Received: 15 July 2022 | *Revised:* 29 August 2022 | *Accepted:* 29 August 2022
DOI: 10.37917/ijeee.18.2.15 *Early View* | *December 2022*

Open Access

Iraqi Journal for Electrical and Electronic Engineering
Original Article

Learning the Quadruped Robot by Reinforcement Learning (RL)

A. A. Issa*, A. A. Aldair
Electrical Engineering Department, University of Basrah, Basrah, Iraq

Correspondence
* A. A. Issa
Electrical Engineering Department,
University of Basrah, Basrah, Iraq
Email: engpg.ali.ashoor@uobasrah.edu.iq

Abstract

In this paper, a simulation was utilized to create and test the suggested controller and to investigate the ability of a quadruped robot based on the SimScape-Multibody toolbox, with PID controllers and deep deterministic policy gradient DDPG Reinforcement learning (RL) techniques. A quadruped robot has been simulated using three different scenarios based on two methods to control its movement, namely PID and DDPG. Instead of using two links per leg, the quadruped robot was constructed with three links per leg, to maximize movement versatility. The quadruped robot-built architecture uses twelve servomotors, three per leg, and 12-PID controllers in total for each servomotor. By utilizing the SimScape-Multibody toolbox, the quadruped robot can build without needing to use the mathematical model. By varying the walking robot's carrying load, the robustness of the developed controller is investigated. Firstly, the walking robot is designed with an open loop system and the result shows that the robot falls at starting of the simulation. Secondly, auto-tuning are used to find the optimal parameter like (KP, KI and KD) of PID controllers and resulting shows the robot can walk in a straight line. Finally, DDPG reinforcement learning is proposed to generate and improve the walking motion of the quadruped robot, and the results show that the behaviour of the walking robot has been improved compared with the previous cases, Also, the results produced when RL is employed instead of PID controllers are better.

KEYWORDS: Quadruped Robot, Servo Motors, PID Controllers, Reinforcement Learning designer.

I. INTRODUCTION

Legged machines have gained increased academic interest in recent years because of too many possible benefits over traditional wheeled or tracked robots in specific applications such as navigating tough terrain, and moving and interacting in human situations [1]. The major challenge for a walking robot is to maintain steady mobility. One of the most well-known statically stable gait stability tests is the mass center criteria, which states that a robot is statically stable if the projection of its center of mass **CoM** onto a horizontal plane falls within the support grid [2]. Various control methods, such as Model-Based Algorithm (MBA) based on the feedback linearization approach, [3], robust control, [4], and so on, have been developed and implemented to monitor the intended trajectories of walking robots.

The modelling system of a quadruped robot with numerous legs has a significant degree of nonlinearity and uncertainty. As a result, a strong controller is necessary to steer the quadruped robot's mobility. The cost of multiple-leg walking robots is undeniably high. As a result, before purchasing a physical robot, a dependable controller must be appropriately constructed and its performance thoroughly researched. Researchers can use simulation programs to analyze and predict a robot's overall performance and enhance its process route planning. For these reasons, using a simulator program is helpful since it may save time and money. Various programs are used to analyze dynamic and kinematic properties of walking robot systems, to offline programming, and to build various control methods [5].

To design and simulate the robot we need two strategies, the first technique is to create the quadruped robot system and build the control system using specific libraries, or toolboxes, only simulation results may be analyzed using this method, and the visible motion of a robotic-system cannot be exhibited, so we need to second technique is used to show the motion results of the robot [5].

In this paper, The SimScape-Multibody toolbox simulator is used to model the movements of certain sorts of robots that execute tasks including path planning, line-following, and barrier avoiding is utilized to simulate and display the motion of a quadruped robot using a suggested controller instead of using two programs to design and simulate of the robot.in this paper, the robot is designed with four legs, where each leg had three joints instead of two joints to increase the

freedom of the robot, also uses one servo-motor for each joint so, the total number of motors is 12. Firstly, in this paper, the quadruped robot is designed with an open-loop system and the results show that the robot falls without completing any step in the forward direction. Secondly, a 12-PID controller is used for each servomotor, to improve the walking-robot motion. Finally, a reinforcement learning strategy is used to control and improve the quadruped robot motion. The efficacy and durability of the developed controller are investigated by varying the quadruped robot's carrying weight.

The simulation results show that the overall quadruped robot performance is improved and the reinforcement learning results are better compared with closed-loop with PID controller and open loop cases.

The rest of this paper is organized as follows: section 2. The explanation of modelling the quadruped robot with Semescap-Multibody in the Matlab program. The control design of a quadruped robot was explained in section 3, the section is divided into three subsections subsec.1 explains the open loop system, subsec. 2 explains the closed loop system with PID controllers, and the final subsection explains the DDPG reinforcement learning design. The simulation results were explanted in section 4, and this section was divided into two subsections one for the results of the closed-loop control system, and the other for the DDPG reinforcement learning method. In section 5 the comparison results are explained. The conclusions were demonstrated in section 6.

II. MODELLING THE QUADRUPED ROBOT BY SEMSCAPE MULTIBODY TOOLBOX

In this section, we use the Matlab SimScap-Multibody toolbox to modulate and simulate the quadruped robot. The SimScape toolbox comprises several libraries and Simulink blocks that may be used to design any robot architecture, such as mobile robots, robotic manipulators with various amounts of links, and leg robots. This allows the user to create both the mechanical and control systems in the same environment. Various analysis modes and powerful visualization tools enable users with minimum mechanical knowledge to simulate complicated dynamical systems. It also includes modelling and control configuration to assist people in obtaining essential simulated results and displaying possible robot movements. [5, 6]. The following Fig. 1 shows a Visual display of a quadruped robot in Matlab SimScape Multibody.

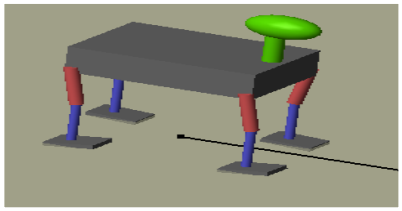


Fig. 1: Visual display of quadruped robot in SimScape-Multibody.

