International Journal of Intelligent Robotics and Applications
https://doi.org/10.1007/s41315-025-00496-6

REGULAR PAPER

Reinforcement learning for hexapod robot trajectory control: a study of Q-learning and SARSA algorithms

Ahmed Benyoucef[1] · Youcef Zennir[1] · Ammar Belatreche[2] · Manuel F. Silva[3] · Mohamed Benghanem[4]

Received: 8 July 2025 / Accepted: 11 September 2025

Abstract
Hexapod robots, with their six-legged design, excel in stability and adaptability on challenging terrain but pose significant control challenges due to their high degrees of freedom. While reinforcement learning (RL) has been explored for robot navigation, few studies have systematically compared on-policy and off-policy methods for multi-legged locomotion. This work presents a comparative study of SARSA and Q-Learning for trajectory control of a simulated hexapod robot, focusing on the influence of learning rate (α), discount factor (γ), and eligibility trace (λ). The evaluation spans eight initial poses, with performance measured through lateral deviation (Ey), orientation error (Eθ), and iteration count. Results show that Q-Learning generally achieves faster convergence and greater stability, particularly with higher γ and λ values, while SARSA can achieve competitive accuracy with careful parameter tuning. The findings demonstrate that eligibility traces substantially improve learning precision and provide practical guidelines for robust RL-based control in multi-legged robotic systems.

Keywords Hexapod robot · Reinforcement learning · Q-learning · SARSA · Trajectory control · Learning parameters

1 Introduction

Recent researches in the field of walking robots has garnered significant attention due to their potential to navigate various challenging terrains (Bjelonic et al. 2017; Lagaza and Pandey 2018). Among these, multi-legged robots, particularly hexapod robots, stand out for their complex structures and capabilities. A hexapod robot, with its six legs, each possessing multiple degrees of freedom, represents one of the more intricate models of walking robots (Ma et al. 2022). When in a stance state, hexapod robots exhibit superior stability compared to bipedal (Ji et al. 2021) and quadruped robots (Boaventura et al. 2013), especially when traversing uneven terrains. However, this enhanced stability often comes at the cost of reduced speed compared to their two- and four-legged counterparts (Silva and Machado 2012).

Numerous studies have focused on six-legged (hexapod) walking robots (Li et al. 2022), as they offer an optimal balance between mobility, stability, and mechanical complexity (Brooks 1989; Kirchner 1997; Kingsley et al. 2006). Their ability to navigate uneven terrain and maintain stability makes them particularly useful for accessing environments that are hazardous or inaccessible to humans, such as space exploration missions or operations within nuclear facilities.

Controlling the locomotion of a hexapod robot involves multiple strategies (Ijspeert 2008; Fuchs 2010). Key challenges include maintaining stability during both transfer and stance states, managing the distance between the legs (Silva and Machado 2008), and coordinating the gait to ensure smooth movement over challenging terrains with high stability (Chen et al. 2020). The importance of stability, particularly during the stance phase (when the legs are on the ground) and the transfer phase (when the legs are in the air), cannot be overstated (Silva and Machado 2008). Furthermore, achieving homogeneous timing and movement across

✉ Mohamed Benghanem
mbenghanem@iu.edu.sa

1 Automatic Laboratory of Skikda, University of Skikda, 21000 Skikda, Algeria

2 Northumbria University, Newcastle Upon Tyne NE1 8ST, UK

3 Instituto Politécnico do Porto, ISEP - Instituto Superior de Engenharia Do Porto, 4249-015 Porta, Portugal

4 Physics Department, Faculty of Science, Islamic University of Madinah, 42351 Madinah, Saudi Arabia

Published online: 25 October 2025

all legs, whether the robot is walking straight or turning, is complex due to the numerous degrees of freedom involved. Controlling all six legs simultaneously for experiments like moving forward, backward, or turning right and left presents a significant challenge.

