberg Modeling Parameters for the Quadruped robot. This D-H parameter model, the Homogeneous Transformation Matrix [20], [21], formulates forward and inverse kinematics calculations [4], [22], [23]. Based on the D-H Parameter modeling value [5], [12], [24], a transformation matrix can be filled for each entry as follows:

$$T_0^1=\begin{bmatrix} cos(\theta_1) & -sin(\theta_1) & 0 & -L_1cos(\theta_1) \\ sin(\theta_1) & cos(\theta_1) & 0 & -L_1sin(\theta_1) \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$T_1^2=\begin{bmatrix} cos(\theta_2) & -sin(\theta_2) & 0 & L_2\,cos(\theta_2) \\ sin(\theta_2) & cos(\theta_2) & 0 & L_2\,sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$T_2^3=\begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0 & L_3cos(\theta_3) \\ sin(\theta_3) & cos(\theta_3) & 0 & L_3sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

The joint angle position equations (θ_1 , θ_2 and θ_3) have been obtained as in the following equation.

$$\theta_1=-atan2(-y_3,x_3)-atan2\left(\sqrt{x_4^2-y_4^2-L_1^2},-L_1\right) \quad (4)$$

$$\theta_2=atan2\left(z_4,\sqrt{x_4^2+y_4^2-L_1^2}\right)-atan2(L_3\sin(\theta_3),L_2+L_3\cos(\theta_3)) \quad (5)$$

$$\theta_3=atan2\left(-\sqrt{1-D^2},D\right) \quad \text{for leg 1 and 3} \quad (6)$$

$$\theta_3=atan2\left(\sqrt{1-D^2},D\right) \quad \text{for leg 2 and 4} \quad (7)$$

III. RESULTS AND DISCUSSION

The control of a quadruped robot consists of balance control with Back Propagation Neural Network (BPNN), and ANFIS alternately controlling the robot's leg movements. The proposed model for the experiment is shown in Fig. 4.

A. Modeling based on Neural Network

The neural network architecture used here is a BPNN, which consists of three layers: the input layer, the hidden layer, and the output layer [25], [26]. Those three layers are input X_i, hidden layer O_i^{L1}, and the output layer O_i^{L2}. The input layer and hidden layer are connected by weight $W^{L1-L2}{}_{i,j}$ and the hidden layer and output layer are connected by weight $W^{L2-L3}{}_{i,j}$. Learning in artificial neural networks involves data pairs, inputs, and outputs. From the multiplication process between the input data and the weighting, temporary output data with errors will be generated, which are then forwarded to the output layer. The output layer will respond called a temporary output. If the output of the output has not reached the desired limit or the mean error square, it will be recalculated to replace the weights in the hidden layer and weights back to the input layer. The following mathematical formula is the basis of the above algorithm:

1) The Forward Propagation formula:

Layer One: $$O_i^{L1}=X_i \quad (8)$$

Layer Two: $$a_j=\sum_{i=1}^{N} O_i^{L1}W^{L1-L2}{}_{i,j} \quad (9)$$

And $$O_i^{L2}=\frac{1}{1+exp^{-(a_j+bias_j)}} \quad (10)$$

Layer Tree: $$b_k=\sum_{j=1}^{N} O_j^{L2}W^{L2-L3}{}_{i,j} \quad (11)$$

and $$O_k^{L3}=\frac{1}{1+exp^{-(b_k+bias_k)}} \quad (12)$$

2) The Backward Propagation formula:

Layer Three, The Error Output formula:

$$Err(MSE)=\frac{1}{2}(O^{L3}-O_K^D)^2 \quad (13)$$

$$b_k=\partial_3=\frac{dErr_k}{db_k}=O_K^D-O_K^{L3} \quad (14)$$

$$a_j=\partial_2=\frac{dErr_k}{da_j}=\frac{dErr_k}{db_k}\cdot\frac{db_k}{dO_j^{L2}}\cdot\frac{dO_j^{L2}}{da_j} \quad (15)$$

$$Err_j=\frac{dErr_k}{db_k}\cdot\frac{db_k}{dO_j^{L2}}=\sum_{k=1}^{L}\partial_3.W_{i,j}^{L2-L3} \quad (16)$$

$$a_i=\partial_2=Err_j.O_j^{L2}.(1-O_j^{L2}) \quad (17)$$

Layer Two, The Error Output formula:

3) The Bias formula:

L2-L3 Weight Updates:

$$\Delta w_{j,k}^{L2-L3}=\eta\frac{dErr_k}{dw_{j,k}^{L2-L3}}=\eta\frac{dErr_k}{db_k}\cdot\frac{db_k}{dw_{j,k}^{L2-L3}}=\eta\partial_3O_j^{L2} \quad (18)$$

$$w^{L2-L3}=W^{L2-L3}+\Delta w_{j,k}^{L2-L3} \quad (19)$$

L1-L2 Weight Updates:

$$\Delta w_{i,j}^{L1-L2}=\eta\frac{dErr_k}{dw_{i,j}^{L1-L2}}=\eta\frac{dErr_j}{da_j}\cdot\frac{da_j}{dw_{i,j}^{L1-L2}}=\eta\partial_2O_i^{L1} \quad (20)$$

$$w^{L1-L2}=w^{L1-L2}+\Delta w_{i,j}^{L1-L2} \quad (21)$$

L2-L3 Bias Updates:

$$\Delta bias_k^{L2-L3}=\eta\frac{dErr_k}{dbias_k^{L2-L3}}=\eta\frac{dErr_k}{db_k}\cdot\frac{db_k}{dbias_k^{L2-L3}}=\eta\partial_3.1 \quad (22)$$

$$bias^{L2-L3}=bias^{L2-L3}+\Delta bias_k^{L2-L3} \quad (23)$$

L1-L2 Bias Updates:

$$\Delta bias_j^{L1-L2}=\eta\frac{dErr_j}{dbias_j^{L1-L2}}=\eta\frac{dErr_k}{da_j}\cdot\frac{da_j}{dbias_k^{L1-L2}}=\eta\partial_2.1 \quad (24)$$

$$bias^{L2-L3}=bias^{L2-L3}+\Delta bias_j^{L2-L3} \quad (25)$$

4) Learning Rate Updates:

$$LR=\mu(k)=\frac{\mu_0}{1+\frac{k}{k_0}} \quad (26)$$

(a) Pitch = 0°, Roll = 20° (a) Pitch = 15°, Roll = 0°

Fig. 4 Quadruped Robot model.