Figure 1 shows the 3-D structure of a quadruped robot that is designed with a Matlab simscape multibody. Where the robot contains a torso and four legs and each leg has three joints.

Figure 1 shows the quadruped robot that is designed in Matlab Simscape, so Fig. 2 shows the main components of a quadruped robot system in Simscape Matlab.

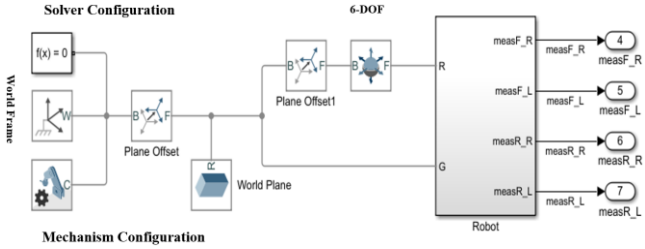


Fig. 2: Main components of a four-leg walking robot.

Figure 2 shows that there are a few blocks that are linked together. Begin with the World Frame block, which must connect a Solver Configuration and a Mechanism Configuration block. The solver block is required for all Simescape models, and the mechanism block is applied to the complete machine to establish parameters such as gravity direction and amplitude. Fig. 3 shows the Robot subsystem in Fig. 2.

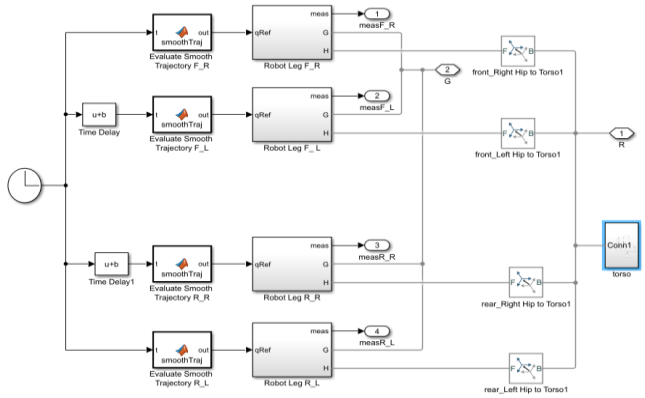


Fig. 3: Robot subsystem that contains four-leg.

Where Fig. 3 shows the walking robot system that includes four legs with three links in each leg. Fig. 4 shows the leg subsystem of a quadruped robot that is shown in Fig. 3.

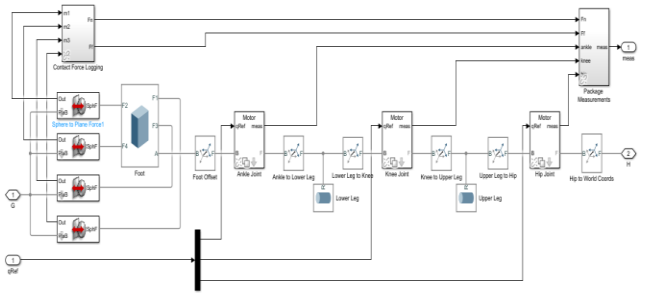


Fig. 4: One-leg components of a quadruped robot (leg subsystem).

Figure 4 show that the leg of the robot contains three joint where each joint contain one servo-motor and one PID controller, Fig. 5 shows the joint subsystem in Fig. 4.

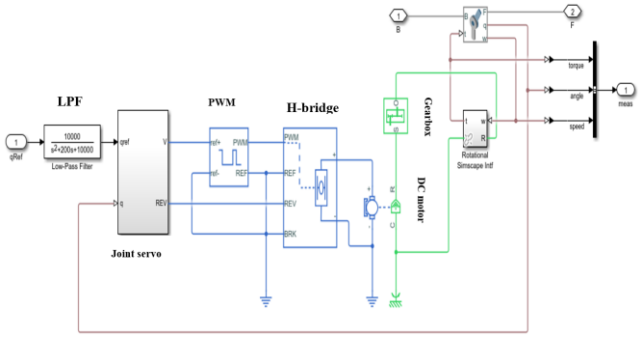


Fig. 5: One joint subsystem of a quadruped robot in.

Figure 5 shows the components of one joint of a quadruped robot, where the joint is controlled by a servomotor and the PID controller in the joint servo subsystem Fig. 6 shows the PID controller system controls the servomotor.

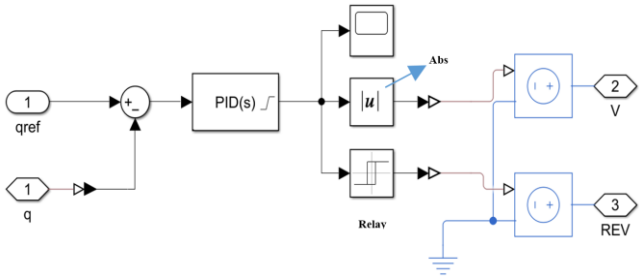


Fig. 6: PID controller system.

Table 1 describes the functionalities of most Simulink blocks used in Fig. 2 to Fig. 6, to help the reader understand them.

TABLE 1
DESCRIPTION OF THE SEMESCAP-MULTIBODY BLOCKS.

Blocks	Name	Descriptions
	Solver-Configuration	Sets the simulation's setup values.
	World-Frame	The reference point of the mechanical model is built. The World's Framework
	Mechanism-Configuration	Initial mechanical and simulation - parameter setup
	Rotational - Joint	A rotational joint is used to interpret motion at angles between the actuators and the fixed base.
	Solid-Block	Solid blocks offer-soled characteristics.
	PS-converter	Converter-Simulink

III- CONTROL SYSTEM OF A QUADRUPED ROBOT

Control methods are created in this paper for a quadruped robot with 4-legs, and 3- links for each leg, and each link is driven by a servomotor. To connect the connections, the revolving joint is employed. As illustrated in Fig. 7, the revolve joint has 2- inputs and 3- outputs.