Researchers have traditionally approached hexapod control using basic control methods, which focus on walking mechanics (Benyoucef and Zennir 2023) and navigation (Benyoucef et al. 2024). Legged animals' locomotion is often described as a rhythmic action (Cai et al. 2021). Central Pattern Generators (CPGs), which use centrally generated rhythms to influence overall behaviour, have been used in certain experiments as open-loop oscillators to obtain a rhythmic gait (Ijspeert 2008). Evidence for the effectiveness of such methods has been found in insects that walk quickly, such as cockroaches (Fuchs 2010). However, with a large number of joints, as in hexapods, open-loop oscillator solutions are typically insufficient for precise control, especially in unstructured terrains (Schilling and Melnik 2018, Aissaoui et al. 2024) develop a kinematic control method for walking on uneven terrain using a non-symmetrical tripod gait. In otherwise benyoucef et al. make as close-loop control tripod gait of hexapod robot's legs (Benyoucef and Zennir 2023; Benyoucef et al. 2024).

Reinforcement learning has also shown increasing relevance for the control of flexible continuum robots, which may represent a promising direction for the next generation of hexapod leg designs. For example, Djeffal et al. (2024) applied Deep Deterministic Policy Gradient (DDPG) to control a high-DoF spatial continuum robot, highlighting the adaptability of RL for bio-inspired continuum architectures that could be adapted to flexible hexapod platforms.

To address the limitations of purely classic approaches, hybrid methods have emerged that combine biologically inspired controllers with learning-based algorithms. For instance, Yang et al. integrated DRL with CPGs to develop adaptive gait generation methods, demonstrating improved terrain adaptability (Yang et al. 2023). This approach leverages the strengths of DRL in handling high-dimensional control problems while benefiting from the rhythmic stability of CPGs. Nonetheless, such methods can be computationally intensive, and real-world implementation may face challenges due to discrepancies between simulated and physical environments.

Building on these developments, advanced reinforcement learning (RL) have gained prominence due to their adaptability in complex environments (Qiu et al. 2023). Among model-free algorithms, State-Action-Reward-State-Action (SARSA) (López-Lozada et al. 2021) and Q-learning (Sutton and Barto 2014), are widely used for optimizing gait (Wang et al. 2019), obstacle avoidance (Ribeiro et al. 2019), and adaptive locomotion (Margolis et al. 2022). Additionally, distributed Q-learning has been applied to trajectory planning (Zennir et al. 2005), enhancing agility and adaptability in unstructured environments. Miki demonstrated the effectiveness of Deep Reinforcement Learning (DRL) by training a neural policy for the ANYmal robot to balance a lightweight ball using its limbs, without requiring tactile sensors (Miki et al. 2022). Fu et al. developed a novel DRL technique for contact motion planning, enabling multi-legged robots to traverse uneven plum-blossom piles (Fu et al. 2021). Actor–Critic methods have also been used to decompose tasks into a hierarchy of sub-tasks (Tan et al. 2018); for instance, Zhigang Huang et al. (2023) applied Hierarchical Reinforcement Learning (HRL) to coordinate low-level and high-level controllers for complex behaviours. A hierarchical free gait motion planning framework for hexapod robots that combines DRL with free gait strategies to improve locomotion in unstructured terrains was also introduced in Wang et al. (2023a). Tan employed Proximal Policy Optimization (PPO) to develop a motion control policy for a quadruped robot, successfully transferring it from simulation to a physical platform (Tan, et al. 2018). Peng et al. introduced a deep learning-based optimization approach for training a bipedal robot in simulated environments, enabling navigation through randomly placed obstacles (Peng et al. 2017). Algorithms based on Markov Decision Processes (MDPs) have also proven effective in robotic locomotion (Socha et al. 2016).

Beyond RL, Model Predictive Control (MPC) has been explored for multi-legged robots. Wang et al. (2023b) enhanced MPC-based path planning and tracking frameworks for hexapod robots, while Thor et al. (2022) introduced a modular neural control architecture with rapid learning capabilities. This allowed behaviour-specific control modules to be incrementally integrated, leading to progressively more complex emergent locomotion behaviours.