TABLE III
THE TILT ANGLE POSITION DATA OF THE ROBOT BODY DURING TESTING

Num	Pitch	Roll	θLF1	θLF2	θLF3	θLR1	θLR2	θLR3	θRF1	θRF2	θRF3	θRR1	θRR2	θRR3
1	0	20	-10.5	-24	20.1	10.5	-11.9	-14.7	-3.5	-26.4	-16.9	3.2	-31.3	-14
2	0	15	-7	-18.1	16.7	7	-8.9	-11.7	-3.2	-20.7	-10.8	2.9	-24.6	-7.4
3	0	10	-4.1	-11.7	13	4.1	-5.6	-8.4	-2.5	-15.9	-5.6	2.3	-19.1	-2
4	0	5	1.8	-4.6	9.1	1.8	-1.9	-4.9	-1.5	-11.7	-0.9	1.3	-14.3	2.7
5	0	0	0	3	5	0	2	-1	0	-8	3.5	0	-10	7
6	0	-5	1.3	11.3	0.6	-1.3	6.3	3.2	1.9	-4.6	7.6	-1.8	-6.1	10.9
7	0	-10	2.3	20.6	-4.3	-2.3	11.1	7.9	4.4	-1.6	11.5	-4.1	-2.4	14.5
8	0	-15	2.9	31	-9.7	-2.9	16.6	13.3	7.5	1	15.1	-7	0.9	17.7
9	0	-20	3.2	43.6	-16.1	-3.2	23.2	19.9	11.2	3.4	18.5	-10.5	3.9	20.7
10	20	0	0	48.1	-15.6	0	0.6	-23.9	0	-32.2	-16.5	0	-8.6	30
11	15	0	0	33.4	-8.2	0	-0.7	-18	0	-25	-9.4	0	-7.3	24
12	10	0	0	21.7	-2.7	0	-0.9	-12.2	0	-19	-4	0	-7.1	18.2
13	5	0	0	11.8	1.6	0	0	-6.5	0	-13.4	0.2	0	-8	12.6
14	-5	0	0	-4.8	7.4	0	5	4.6	0	-2.6	5.9	0	-13	1.4
15	-10	0	0	-11.8	8.9	0	9.1	10.3	0	2.9	7.2	0	-17.1	-4.4
16	-15	0	0	-18	9.3	0	14.5	16.6	0	8.7	7.6	0	-22.5	-10.7
17	-20	0	0	-23.4	8.7	0	21.9	24.3	0	14.8	6.9	0	-30	-18.4
18	15	15	-3.1	5.5	6.8	6.1	-6.7	-21.8	-1.6	-30.6	-17.5	3.3	-14.4	9.2
19	10	10	-2.2	4.2	6.5	3.3	-5	-14.6	-1.4	-21.8	-8.7	2.2	-12.7	8.7
20	5	5	-1.1	3.4	5.9	1.4	-2.1	-7.7	-0.9	-14.6	-2	1.1	-11.3	7.9
21	-5	-5	-1.1	3	3.9	-0.9	7.3	5.9	1.4	-1.7	8	-1.1	-9	5.9
22	-10	-10	2.2	3.4	2.6	-1.4	14	13.5	3.3	4.8	11.5	-2.1	-8.2	4.5
23	-15	-15	3.3	4.2	1.1	-1.6	23.3	23	6.1	11.5	14	-3.1	-7.7	3.1
24	15	-15	1.5	47.5	-16.6	-3.3	6.4	-3.2	3.1	-12.3	5.3	-6.1	-1.3	27.9
25	10	-10	1.4	29.3	-7.5	-2.2	4.7	-2.7	2.1	-10.7	5	-3.3	-3	20.6
26	5	-5	0.9	15.2	-0.6	-1.1	3.3	-2	1.1	-9.3	4.4	-1.4	-5.9	13.8
27	-5	5	-1.4	-7.9	9.6	1.1	1	0.1	-1.1	-6.9	2.4	0.9	-15.3	0
28	-10	10	-3.4	-18	13.2	2.1	0.2	1.5	-2.2	-6	1.1	1.4	-22	-7.6
29	-15	15	-6.1	-27.3	15.8	3.1	-0.3	2.9	-3.3	-5.2	-0.4	1.5	-31.4	-17.2

Fig. 5 shows the BPNN architecture [27], [28] where the input layer is designed with three n: the gyro sensor is x, and y. Then the hidden layer uses 20 nodes, while the Output layer uses 12. Table 3 shows the input data from measuring the angle of the quadruped robot's legs when standing on a flat plane and an inclined plane with a predetermined tilt angle. The unique data is the angle between the legs of the robot as much as the tilt test of the robot body both in pitch and roll, as shown in Fig. 6.

B. Modeling based on ANFIS

ANFIS consists of many nodes and specific functions. The value of nodes and functions can be changed to approximate the learning value. After studying the sample many times, the node and function will get the best value.

After learning, the difference between the sample and output values is an objective function optimized using the Gradient Descent Method. The Chain Rule is the basis for adjustment, which aims to subtract the mean squared error of the objective function.

From the perspective of inverse kinematics, solving for angle θ_1 Eq. (4) only needs to use the x and y. Therefore, only two inputs from ANFIS1 (x, y) exist. Whereas angle θ_2 Eq. (5) and angle θ_3 Eq. (6) and Eq. (7) depend on x and y, so ANFIS2 and ANFIS3 using two inputs (x, y) in Fig. 7.

ANFIS, Adaptive Neuro-Fuzzy Inference System, is a kind of artificial neural network based on the Fuzzy Inference System from Takagi-Sugeno [29] [30].