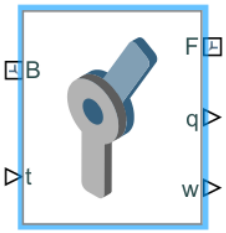
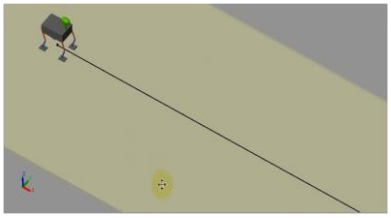


Fig. 7**:** The Revolve joint

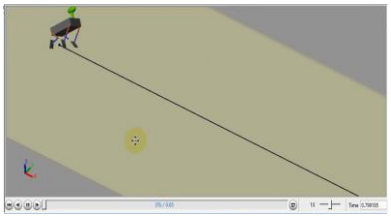
Inputs **B** and t indicate the control signal given to a servomotor's torque and the port that links the current link to the previous link respectively. first output **F** represents the connecting point between the current and upcoming links. The second output **q** reflects the current link's angle and the last output **w** measures speed. A servomotor may be operated by sending a series of variable width electrical pulses over the line. The voltage provided to the servo motor is modified by varying the minimum and maximum duration of the pulses, resulting in a change in motor speed owing to voltage change. When the voltage changes, the current varies, which causes the torque to alter. Therefore, in this paper, we apply three cases to the robot first when the open-loop system is applied with a servomotor, second, when a closed-loop system with PID controllers is chosen, and finally, Reinforcement learning techniques are applied to the quadruped robot.

A. open loop system.

In this situation, the quadruped robot is created with no controls and tests if the robot can finish the move on the road steadily. However, the data demonstrate that the quadruped robot falls and lost its motion stability after the simulation began. Fig. 8a, depicts the quadruped robot in its original posture before the simulation began. Fig. 8b, depicts the quadruped robot's inability to walk without going forward.



(a)



(b)

Fig. 8: (**a)** Quadruped robot stand at the initial position (**b)** quadruped robot falls after the simulation starts.

B. Closed loop control system.

In case one of the open loop system the quadruped robot falls immediately after the simulation begins, so the robot needs to control to improve the efficiency of the quadruped robot motion, so in this paper, we propose two controllers design, a closed-loop control system with PID controllers and DDPG reinforcement learning.

Because the quadruped robot model contains twelve joint, three per leg, and there is a servomotor for each joint totally of twelve servomotors, so we need twelve PID controllers, one for each servomotor. Fig. 6 shows the block diagram of the PID controller system for one joint.

Where the best parameters of PID controllers were obtained with PID tuner by the auto-tuning method.

C. PID controllers.

The PID term sign to the initial letter of the multiple term names that comprise the conventional 3-term controller. In the controller, these are P for the proportional term, I for the integral term, and D for the derivative term. Appropriate adjustment of these parameters will enhance plant performance, decrease overshoot, remove the steady-state error, and promote system stability. The biggest issue with the basic controller is determining the proper PID gains. When plant characteristics and operating circumstances vary, the controller may not offer the needed control performance if fixed gains are used. As a result, tuning must be undertaken to ensure that the controller can deal with fluctuations in the plant [7]. The following Fig. 9 shows the block diagram of the PID controller.

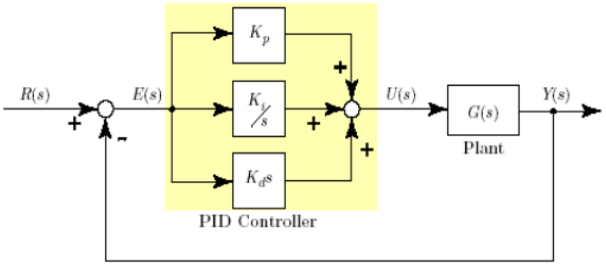


Fig. 9: Block diagram of PID controller [9].

The output of the PID controller is given as the following.

$$u(t) = Kp\mathbf{e}(t) + Ki\int \mathbf{e}(\mathbf{t})dt + Kd\dot{\mathbf{e}}(t) \qquad (1)$$

Where *u* is the control signal, and *e* is the error value. Then the transfer function.

PID controller tuning approaches such as Ziegler-Nichols rules, Cohen-Coon rules, and others have been developed. Because they give, straightforward tuning criteria for determining the PID parameters, these approaches are used directly [8-10]. In this paper, the auto-tuning tool with a PID tuner in Matlab extracts the best parameters of the PID controller.

D. Reinforcement learning (RL) of a walking robot.

Reinforcement learning (RL) is the study of how an agent interacts with its environment to develop a policy that optimizes predicted cumulative rewards for a task. Recently, RL has seen a surge in attention and focus as a consequence of promising discoveries in fields such as operating continuous systems in robotics, and playing **GO** game (**Go** is an **abstract strategy board game** for two players in which the aim is to surround more territory than the opponent), Atari, and professional video games. RL is utilized to make a robot learn to walk in a controlled environment. The primary goal of this paper is to learn a quadruped robot to walk by creating a model in simscpe Matlab. This work explains how to learn a quadruped robot to walk using a deep deterministic policy gradient (DDPG) agent. (DDPG) an algorithm is a model-free, online, off-policy reinforcement learning method. Deep Deterministic Q-learning and Policy gradients are combined in the reinforcement learning method known as Policy Gradient (DDPG). As an actor-critic technique, DDPG uses two models: the actor and the critic. Instead of producing a probability distribution of actions, the actor is a policy network that receives the state as input and outputs the precise action (continuous). A Q-value network that receives input from state and action and outputs the Q-value is the critic. The DDPG is a method that is "off" policy. The "deterministic" in DDPG refers to the fact that the actor computes the action directly rather than a probability distribution over actions. DDPG is employed in the continuous action setting [11-13].