In the context of path planning, Li et al. (2022) proposed an Improved Double Deep Q-Network algorithm to address multi-objective optimization challenges, improving navigation efficiency in dense environments but at the cost of increased computational overhead, which limits its real-time applicability (Chen et al. 2024). In contrast, non-RL approaches rely on deterministic or biologically inspired algorithms. Owaki et al. introduced a dynamical systems approach for reliable obstacle navigation in blind modular robots, which proved effective in simpler scenarios but lacked adaptability to unforeseen environments (Travers et al. 2016). Similarly, Yang et al. applied CPGs with terrain adaptation, offering efficient control but struggling in dynamic or uneven terrains due to the absence of explicit learning mechanisms (Yang et al. 2023). Traditional approaches, such as the Dynamic Window Approach

(DWA), remain popular for obstacle avoidance because of their simplicity and computational efficiency (Dobrevski and Skocaj 2020). However, such methods often struggle to generalize to unstructured environments and can exhibit limited robustness when facing dynamic obstacles (Qu et al. xxxx). Overall, RL-based approaches demonstrate greater adaptability and learning capacity in complex and unstructured settings, but they introduce challenges in computational load and real-time feasibility. In contrast, classic methods offer simplicity and efficiency but are less capable of handling environmental variability or complexity.

While these investigations have yielded positive results in locomotion control and navigation of robots across various environments, there is a noticeable gap in comparative studies between different models and algorithms. Even fewer researchers have attempted to enhance existing models by systematically comparing their performance under varying conditions.

Given this, this study proposes a comparison between two core model-free approaches to reinforcement learning reinforcement learning (RL) methods: Q-learning (λ) and SARSA (λ). These algorithms are particularly well-suited for controlling the movement of a hexapod robot using a predefined gait in a simple, obstacle-free environment. The primary objective is to enable the robot to navigate from different starting points to a designated target location. To achieve this, the robot is placed in various starting poses and learns how to navigate to the target through iterative training. The simulation process is divided into distinct Experiments, during which we systematically adjust key algorithm parameters learning rate (α), discount factor (γ), and eligibility trace (λ) to analyse their influence on performance. The study is structured into two primary phases. In the first phase, the hexapod robot learns to walk and adapts its locomotion to transition from the initial to the target pose on a flat plane along different trajectories, first, by applying the Q-learning algorithm, followed by the SARSA algorithm. In the second phase, we progressively vary the parameters (α, γ, λ) to evaluate their impact on the performance and adaptability of the RL algorithms in improving the robot's locomotion and navigation. Finally, we present the simulation results and conduct a comparative analysis to assess the effectiveness of Q-learning (λ) and SARSA (λ) in guiding the robot along the trajectories. This study aims to examine the influence of the learning parameters on ability of the robot to follow trajectories with precision, adapt to changing environments, and achieve efficient locomotion. Additionally, we seek to determine which algorithm performs better under these conditions and identify the optimal parameter settings for achieving superior performance.

In summary, this work aims to evaluate and compare the performance of Q-learning with eligibility traces and SARSA with eligibility traces in a hexapod robot trajectory-following experiments. It further investigates the influence of key learning parameters (α, γ, λ) on the effectiveness of each algorithm. The main contributions of this paper are: 1) an empirical comparison of two reinforcement learning algorithms applied to hexapod locomotion, 2) insights into parameter effects on learning convergence and accuracy, and 3) guidelines for algorithm selection in similar robotic applications.

The organization of the paper is as follows: Sect. 2 introduces locomotion and kinematic model of the hexapod robot. Section 3 describes the methods and algorithms used in this study, providing a detailed overview of the employed techniques. Section 4 focuses on path planning and locomotion control, explaining how these components are integrated to enhance the robot's functionality. Section 5 outlines and discusses the results from the simulation, while Sect. 6 concludes the paper with a summary of the findings.

2 Multi-legged robot model

We dedicate this section to the modelling of the hexapod robot, addressing all key elements of its design, solve its kinematic model, and analyse its locomotion to enable control of its locomotion while ensuring high stability during walking.

2.1 The hexapod robot's structural configuration

In this work, we used a hexapod robot with $n=3$ degrees of freedom (DoF) per leg. The robot's right side has three legs numbered 1, 2, and 3 (from front to back of right side, respectively), and the opposite side has legs numbered 4, 5 and 6 (as shown in Fig. 1). Each leg has three joint angles, denoted as θij, where 'i' represents the joints (Coxa (c), Femur (f), and Tibia (t)) and 'j' denotes the leg number ($j=1, 2\ldots 6$). Figure 1 illustrates the hexapod robot legs along with its joints.