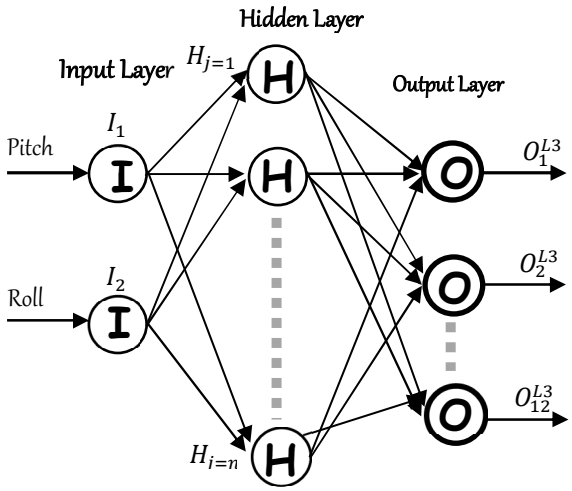
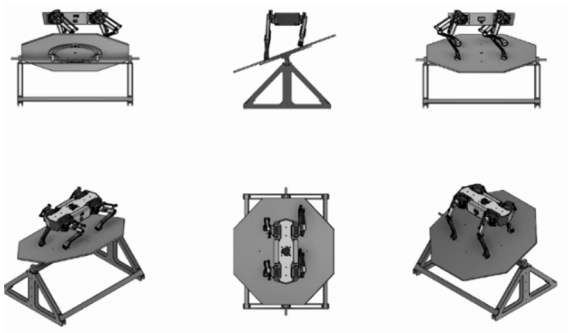
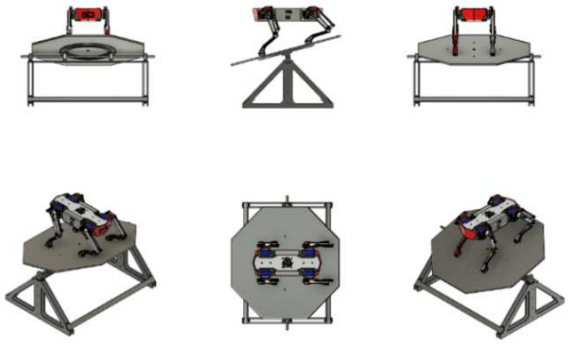


Fig. 5 Multilayer BPNN



(a) Pitch=0°, Roll= 20°



(b) Pitch=15°, Roll= 0°

Fig. 6 Tilt and Roll of the robot body

The following is a first-order Takagi-Sugeno fuzzy model, namely the set with two fuzzy IF-THEN rules [31] as follows:

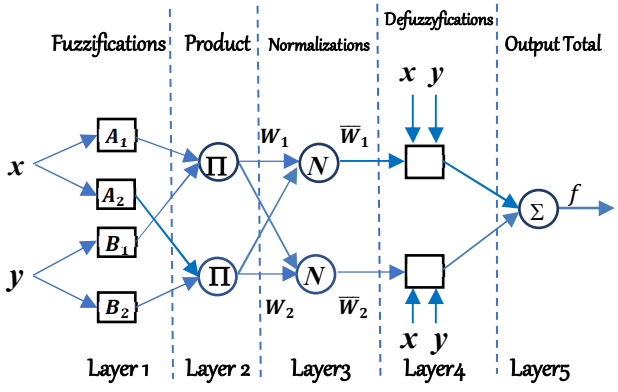


Fig. 7 ANFIS

Rule 1:

$$IF\ x\ is\ A_1\ and\ y\ is\ B_1\ THEN\ f_1 = p_1x + q_1y + r_1 \quad (27)$$

Rule 2:

$$IF\ x\ is\ A_2\ and\ y\ is\ B_2\ THEN\ f_2 = p_2x + q_2y + r_2 \quad (28)$$

All nodes in layer one are adaptive nodes described with the box field [30], while the output of layer one is a fuzzy membership class, given by Eq. (29) and Eq. (30) the following:

$$O_{1,i} = \mu A_i(x), i = 1,2, \quad (29)$$

or

$$O_{1,i} = \mu B_{i-2}(y), i = 3,4 \quad (30)$$

The membership class of the fuzzy set is A_1, A_2, B_1 and B_2 is $O_{1,i}$. Whereas A_1 and $B_{i-2}$are label languages such as "high" or "low" associated with this node, with input variables x and y for node i. So $\mu A_1(x)$ and $\mu B_{i-2}(y)$ can take a fuzzy membership function. So if the membership function uses a bell shape, then $\mu A_i(x)$ will be as in Eq. (31) and Eq. (32).

$$\mu A_i(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]b_i}, i - 1,2, \quad (31)$$

Or uses Gausian such as:

$$\mu A_i(x) = exp\left[-\left(\frac{x-c_i}{a_i}\right)^2\right] \quad (32)$$

The set of premise parameters consisting of a_i, b_i, and c_i, has a value that varies, causing variations in the shape of the Bell and Gaussian functions, and affecting fuzzy sets with various forms of membership functions. Second layer involves a multiplication operator, which multiplies between inputs to produce an output product in the form of nodes representing the firing strength rules. For more details in Eq. (33) below. The node's shape is denoted by a circle labeled Π, which means the product of the incoming signal that produces the product output.

$$O_{2,i} = \omega_i = \mu A_i(x) * \mu B_i(y), i = 1,2. \quad (33)$$

The third layer is also called the alpha-predicate (total number of firepower), which is represented by a circle labeled N [30], shows that this node normalizes on the alpha predicate of the previous layer, as represented by Eq. (34):

$$\bar{\omega}_i = \frac{\omega_i}{\omega_1+\omega_2}, i = 1,2. \quad (34)$$

In the fourth layer there are parameter sets, namely p_i, q_i, and r_i, which are referenced as consequent parameters, as in Eq. (35).

$$O_{4,i} = \bar{\omega}_i f_i = \bar{\omega}_i(p_ix + q_iy + r_i), i = 1,2. \quad (35)$$

The fifth layer is a node that performs a total calculation of all inputs from the previous layer, symbolized by Σ which means the total number, can be written Eq. (36) as follows:

$$O_1^5 = output = \sum_i \bar{\omega}_i f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i} \quad (36)$$

The proposed integrated ANFIS controller structure consists of 3 sub-ANFIS, which is used to determine the motion of each joint angle, which is done alternately for the arms, namely front right (FR), front left (FL), rear right (RR), and rear left (RL) as shown in Fig 8. Each block identified as ACi represents the i-th subANFIS whose structure is as shown

in Fig 9. The whole series of ANFIS sub-ANFIS represents a two / three output neuro-fuzzy network.

Each sub-ANFIS generates a control angle for each connection based on the input it consists of x, and y which is part of the input of the inverse kinematic function parameter. Before fully implemented, controllers need to be trained to achieve better system performance. During training, the error signal in the output can be redistributed and used to adjust the parameters controller.

IV. Result and Discussion

This research contributes to present experiments that show the effectiveness of the controller in simulation and experiments on real robots.

A. Modeling based on Neural Network

In the training of neurons in the hidden layer (layer 1), the differentiated tangent function of type S (tangsig) from BPNN is used based on the D_n (DOF Angle) prediction model of the robot legs as represented by Eq (37), while in the training of neurons in the output layer (layer 2), the Purelin linear function is used.