The walking robot can be learned by trial and error using reinforcement learning (RL) methods. The model and everything outside the learning process is referred to as the environment in reinforcement learning, and the reinforcement learning (RL) environment for this study is a quadruped robot. The goal of the training is for the robot to walk in a straight path with as little control effort as feasible. With our environment model in place, we have picked a reinforcement-learning method, which will decide how to actuate the joints depending on data from the environment. The agent learns to make the proper decisions for successful walking after much trial and error. The agent makes decisions based on observations, where observations provide the agent with information about the condition of the environment, such as location, velocities, forces, and any other signal that we choose to depend on what the agent needs to know to walk. The following Fig. 10 shows the quadruped robot system with reinforcement learning RL toolbox.

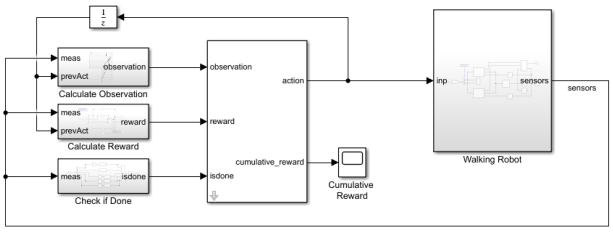


Fig. 10: Quadruped robot system with RL blocks.

The following three points explain the observations information condition that is delivered to the agent to make a decision action, point two shows action details from the agent that applied to the robot system, and point three explain the reward details that is getting from the environment to help the agent to make the best action.

1) Observations.

The environment (robot system) provides the following observations to the agent.

- y and z body position.
- x, y and z body velocity.
- Body angles
- Joint angles.
- The Contact force between feet and the ground.
- The commanded torque that was commanded in the previous time step.
- Position of each joint.

In addition, there is some assumption for the walking robot to make help speed up the learning of the robot. The first assumption is, that the legs of the robot are straight, and the second assumption is the ankles of all legs are flat in the neutral 0 rad position.

The foot contact is modelled using the Spatial Contact force (SimScape-Multibody) block. The agent may operate three particular joints on both legs of the quadruped robot by delivering torque signals to three joints.

2) Actions.

After the training end, the agent creates twelve actions, where the robot has twelve joints, three per leg. And these actions represent the torque that is applied to the robot joints, Ankle, Knee and hip.

3) Reward.

The agent receives the following reward at each time step during training. Because we want the robot to walk in a straight line forward, the first reward is added forward velocity (Vx) in the x-direction (which means that the robot moving in forward in a straight line).

The second and third rewards apply a penalty on the **y** and **z** dimension displacement, which makes sure the robot, does not deviate too far from the line or fall dawn.

The fourth reward is applying a penalty on the joint effort or joint torque, the reason for this penalty is to reject the aggressive motion.

The final reward (positive reward) is a duration reward to prevent a very common local minimum where the robot very early on learns to fall forward. The following Fig. 11 shows the block diagram of overall rewards in the RL model of the robot.

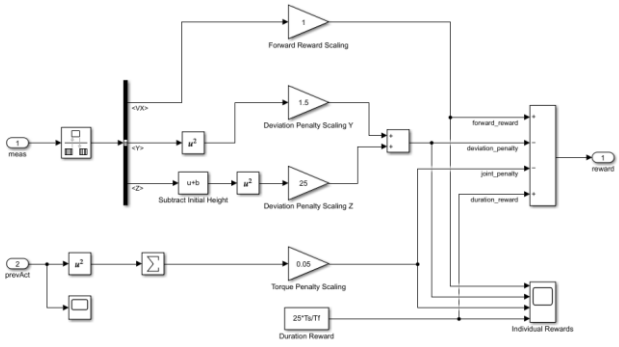


Fig. 11: Block diagram of reward function in RL model.

IV. Simulation Results

In this section, the simulation results were divided into two subsections one for a closed-loop control system with the PID controller's method and the other for the DDPG reinforcement learning method.

A. Simulation results of the closed-loop control system.

The auto-tuning tool with Matlab Simulink is used to find the optimal parameters of the PID controller i.e. KP, KI and KD the following table 2 shows the best parameters that are found from the auto-tuning method in the PID tuner tool.

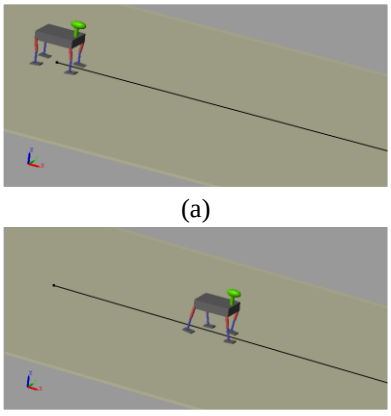
TABLE 2
PARAMETERS OF PID CONTROLLERS (AUTO-TUNING METHOD).

PID	Kp	Ki	Kd
Controller 1	28.64507	634.67449	-0.342715
Controller 2	22.05537	452.04501	0.0790049
Controller 3	18.55109	251.79850	0.0458747
Controller 4	23.10922	643.77684	-0.256986
Controller 5	19.29109	375.69966	0.0304624
Controller 6	11.41438	171.16699	0.0526633
Controller 7	23.79493	216.2899	-0.01950
Controller 8	22.9857	401.729	-0.06874
Controller 9	46.9703	371.527	-0.52653
Controller 10	25.95642	580.8400	-0.118329
Controller 11	18.23887	314.4904	-0.12582
Controller 12	12.90897	156.2225	0.182405

The robustness of the designed controller is studied by changing the carried weight of the quadruped robot Simulation results can be divided into three cases, the first is when the robot is without any carried weight. The second case is when the robot is loaded with a 1-kg weight, and finally when a 2-kg weight is added to the robot.

Case One: A quadruped robot without load.

Figs. 12 a and b show the robot simulation results without load at stars and after a few meters respectively.



(a)

(b)

Fig. 12 (**a)** Walking robot with no load at initial position (**b)** walking robot with no load after a few meters.

Because a walking robot has four legs, each with 3-links, demonstrating the performance of each joint is difficult. As a result, this piece only depicts the performance of the front

right leg. Fig. 13 shows the PID control efforts for the front right leg of the quadruped robot.