2.2 Locomotion and kinematic model of the robot

The locomotion of the robot refers to the movement of its legs and body for walking forward, backward, or turning right and left. Additionally, it can move its body by pitching, yawing, and rolling. Figure 2 presents the model parameters (Fc, Ls, H_B, Sp, V_R and β) adopted for controlling the locomotion of the robot's legs. Only the right side of the robot is shown in Fig. 2 to explain the locomotion of the legs and joints, as the left side is identical.

$L1$, $L2$ and $L3$ are the number of robot's legs. Each leg has three links lj,i where i is the link name (i: c, f and t), for

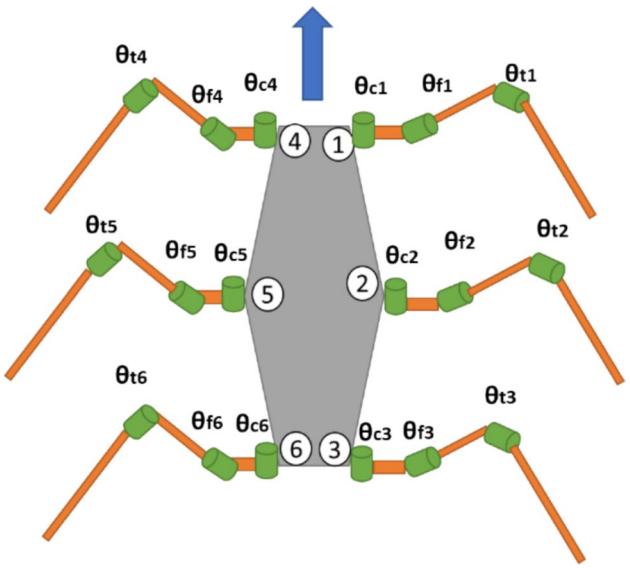

Fig. 1 Configuration and distribution of the hexapod robot's leg joints (θ_{cj}, θ_{fj}, θ_{tj})

coxa link, femur link and tibia link, respectively, and j is the number of leg (j=1,2,3,…6). "Pi L1" is the initial position of the leg 1 when it was walking, "Pm L1" is its middle position and "Pf L1" is the final position of the end effector. Table 1 illustrates an example of locomotion of the leg 1 with these three positions.

The motion is described using a world coordinate system as outlined in Silva et al. (2005). The kinematic model includes the parameters presented in Table 2. Additionally, a periodic trajectory is defined for each leg, with the robot's velocity given by $V_R = \frac{L_s}{T}$. For more detail see the work of Silva et al. (2008).

For instance, the swing leg trajectory can be modelled using a simplified cycloid function, Eq. (1)

Table 1 Position of joints angle of single leg

Position	θ_c (°)	θ_f (°)	θ_t (°)
Initial position Pi	0	30	0
Middle position Pm	45	60	0
Final position Pf	45	30	0

Table 2 The kinematic model parameters

Symbol	Description
T	A cycle time
β	A duty factor
t_T	Transfer time, where: $t_T = (1-\beta).T$
t_S	Support time, where: $t_S = \beta.T$
L_S	Step length
S_P	Stroke length
H_B	Robot body height
F_C	Peak foot clearance
O_i	Foot trajectory offset
VR	A robot velocity

$$y(t) = F_C \cdot sin(\frac{\pi t}{t_T}) \tag{1}$$

where F_C denotes the maximum foot clearance (Fig. 2) and t_S represents the step duration (Table 2).

For the locomotion of a hexapod robot, both forward and inverse kinematics are required. Benyoucef and Zennir (2023); Benyoucef et al. (2024) presented the mathematical models for the forward and inverse kinematics of a hexapod robot leg.

3 Methods and algorithms

Achieving stable, precise, and efficient locomotion in hexapod robots necessitates the use of robust control algorithms. In this section, we present two model-free reinforcement