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{37}$$

The error value of O_n is e_n is the difference between the actual target value of O(n) and the output value of the BPNN prediction under the same sample set, as shown in Eq. (38).

$$e_n = D_n - O_n \tag{38}$$

After doing some trial-and-error experiments with MATLAB to determine the number of hidden layers from 2 to 20, as shown in Fig.10, it can be concluded that the best number of hidden layers is 10, with a total of 12 epochs and six validation checks. The result of output performance as shown in Fig. 11: Training: R=0.95957, Validation: R=0.99825, Test=0.99942, All: R=96679, which means the prediction results are very accurate. From the graphic image, namely Fig. 12 – Fig. 15, the predicted output from BPNN with a target from each leg is very precise, and the error can be said to be zero.

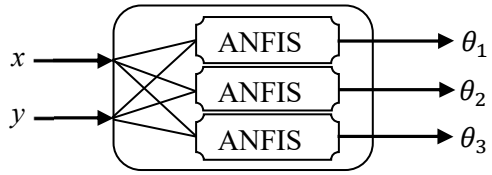


Fig. 8 ANFIS Architecture as IK

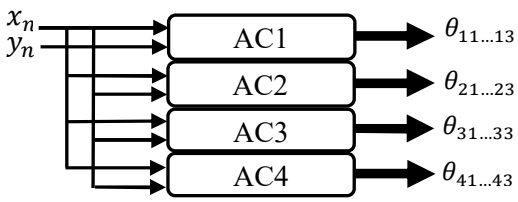


Fig. 9 Scheme of integrated ANFIS Controller (AC)

Fig. 10 BPNN Output Base On **LF-Leg** DoF (Hip-Thigh-Calf)

Fig. 11 BPNN Output Base On **LR-Leg** DoF (Hip-Thigh-Calf)

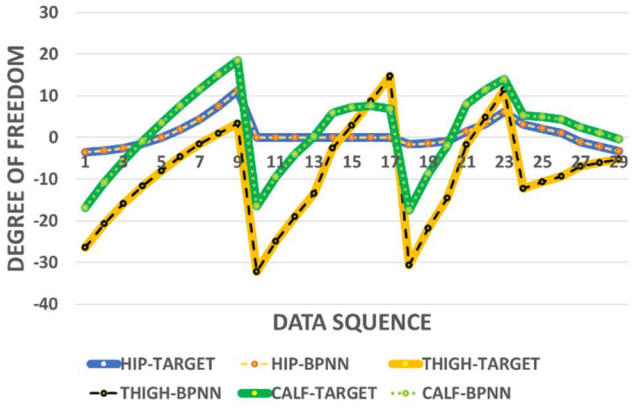


Fig. 12 BPNN Output Base On **RF-Leg** DoF (Hip-Thigh-Calf)

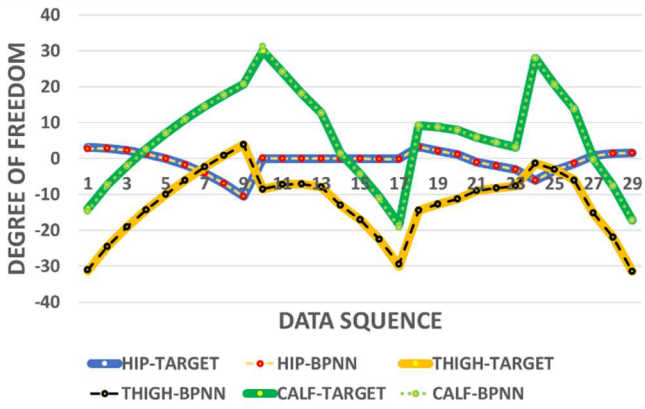


Fig. 13 BPNN Output Base On **RR-Leg** DoF (Hip-Thigh-Calf)

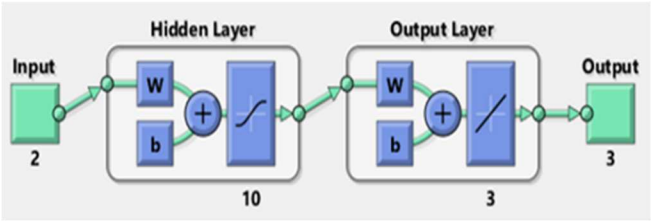


Fig. 14 Number of Hidden layer and Output Layer

BPNN output very precisely and quickly reaches the target being trained, as shown in Fig.12 - Fig.15, thus supporting its use for adjusting the balance of the actual robot body, as shown in Fig. 22. To be able to take advantage of these results, the weight and bias values of the BPNN training are given as in Table 4. Testing the prediction of the movement of the robot's legs can be done using the forward propagation formula as in Eq.(8)–Eq. (12).

TABLE IV
WEIGHTS AND BIASES FROM BPNN TRAINING RESULTS OF RF-LEG

Hidden	IW1	IW2	IW3	IW4	IW5	IW6	IW7	IW8	IW9	IW10
H1	-0.0007	0.0224	0.5546	-0.157	0.345	0.2454	-0.465	-0.07	0.4107	0.0779
H2	-0.805	0.5923	0.5054	0.2154	0.1996	0.3024	-0.135	1.2711	-2.299	0.4426
H3	-1.1304	0.6898	0.7393	0.2608	0.3287	0.3847	-0.188	0.0979	0.0184	-1.784
Bias 1	-3.0688	1.1871	-2.591	-2.352	-0.293	0.0531	2.3672	3.3118	2.4178	6.6962
	LW1	LW2	LW3							
Bias 2	-0.001	0.1588	1.3605							