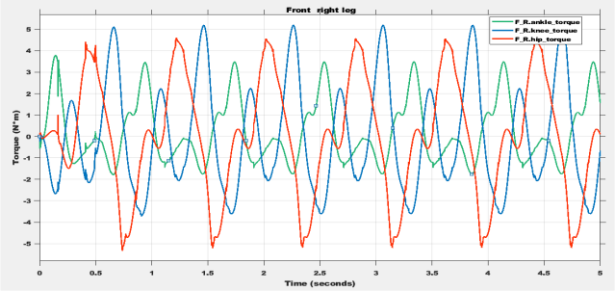


Fig. 13: Front right leg control efforts of the walking robot with no load.

Also, the comparison between the desired angle and the act. the 3-joints ankle, knee and hip are shown in Figs. 14-16.

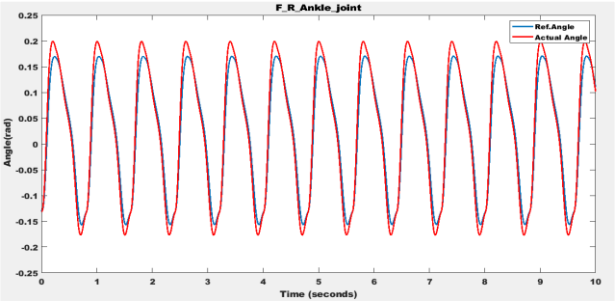


Fig. 14: Desired angle and actual angle comparisons for the hip joint.

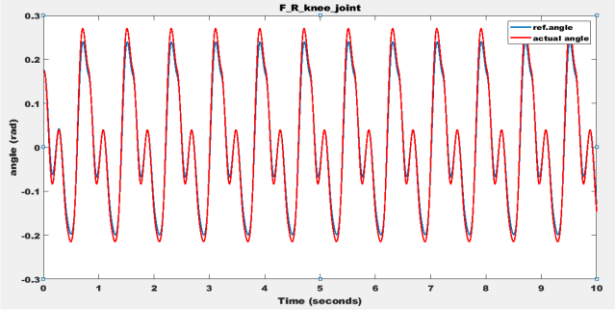


Fig. 15: Desired angle and actual angle comparisons for the knee joint.

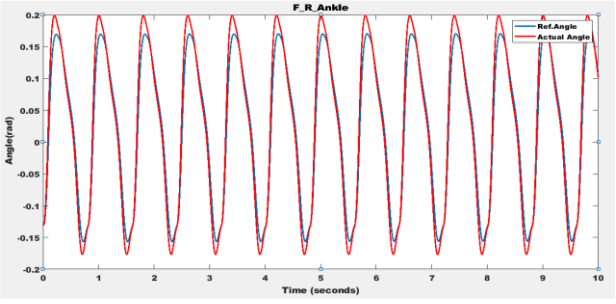
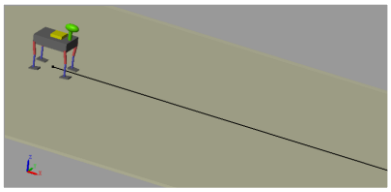


Fig. 16: Desired angle and actual angle comparisons of the ankle joint.

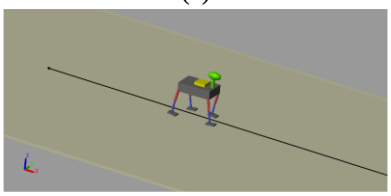
From the above results of the case, one with a closed-loop control system with PID controllers show that the walking of the robot is stable and the robot is able to reach the final goal without falling.

Case Two: A quadruped robot when loaded with a 1-kg.

When the 1-kg load is added to the torso of a quadruped robot. The following Figs. 17 a and b show the walking robot standing at the stars and after a few meters.



(a)



(b)

Fig. 17: **(a)** Walking robot with a 1-kg stand at initial position **(b)** walking robot with a 1-kg after few meters.

The following Fig. 18 shows the PID control efforts of the front right leg of a walking robot with a 1–kg load.

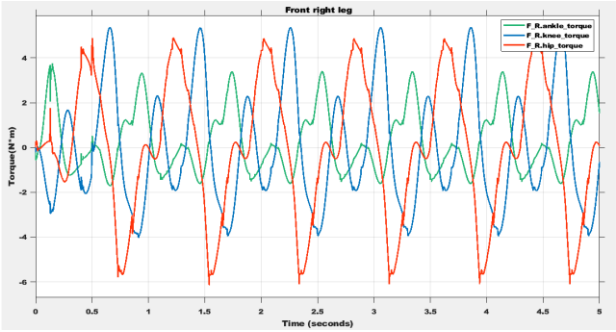


Fig. 18: PID control efforts for front right leg hip, knee and ankle links of a walking robot with a 1-kg weight.

Also, the comparison between the desired angle with the actual for the 3-joints ankle, knee and hip are shown in Figs. 19-21.

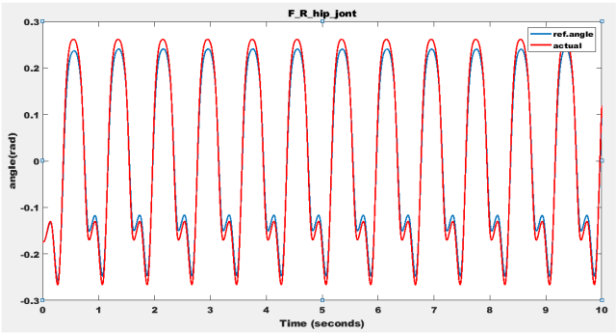


Fig. 19: Desired angle and actual angle comparisons for the hip joint.