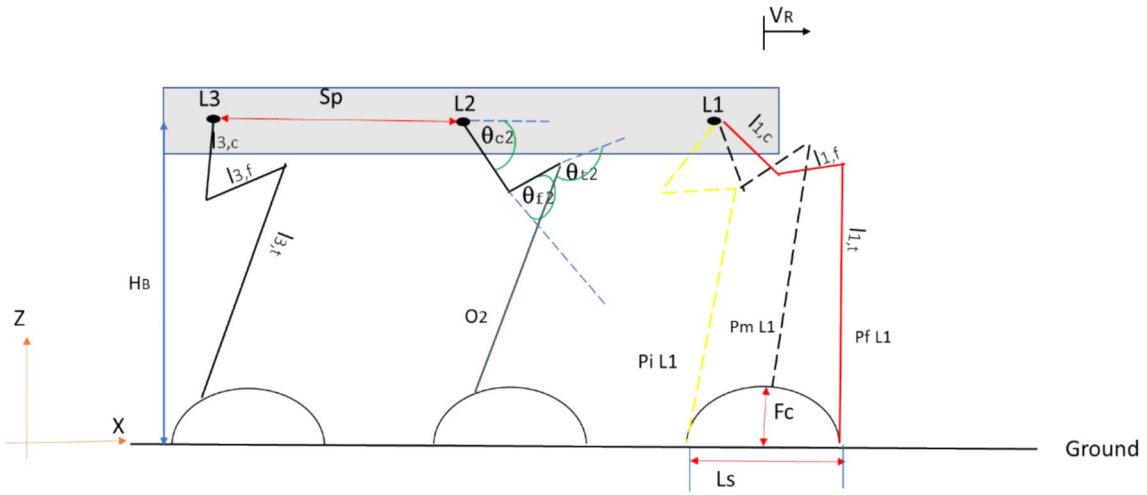

Fig. 2 Motion description of the hexapod robot based on coordinate system and kinematic variables

learning approaches Q-Learning and SARSA and describe how they are employed to control the movement and navigation of the robot.

3.1 Reinforcement Learning (RL)

Reinforcement Learning (RL) is a subset of machine learning where an agent acquires decision-making capabilities through interactions with its environment, receiving feedback in the form of rewards or penalties (AlMahamid and Grolinger 2021; Ding et al. 2020; Coelho et al. 2021). The core aim is to optimize the total accumulated reward over time. RL methods are especially well-suited for tasks in dynamic settings, where ongoing adaptation of the agent's strategy based on environmental responses is essential (Dulac-Arnold et al. 2021). RL revolves around the interaction between the agent and its environment. The agent takes actions that affect the environment's state, which represents its current situation, and receives feedback in the form of rewards, guiding it toward more optimal behaviours. The agent follows a policy, a strategy for choosing actions based on the current state. Another key component is the value function, which estimates the expected cumulative reward for being in a given state while adhering to a particular policy.

RL can be classified into two main categories: model-free and model-based approaches. In model-free RL, the agent directly learns the policy through interactions, without creating a model of the environment. Conversely, model-based RL involves the agent either constructing or leveraging a model to simulate the environment and plan its actions accordingly.

3.2 Q-learning

Q-Learning is a popular model-free reinforcement learning algorithm (Sutton and Barto 2014), primarily aimed at finding the optimal action-selection policy for a given finite Markov Decision Process (MDP). This is achieved by learning the Q-function, a value function that estimates the expected utility of taking a specific action in a given state and then following the optimal policy. The learning process is driven by iterative updates of Q-values (state-action values), guided by the Bellman equation (Eq. 2) (Lin et al. 2016; Nishigai and Ito 2011; Sutton and Barto 2018):

$$Q(s,\alpha) \leftarrow Q(s,\alpha) + \alpha.[r + \gamma.max_{\alpha'}Q(s',\alpha') - Q(s,\alpha)] \quad (2)$$

where:

- Q(s, a) is the current value of taking action "***a***" in state "***s***".
- α is the learning rate ($0<\alpha\leq 1$).
- r is the reward received after taking action a.
- γ is the discount factor ($0\leq\gamma<1$).
- $max_{\alpha'}Q(s',\alpha')$ is the maximum estimated future reward for the next state "s′".

Lambda (λ) in Q-Learning refers to the use of eligibility traces (Sutton and Barto 2014), which are a mechanism for more efficiently updating the Q-values. The parameter λ controls the decay of these traces (Rummery and Niranjan 1994), blending between one-step and n-step methods. This approach is often known as **Q(λ)** (Siciliano et al. 2016)**,** or Q-Learning with Eligibility Traces.

Q-learning consists of two types: Distributed and Centralized Q-learning, which are described in Sects. 3.2.4 and 3.2.5.