TABLE V
TRAINING, TARGET AND TESTING DATA OF RF LEG

Num	Training		Target			Testing				
	Pitch	Roll	RF1	RF2	RF3	Pitch	Roll	RF1	RF2	RF3
1	0	20	-3.50	-26.40	-16.90	0	20	-3.50	-26.40	-16.90
2	0	15	-3.20	-20.70	-10.80	0	15	-3.20	-20.70	-10.80
3	0	10	-2.50	-15.90	-5.60	0	10	-2.50	-15.90	-5.60
4	0	5	-1.50	-11.70	-0.90	0	5	-1.50	-11.70	-0.90
5	0	0	0.00	-8.00	3.50	0	0	0.00	-8.00	3.50
6	0	-5	1.90	-4.60	7.60	0	-5	1.90	-4.60	7.60
7	0	-10	4.40	-1.60	11.50	0	-10	4.40	-1.60	11.50
8	0	-15	7.50	1.00	15.10	0	-15	7.50	1.00	15.10
9	0	-20	11.20	3.40	18.50	0	-20	11.20	3.40	18.50
10	20	0	0.00	-32.20	-16.50	20	0	0.00	-32.20	-16.50
11	15	0	0.00	-25.00	-9.40	15	0	0.00	-25.00	-9.40
12	10	0	0.00	-19.00	-4.00	10	0	0.00	-19.00	-4.00
13	5	0	0.00	-13.40	0.20	5	0	0.00	-13.40	0.20
14	-5	0	0.00	-2.60	5.90	-5	0	0.00	-2.60	5.90
15	-10	0	0.00	2.90	7.20	-10	0	0.00	2.90	7.20
16	-15	0	0.00	8.70	7.60	-15	0	0.00	8.70	7.60
17	-20	0	0.00	14.80	6.90	-20	0	0.00	14.80	6.90
18	15	15	-1.60	-30.60	-17.50	15	15	-1.60	-30.60	-17.50
19	10	10	-1.40	-21.80	-8.70	10	10	-1.40	-21.80	-8.70
20	5	5	-0.90	-14.60	-2.00	5	5	-0.90	-14.60	-2.00
21	-5	-5	1.40	-1.70	8.00	-5	-5	1.40	-1.70	8.00
22	-10	-10	3.30	4.80	11.50	-10	-10	3.30	4.80	11.50
23	-15	-15	6.10	11.50	14.00	-15	-15	6.10	11.50	14.00
24	15	-15	3.10	-12.30	5.30	15	-15	3.10		5.30
25	10	-10	2.10	-10.70	5.00	10	-10	2.10	-10.70	5.00
26	5	-5	1.10	-9.30	4.40	5	-5	1.10	-9.30	4.40
27	-5	5	-1.10	-6.90	2.40	-5	5	-1.10	-6.90	2.40
28	-10	10	-2.20	-6.00	1.10	-10	10	-2.20	-6.00	1.10
29	-15	15	-3.30	-5.20	-0.40	-15	15	-3.30	-5.20	-0.40

After ANFIS generates a model, then the model must be tested for validation to the desired model criteria. The purpose of this process is to see how far did ANFIS succeed in doing system modeling. ANFIS validates this model by comparing the output of the data that has been processed learning with other data sets that there is no learning process, other than that the three data sets are independent of each other, so the comparison earlier will result in an "error" that can be used as a measure of the success rate of this model.

The following is a snippet of data for the Right Front (RF) with training, testing, and target data for ANFIS training. In contrast, for the testing data, part of the training data is taken by eliminating the column containing zero angle, as shown in Table 5.

After doing several experiments on the number of MF (Membership Function) it was found that at least 5 MF is needed for each input, so the predictions produced are very close to the target and even the same, with training results RMSE = 0.068233, carried out with only two epochs. Then give the names of the rules, as shown in Fig. 16, which consists of MF 5 for Pitch and MF 5 for Roll, as in Fig 17.

The training results in the prediction of the value of the servo angular movement control according to the input from the IMU sensor. The prediction results are shown in Fig. 18 - Fig.21.

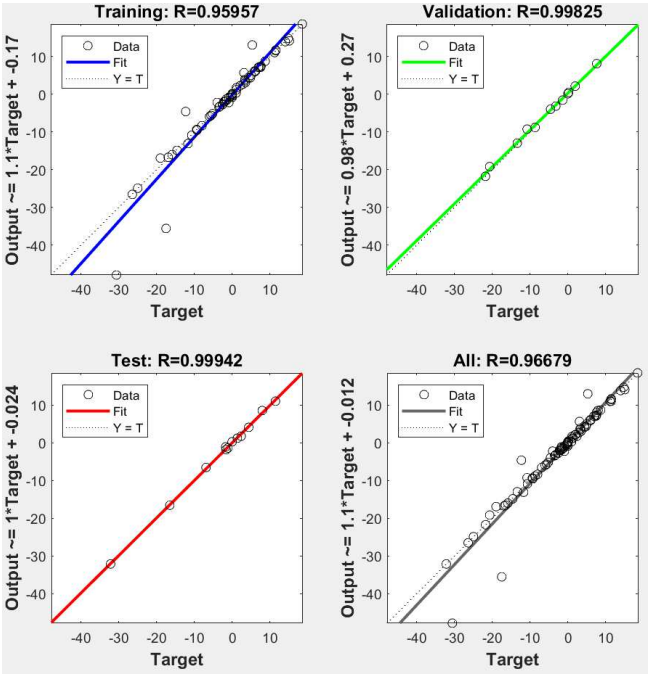


Fig. 15 Output performance from BPNN training

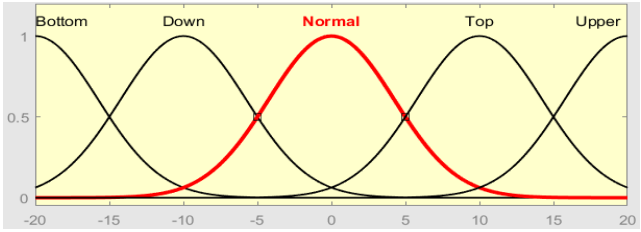


Fig. 16 Membership Function plot for input variable Picth.

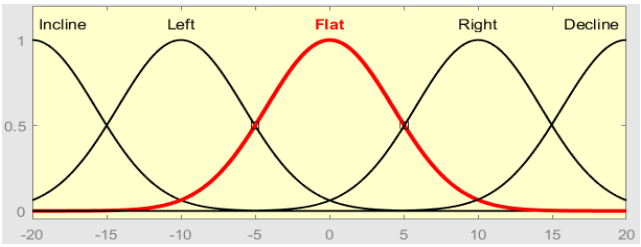


Fig. 17 Membership Function plot for input variable Roll.

After getting the optimal weight, they were then followed by mapping. This process is done in the microcontroller on the quadruped robot. In this stage, the value of the optimal weights is again used as input in the backpropagation learning process. It is just that this learning process does not repeatedly accomplish this because the weight of the weight used is the most optimal, so in a single forward propagation process, the value of the resulting output is close to the target value.

B. Modeling based on ANFIS

In this research, the ANFIS architecture is used, which is designed for particular purposes so that it can produce three outputs from the five inputs given from the angle data from the measurement results of the robot's balance movement after being given the slope of the footing surface. ANFIS then trains the data to produce the produces the appropriate output based on the input given, which is in the form of input data, as shown in Table 3. above.