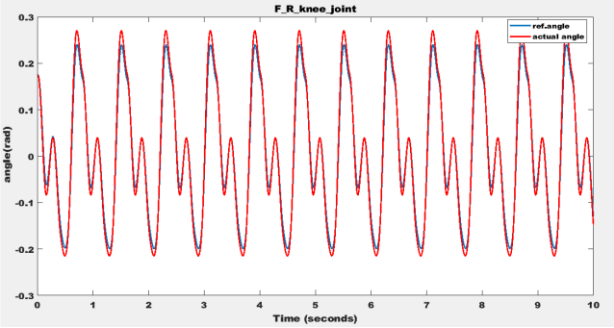


Fig. 20: Ref. angle with actual angle comparisons for the knee joint.

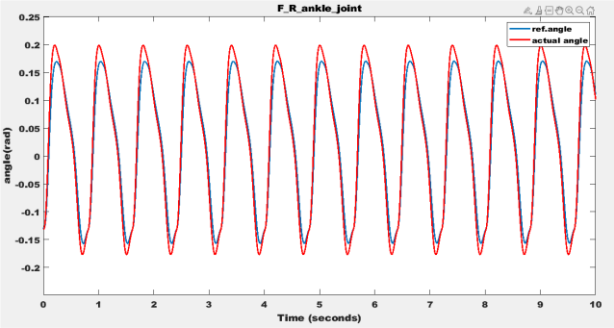
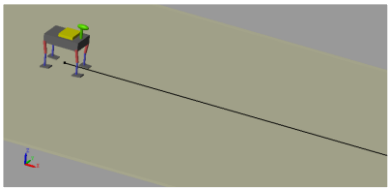


Fig. 21 Ref. angle with actual angle comparison for the ankle joint.

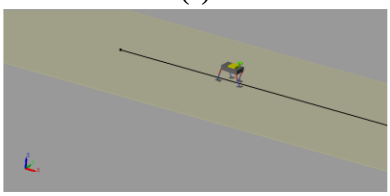
From the above results of the case, two when the robot is loaded with 1-kg on the torso with a closed-loop control system with PID controllers show that the walking of the robot is stable and the robot is able to reach the final goal without falling.

Case Three: A quadruped robot when loaded with a 2-kg.

When a 2-kg load is loaded to the torso of the quadruped robot. The following Figs. 22 a and b show the walking robot with a 2-kg weight standing at the start and after a few meters respectively.



(a)



(b)

Fig. 22: (**a)** Walking robot with a 2-kg stand at initial position (**b)** walking robot with a 2-kg after few meters.

The following Fig. 23 shows the PID control efforts of the front right leg of a walking robot with a 2–kg load.

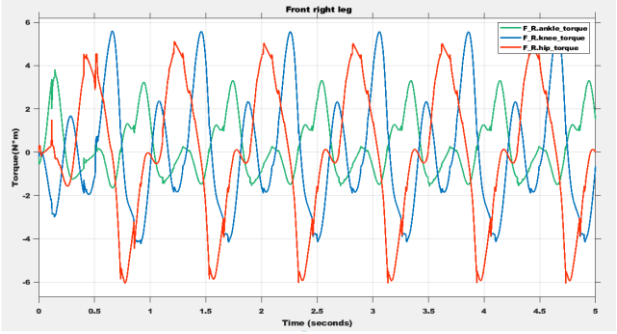


Fig. 23: PID control effects for front right leg hip, knee and ankle links of a walking robot with a 2-kg weight.

In addition, the comparison between the reference angle with the actual for the 3-joints ankle, knee and hip are shown in Figs. 24-26.

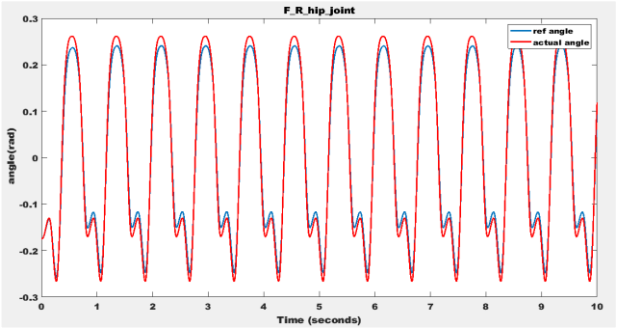


Fig. 24: Desired angle and actual angle comparisons for the hip joint.

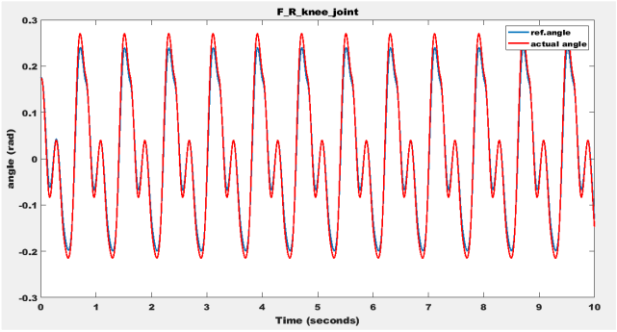


Fig. 25: Ref. angle with actual angle comparisons for the knee joint.

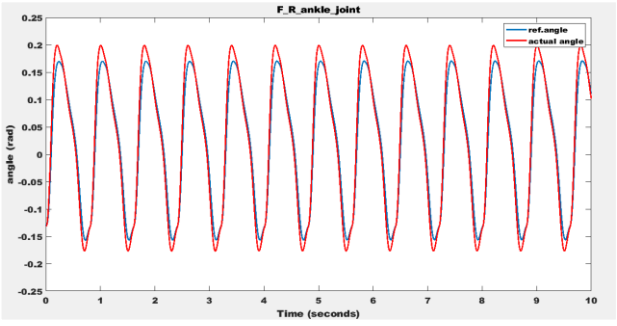


Fig. 26: Ref. angle with actual angle comparison for the ankle joint.

From the above results of the case, two when the robot is loaded with 2-kg on the torso with a closed-loop control system with PID controllers show that the walking of the robot is stable and the robot is able to reach the final goal without falling.

The following table 3 shows the results of a travelled distance in 10 sec. for three cases with the closed-loop control system.

TABLE 3
TRAVELLED DISTANCES OF ROBOT WITH PID.

Load (kg)	Travelled-distance(m) with PID
No load	2.2
1	1.9
2	1.25

B. Simulation Results of DDPG reinforcement learning.

In this section, DDPG reinforcement learning (RL) with Matlab is used to learn the quadruped robot system, whereas the DDPG reinforcement-learning algorithm is used to learn the quadruped robot for walking in a straight line. Where the RL-designer toolbox with Matlab is used to train the robot, the RL-designer toolbox is turned on several times to get the best training for a quadruped robot to walk in a straight line.

The final simulation of a quadruped robot is getting when the robot is training with 5000 episodes, and the following Fig. 26 shows the training results after running 5000 episodes.

The following parameters have been chosen to train the quadruped robot.

- Sample time (Ts) = 0.025 sec.
- Final end time (Tf) = 10 sec.
- Discount factor = 0.99.
- Mini batch size = 128.
- Experience buffer length = 500000.
- Target smooth factor = 0.001.
- Mean attraction constant = 5.
- Variance = 0.5.
- Variance decay rate = 0.00001.
- Maximum episodes = 5000.
- Maximum steps per episode = Tf/Ts.
- Score averaging window length = 100.
- Stop training value = 200.
- Save agent value = 200.
- Learning rate = 0.001.

As shown in Fig. 27, the 3rd at least of the episodes were pretty short and didn't generate much of a rapid happened where the RL algorithm was able to explore out of some local min. and then the reward became much higher until it trended upward to the point where the training cancelled, also we can see the blue curve where the individual episode rewards are stall very noisy but there is a general upwards trend with a lot of noise for that moving as we see that the (DDPG) is a high variance method, which indicates that the reward is not guaranteed to increase monotonically and the final reward that is getting from the training results is about 194.

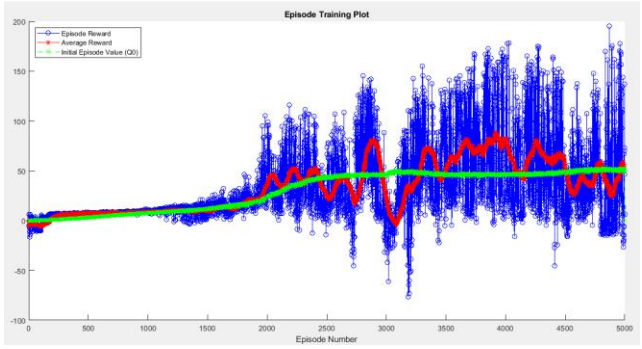
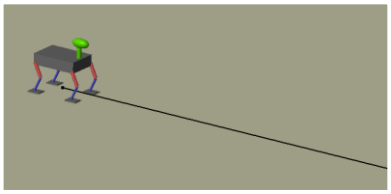


Fig. 27: Final episode training of the quadruped robot with 5000 episodes.

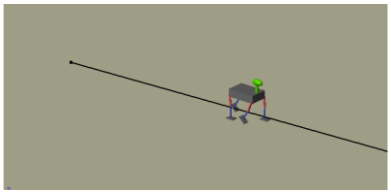
The robustness of the designed controller is studied by changing the carried weight of the quadruped robot. Simulation results can be divided into three cases, the first is when the robot is without any carried weight. The second case is when the robot is loaded with a 1-kg weight, and finally when a 2-kg weight is added to the robot.

Case One: A quadruped robot without load.

Figs. 28 a and b show the robot simulation results without load at stars and after a few meters respectively.



(a)



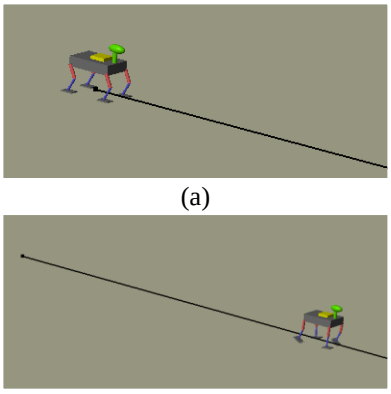
(b)

Fig. 28: a Quadruped robot with no load stand at initial position b walking robot with no load after few meters with RL.

From the above result of the case, DDPG reinforcement learning shows that the walking of the robot is stable and the robot is able to reach the final goal without falling.

Case Two: A quadruped robot when loaded with a 1-kg.

When the 1-kg load is added to the torso of a quadruped robot. The following Figs. 29 a and b show the walking robot standing at the stars and after a few meters.



(a)

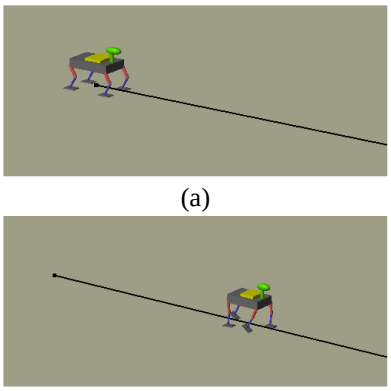
(b)

Fig. 29: **a** Walking robot with 1 kg stands at the initial position **b** walking robot with a 1-kg after a few meters with RL.

From the above results of the case, two when the robot is loaded with 1-kg on the torso with DDPG reinforcement learning show that the walking of the robot is stable and the robot is able to reach the final goal without falling.

Case Three: A quadruped robot when loaded with a 2-kg.

When the 2-kg weight is loaded onto the torso of a quadruped robot. Figs. 30 a and b show the walking robot with a 2-kg weight standing at the start and after a few meters respectively.



(a)

(b)

Fig. 30: **a** Walking robot with a 2-kg stands at the initial position **b** walking robot with a 2-kg after a few meters with RL.

From the above results of the case, two when the robot is loaded with 2-kg on the torso with DDPG reinforcement learning show that the walking of the robot is stable and the robot is able to reach the final goal without falling.
The following table 4 shows the results of a travelled distance in 10 sec. for three cases with the closed-loop control system.

TABLE 4
TRAVELLED DISTANCES OF ROBOT WITH DDPG.