According to the input data, there are two pitch and roll angles, the slope of the robot's body, and the twelve angles of the four legs (hip, thigh and calf), so the input and output nodes used are two and twelve as shown in Fig. 9.

ANFIS output is exact and quickly reaches the target being trained, as shown in Fig. 18 - Fig.21. This is because ANFIS uses 25 rules, as shown in Table 6, obtained from the results of the MF 5x5 setting. So that with these 25 rules can be applied to low-level programming (machine language) to predict the movement of the robot's legs based on the targets from the IMU input, namely pitch and roll, thus supporting its use to adjust the actual balance of the robot's body as shown in Fig. 22. The following is an overview of the overall output of ANFIS when measuring the balance of the robot body based on input from the IMU sensor, namely pitch and roll as shown in Fig. 23.

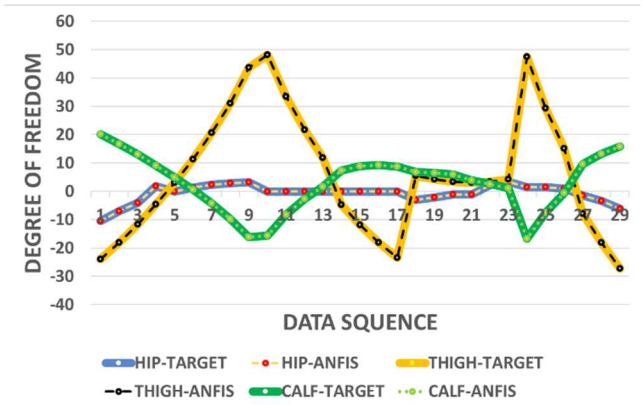


Fig. 18 ANFIS Output Base On LF-Leg DoF (Hip-Thigh-Calf)

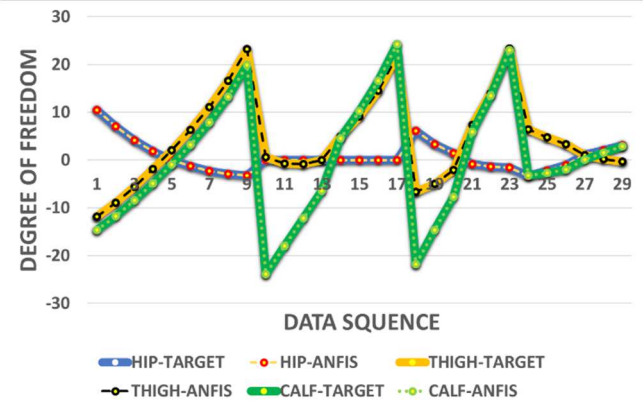


Fig. 19 ANFIS Output Base On LR-Leg DoF (Hip-Thigh-Calf)

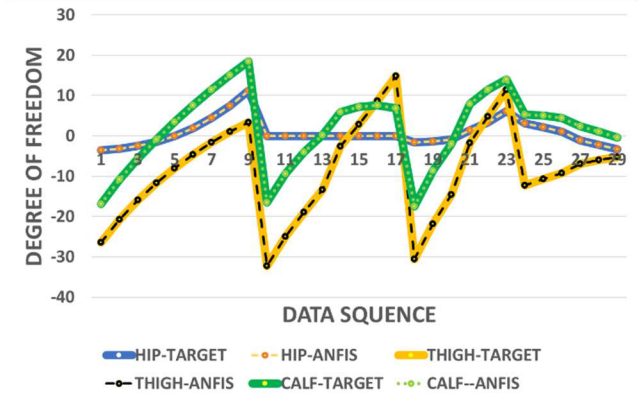


Fig. 20 ANFIS Output Base On RF-Leg DoF (Hip-Thigh-Calf)

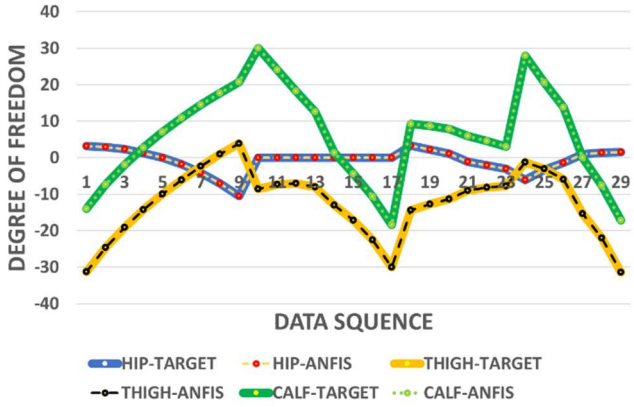


Fig. 21 ANFIS Output Base On RR-Leg DoF (Hip-Thigh-Calf)

Fig. 22 Implementation of output from ANFIS

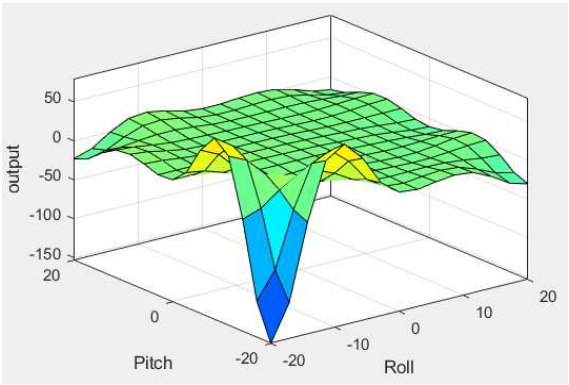


Fig. 23 Self balancing measurement base on Pitch and Roll

IV. Conclusion

Both methods have been tested with the same input, with various choices of parameters that best suit the method used, so that it can produce the desired output with perfect results, namely with an error of 0. The backpropagation method (BPNN) can produce output per the number of 12 targets at once by giving a total weight of 120 to produce 12 targets with ten hidden layers, ten input biases, and one output bias. The ANFIS method only produces one output, so several ANFIS must be used to handle 12 targets, 3 ANFIS handles each leg, so there are a total of 12 ANFIS. BPNN relies on multiplication and addition arithmetic calculations so that during implementation, it is constrained by speed and accuracy in predicting targets based on input from IMU pitch and roll data. ANFIS relies on speed with the help of pre-arranged rules so that speed and accuracy can be achieved properly. ANFIS with 5x5 MF on each input will produce 25 rules, so the achievement of target prediction is easy to do when implemented on a quadruped robot machine. Future research is to develop a reliable method like Zero Moment Point (ZMP) [32] [33] [34] [35] that can control the movement of the robot's legs to be able to walk across irregular and sloping terrain combined with the BPNN or ANFIS method as a balanced guard for the robot's body so that it does not fall.