Load (kg)	Travelled-distance(m) with DDPG
No load	7.5
1	7
2	6.6

V. RESULTS COMPARISON

In this section, there are some comparisons between three previously introduced cases for controlling the gait stability of a quadruped robot to attain its goal along a particular path. The comparison is based on the link between the total distance travelled and the time required to complete the entire path.

In the case of a closed loop control system with PID controllers, we can see from the table 3 the robot can travel the distance in three cases with little different distances as the load increase, while the robot in the case with no load has a larger travelled distance (2.2 m), and when the robot in the case with a 2-kg load has a smaller travelled distance (1.25 m), and when the robot in case of the 1-kg load has the travelled distance is (1.9 m). However, the robot in each case can reach the final time simulation.

In the case of DDPG reinforcement learning we can see from the table, 4 the robot can travel the distance in three cases with little different travelled distance as the load increase, while the robot in the case with no load has a larger travelled distance (7.5 m), and when the robot in the case with a 2-kg load has smaller travelled distance (6.6 m), and when the robot with a 1-kg has the travelled distance (7 m). However, the robot in each case can reach the final time simulation.
, in addition the travelled distance in the case of using DDPG as a control system is higher than that of the case with PID controllers (closed loop system) with the same simulation time (10 sec.).

Therefore, the DDPG reinforcement learning technique has an advantage over another case (closed loop system with PID controllers) in the travelled distance with the same time (10 sec.).

The following table 5 shows the comparison between the PID control system method and the DDPG reinforcement learning method with a travelled distance of 10 seconds.

TABLE 5
TRAVELLED DISTANCES COMPARITIONSOF ROBOT.

load (kg)	Travelled-distance(m) with PID	Travelled-distance(m) with DDPG
No load	2.2	7.5
1	1.9	7
2	1.25	6.6

VI. Conclusion

In this study, the following points can be noted.

1- The proposed quadruped robot model with three moveable links per leg will provide significant benefits in terms of moving easily and smoothly.

2- The robot is built using the SimScape-Multibody, in addition, the robot was tested without using any controller, and the results suggest that the robot is not completing its gait to the final goal. As a result, we offer two learning methods: the PID controller approach and reinforcement learning (RL).

3- The proposed controllers' robustness was evaluated by loading more weights onto the top of the quadruped robot. There are two distinct weights added, 1-kg and 2-kg. In each case, the controlled system was stable and flowed in a correct motion notwithstanding disturbance rejection.

4- As a result, the whole proposed system demonstrates practicability and efficacy in operating a quadruped robot. And the control design with DDPG reinforcement learning has an advantage over the control system with PID controllers.

VII. Feature works

The proposed system architecture will be built practically in future development. As a result, real-time experiments will be carried out to validate the suggested control systems, and then extensive comparisons between theoretical and actual trials will be carried out.

Conflict of Interest

The authors have no conflict of relevant interest to this article can be used.

References

[1] S. Ali, M. Khorram, A. Zamani, and H. Abedini "PD Regulated Sliding Mode Control of a Quadruped Robot ", International Conference on Mechatronics and Automation. IEEE, 2011.

[2] R. B. McGhee and A. A. Frank, University of southern California handbook, vol. 3 “On the stability properties of quadruped creeping gaits,” Math. Biosci., pp. 331–351, CRC 1968.

[3] K. Mitobe, N. Mori, K. Aida, and Y. Nasu, “Nonlinear feedback control of a biped walking robot,” Proc. - IEEE Int. Conf. Robot. Autom., vol. 3, pp. 2865–2870, 1995.

[4] S. Tzafestas, M. Raibert, and C. Tzafestas, “Robust sliding-mode control applied to a 5-link biped robot,” J. Intell. Robot. Syst. Theory Appl., vol. 15, no. 1, pp. 67–133, 1996.

[5] A. Aldair, A. Al-Mayyahi, and B. H. Jasim, “Control of Eight-Leg Walking Robot Using Fuzzy Technique Based on SimScape Multibody Toolbox,” IOP Conf. Ser. Mater. Sci. Eng., vol. 745, no. 1, 2020.

[6] M. Schlotter, “Multibody System Simulation with SimMechanics,” Analysis, no. May, pp. 1–23, 2003.

[7] R. S.Ali, A. Aldair, and A. K. Almousawi, “Design an Optimal PID Controller using Artificial Bee Colony and Genetic Algorithm for Autonomous Mobile Robot,” Int. J. Comput. Appl., vol. 100, no. 16, pp. 8–16, 2014.

[8] M. F. Aranza, J. Kustija, B. Trisno, and D. L. Hakim, “Tunning PID controller using particle swarm optimization algorithm on automatic voltage regulator system,” IOP Conf. Ser. Mater. Sci. Eng., vol. 128, no. 1, 2016.

[9] Anwer Abdulkareem Ali, Mofeed Turky Rashid," Design PI Controller for Tank Level in Industrial Process", Iraqi Journal for Electrical and Electronic Engineering, DOI: 10.37917, June 2022.

[10] A. Aldair, A. Al-Mayyahi, and W. Wang, “Design of a Stable an Intelligent Controller for a Quadruped Robot,” J. Electr. Eng. Technol., vol. 15, no. 2, pp. 817–832, 2020.

[11] P. Henderson, R. Islam, and P. Bachman “Deep Reinforcement Learning That Matter” arXiv:1709.06560v3[cs.LG], 30 Jan 2019 .

[12] Y. Duan, Xi Chen, and R. Houthooft “Benchmarking Reinforcement Learning For Continuous Control.” arXiv:1709.06560v3[cs.LG], 27 May 2016 .

[13] S. Guha, “Deep Deterministic Policy Gradient (DDPG): Theory and Implementation,” Towar. Data Sci., pp. 1–10, 2020,

[14] N. Heess, and D. TB, S. Sriram, "Emergence of Locomotion Behaviours in Rich Environments" arXiv:1707.02286 [cs.AI], 7 Jul 2017.