TABLE VI
ANFIS RULES FOR LOW-LEVEL PROGRAMMING

Rule of MF5x5 from Pitch and Roll Base on RF-LEG
1. if (Pitch is Bottom) and (Roll is Incline) then (output is out1mf1) (1)
2. if (Pitch is Bottom) and (Roll is Left) then (output is out1mf2) (1)
3. if (Pitch is Bottom) and (Roll is Flat) then (output is out1mf3) (1)
4. if (Pitch is Bottom) and (Roll is Right) then (output is out1mf4) (1)
5. if (Pitch is Bottom) and (Roll is Decline) then (output is out1mf5) (1)
6. if (Pitch is Down) and (Roll is Incline) then (output is out1mf6) (1)
7. if (Pitch is Down) and (Roll is Left) then (output is out1mf7) (1)
8. if (Pitch is Down) and (Roll is Flat) then (output is out1mf8) (1)
9. if (Pitch is Down) and (Roll is Right) then (output is out1mf9) (1)
10. if (Pitch is Down) and (Roll is Decline) then (output is out1mf10) (1)
11. if (Pitch is Normal) and (Roll is Incline) then (output is out1mf11) (1)
12. if (Pitch is Normal) and (Roll is Left) then (output is out1mf12) (1)
13. if (Pitch is Normal) and (Roll is Flat) then (output is out1mf13) (1)
14. if (Pitch is Normal) and (Roll is Right) then (output is out1mf14) (1)
15. if (Pitch is Normal) and (Roll is Decline) then (output is out1mf15) (1)
16. if (Pitch is Top) and (Roll is Incline) then (output is out1mf16) (1)
17. if (Pitch is Top) and (Roll is Left) then (output is out1mf17) (1)
18. if (Pitch is Top) and (Roll is Flat) then (output is out1mf18) (1)
19. if (Pitch is Top) and (Roll is Right) then (output is out1mf19) (1)
20. if (Pitch is Top) and (Roll is Decline) then (output is out1mf20) (1)
21. if (Pitch is Upper) and (Roll is Incline) then (output is out1mf21) (1)
22. if (Pitch is Upper) and (Roll is Left) then (output is out1mf22) (1)
23. if (Pitch is Upper) and (Roll is Flat) then (output is out1mf23) (1)
24. if (Pitch is Upper) and (Roll is Right) then (output is out1mf24) (1)
25. if (Pitch is Upper) and (Roll is Decline) then (output is out1mf25) (1)

Nomenclature

T_2^3, T_1^2, T_0^1 Transfer function.
X_i input i^{th}
O_i^{L1} hidden layer input
O_i^{L2} hidden layer output
W weight
Δw delta weight
$\Delta bias$ delta bias
x, y input vector.
$O_{n,i}$ neuron in of layer n, associated to input i.
$A_i(x), \mu B_i(y)$ the membership function.
a_i, b_i, and c_i premise parameters.

Greek letters
μ membership function.
Π the product of the incoming signal μ.
ω the product output.
θ angles leg

Subscripts
i node.

Acknowledgment

We thank the Indonesian Dissertation Scholarship Program Manager. We are grateful to Indonesian Ministry of Finance, which has provided scholarships to complete my doctoral program. We are obliged to Boris ALP for sharing the quadruped robot on the page https://grabcad.com/library/diy-quadruped-robot-1.

REFERENCES

[1] J. Kim, T. Kang, D. Song and S.-J. Yi, "PAWDQ: A 3D Printed, Open Source, Low Cost Dynamic Quadruped," in *18th International Conference on Ubiquitous Robots (UR)*, Gangneung-si, Gangwon-do, Korea, July 12-14, 2021.

[2] Y. Shi, S. Li, M. Guo, Y. Yang, D. Xia and X. Luo, "Structural Design, Simulation and Experiment of Quadruped Robot," *Applied Sciences,* vol. 11, no. 22, p. 10705, 2021.

[3] H. Sun, T. Fu, Y. Ling and C. He, "Adaptive Quadruped Balance Control for Dynamic Environments Using Maximum-Entropy Reinforcement Learning," *Sensors,* vol. 21, no. 17, p. 5907, 2021.

[4] M. A. Şen, B. Veli and K. Mete, "Three Degree Of Freedom Leg Design For Quadruped Robots And Fractional Order Pid (Piλdμ) Based Control," *Konya Journal of Engineering Sciences,* vol. 8, no. 2, pp. 237-247, 2020.

[5] S. Wasista, H. Tjandrasa and W. Wibisono, "Swing Trajectory Model for the New Design of Quadruped Robot Using V-REP Simulator," in *International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM)*, Surabaya, 2020.

[6] Y. H. Lee, Y. H. Lee, H. Lee, H. Kang, L. T. Phan, S. Jin, Y. B. Kim, D.-Y. Seok, S. Y. Lee, H. Moon, J. C. Koo and a. H. R. Choi, "Force-controllable Quadruped Robot System with Capacitive-type Joint Torque Sensor," in *International Conference on Robotics and Automation (ICRA)*, Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019.

[7] Y. Jia, X. Luo, B. Han, G. Liang, J. Zhao and Y. Zhao, "Stability Criterion for Dynamic Gaits of Quadruped Robot," *Applied Sciences,* vol. 8, no. 12, November 2018.

[8] J.-S. R. Jang, "ANFIS : Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions On Systems, Man, and Cybernetics,* vol. 23, no. 3, pp. 665-685, 1993.

[9] J.-H. Qin, J. Luo, K.-C. Chuang and T.-S. Lan, "Stable Balance Adjustment Structure of the Quadruped Robot Based on the Bionic Lateral Swing Posture," *Mathematical Problems in Engineering,* vol. 2020, 2020.

[10] J. Cui, Z. Li, Y. Kuang and H. Cheng, "Standing balance maintenance by virtual suspension model control for legged robot," *Advances in Mechanical Engineering Robotics: Intelligence, Learning, and Control - Research Article,* vol. 12, no. 9, pp. 1-13, 2020.

[11] V. Bakırcıoğlu, M. A. Şen and M. Kalyoncu, "Adaptive Neural-Network Based Fuzzy Logic (ANFIS) Based Trajectory Controller Design for One Leg of a Quadruped Robot," 2016.

[12] X. Zhu, J. Wan, C. Zhou and W. Xu, "A composite robust reactive control strategy for quadruped robot under external push disturbance," *Computers and Electrical Engineering,* vol. 91, no. : 107027, pp. 1-17, 2021.

[13] T. Sun, Z. Dai and P. Manoonpong, "Distributed-force-feedback-based reflex with online learning for adaptive quadruped motor control," *Neural Networks 142 ,* p. 410–427, 2021.

[14] I. Gonzalez-Luchena, A. Gonzalez-Rodriguez, A. Gonzalez-Rodriguez, C. Adame-Sanchez and F. Castillo-Garcia, "A new algorithm to maintain lateral stabilization during the running gait of a quadruped robot," *Robotics and Autonomous Systems 83 ,* p. 57–72, 2016.

[15] P. Biswal, K. Mohanty and Prases, "Development of quadruped walking robots: A review," *Ain Shams Engineering Journal 12,* p. 2017–2031, 2021.

[16] S. Tripathy and S. Gaur, "Rough terrain quadruped robot- BigDog," *Materials Today: Proceedings,* 2021.

[17] C. Gonzalez, V. Barasuol, M. Frigerio, R. Featherstone, D. G. Caldwell and C. Semini, "Line Walking and Balancing for Legged Robots with Point Feet," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA (Virtual), October 2020.

[18] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, J. Lee, V. Tsounis, V. Koltun and M. Hutter, "Learning Agile and Dynamic Motor Skills for Legged Robots," *Science Robotics,* vol. 4, no. 26, pp. 1-20, 2019.

[19] W. Sun, X. Tian, Y. Song, B. Pang, X. Yuan and Q. Xu, "Balance Control of a Quadruped Robot Based on Foot Fall Adjustment," *Applied Sciences ,* vol. 12, no. 5, pp. 1-14, 2521, February 2022.

[20] M. W. Spong, S. Hutchinson and M. Vidyasagar, Robot Dynamics and control. 1st ed., New York: John Wiley & Sons, Inc., 2006.

[21] M. Ning, J. Yang, Z. Zhang, J. Li, Z. Wang, L. Wei and P. Feng, "Method of Changing Running Direction of Cheetah-Inspired Quadruped Robot," *Sensors,* vol. 24, no. 22, 9601, 2022.

[22] X. Zheng, Y. Zheng, Y. Shuai, J. Yang, S. Yang and Y. Tian, "Kinematics analysis and trajectory planning of 6-DOF robot," in *Information Technology, Networking, Electronic and Automation Control Conference (ITNEC 2019)*, Chengdu, China, 2019.

[23] M. Chen, H. Chen, X. Wang, J. Yu and Y. Zhang, "Design and Control of a Novel Single Leg Structure of Electrically Driven Quadruped Robot," *Mathematical Problems in Engineering,* vol. 2020, no. 21, pp. 1-12, May 2020.

[24] S. Kucuk and Z. Bingul, "Robot Kinematics: Forward and Inverse Kinematics," *Industrial-Robotics-Theory-Modelling-Control,* vol. 285, no. 8, p. 964, December 2006.

[25] Setiawardhana, R. Dikairono, D. Purwanto and T. Arief Sardjono, "Ball Position Estimation in Goal Keeper Robots Using Neural Network," *International Review of Automatic Control (IREACO),* vol. 12, no. 1, pp. 38-47, January 2020.

[26] R. Gao, "Inverse kinematics solution of Robotics based on neural network algorithms," *Journal of Ambient Intelligence and Humanized Computing ,* March 2020.

[27] A. Alazzam and T. Tashtoush, "Lead-Free Solder Reliability Modeling Using Adaptive Neuro-Fuzzy Inference System (ANFIS)," *Jordan Journal of Mechanical and Industrial Engineering,* vol. 15, no. 2, pp. 181-189, 2021.

[28] T. Ibarra-Pérez, J. Manuel Ortiz-Rodríguez, F. Olivera-Domingo, H. A. Guerrero-Osuna, H. Gamboa-Rosales and M. d. R. Martínez-Blanco, "A Novel Inverse Kinematic Solution of a Six-DOF Robot Using Neural Networks Based on the Taguchi Optimization Technique," *Applied Sciences,* vol. 12, no. 19, pp. 20, 9512, Sep 2022

[29] S. Wasista, H. Tjandrasa and W. Wibisono, "Quadruped Robot Control Base on Adaptive Neuro-Fuzzy Inference System With V-REP Simulator," in *3rd International Conference on Information and Communications Technology (ICOIACT)*, Yogyakarta, 2020.

[30] D. Karaboga and E. Kaya, "Adaptive Network Based Fuzzy Inference System (ANFIS) Training Approaches: A Comprehensive Survey," *Artificial Intelligence Review, An International Science and Engineering Journal,* vol. 52, no. 4, pp. 2263-2293, Dec. 01, 2019.

[31] B. Sarwar, I. S. Bajwa, N. Jamil, S. Ramzan and N. Sarwar, "An Intelligent Fire Warning Application Using IoT and an Adaptive Neuro-Fuzzy Inference System," *Sensors,* vol. 19, no. 14, pp. 1-18, 3150, July 2019.

[32] H. Khalili, "Path planning of a quadruped robot on a flat surface using ZMP for stability," *Journal of Research in Science, Engineering and Technology,* vol. 7, no. 2, pp. 6-20, February 2019.

[33] Y. d. Viragh, M. Bjelonic, C. D. Bellicoso, F. Jenelten and M. Hutter, "Trajectory Optimization for Wheeled-Legged Quadrupedal Robots Using Linearized ZMP Constraints," *IEEE Robotics And Automation Letters,* vol. 4, no. 2, pp. 1633-1640, April 2019.

[34] X. Zhu, M. Wang, X. Ruan, L. Chen, T. Ji and X. Liu, "Adaptive Motion Skill Learning of Quadruped Robot on Slopes Based on Augmented Random Search Algorithm," *Electronics,* vol. 11, no. 6, 842, pp. 1-15, 2022.

[35] X. Meng, W. Liu, L. Tang, Z. Lu, H. Lin and J. Fang, "Trot Gait Stability Control of Small Quadruped Robot Based on MPC and ZMP Methods," *Processes ,* vol. 11 , no. 1, pp. 1-14, 252, January 2023.