3.2.1 Application of Q-learning in hexapod robots

As previously stated, hexapod robots require complex control algorithms for efficient and adaptive locomotion. Q-Learning can be applied to hexapod robots in the following ways (Qiu et al. 2023; Singh et al. 2020):

1. Locomotion Control:
   - **Gait Learning:** Q-Learning can be used to learn and optimize different gaits for the hexapod to efficiently navigate various terrains. By defining states as different leg positions and actions as movements of the legs, the hexapod can learn which gaits provide the best stability and speed for given conditions.
2. Navigation and Path Planning:
   - **Obstacle Avoidance:** Hexapod robots equipped with sensors can use Q-Learning to learn to navigate around obstacles. The state can include sensory inputs and robot position, while actions involve movement decisions. Rewards can be given for progress towards the goal while penalizing collisions.
3. Adaptive Behaviour:
   - **Environmental Adaptation:** The hexapod can adapt to changes in its environment, such as varying ground textures or slopes, by learning from the feedback it receives during its movements. This helps the robot maintain stability and efficiency in diverse settings.
4. Experiment-Specific Learning:

- **Specific Missions:** Hexapod robots can be trained using Q-Learning for specific Experiments, like search and rescue, where the robot learns optimal strategies for covering ground efficiently and locating targets based on real-time feedback.

By leveraging Q-Learning, hexapod robots can autonomously improve their performance over time, leading to more robust and adaptable robotic systems.

3.2.2 Q-Learning (λ) algorithm for controlling a hexapod robot

In our simulation, we use distributed $Q(\lambda)$ as presented in Algorithm 1, which will be explained in the Sect. 3.2.4.

The update at each moment t of the $Q(s, a)$ value associated with the couple (status, action a) is written in Eq. (3):

$$Q^i(s,a) \leftarrow Q^i(s,a) + \alpha.\left[r + \gamma.max_{a\prime}Q^i(s',a') - Q^i(s,a)\right] \quad (3)$$

With $\alpha \in [0, 1]$ is the learning step, $\gamma \in [0, 1]$ is the factor of weighting, and $\lambda \in [0, 1]$ is the decay rate. Algorithm 1 presents Q-Learning algorithm for each agent "i".

Algorithm 1: Q-Learning (λ) algorithm for controlling a hexapod robot.

```
Initialize arbitrarily Q^i(s, a), λ ∈ [0, 1]
Repeat (for each episode):
    Initialize s (any stable configuration)
    Repeat (for each step)
        Take action according to strategy from Q
        Observe r, s'
        Update Q^i(s, a) (Equation 3)
        If falls:
            reset (any stable configuration)
        Else s ← s'
    Until end step
Until correct operation or end of the episode
```

The Q-table, a matrix of dimensions (s, a), was initialized with zeros, $Q(s, a) = 0$. The *initial state* sss and the *reset* condition refer to any stable configuration in which the robot's centre of gravity is aligned with the posture centre. Among the 64 possible global position configurations of the hexapod robot, 18 are classified as stable cases, as presented by Zennir in Singh et al. (2020). These configurations were used with both the Q-Learning (Algorithm 1) and SARSA (Algorithm 2). A more detailed explanation of the robot's stability when using the tripod gait is provided in Sect. 4.2. Table 3 summarizes the inputs and outputs of the Q-Learning algorithm.

Table 3 Input and output parameters of Q-Learning (λ)

Input	Output
State space s	Updated Q-value table $Q(s, a)$
Action space a	
Learning rate α	
Discount factor γ	
Exploration rate ϵ	
Eligibility trace λ	

3.2.3 Centralized approach to reinforcement learning

Consider a scenario involving N agents (or actors) learning a joint strategy where a joint strategy encompasses the set of strategies employed collectively by all agents to contribute to a unified experiment. In a centralized reinforcement learning framework, the state information is aggregated at a central decision-making unit, which is responsible for updating utility values and selecting actions for each agent. Let $|A|$ denote the number of possible actions available to each agent (assumed identical across agents for generality), and $|S|$ represent the total number of possible system states. Under this centralized approach, the system must maintain N Q-value tables, each of size $|S| \times |A|$, corresponding to the value function $Q(s, a)$ for each agent.

In this framework, a uniform reinforcement signal is assigned to all agents, regardless of their individual roles in the success or failure of the collective task. This setup closely resembles a stochastic team game scenario, where a single Q-table $Q(s, a)$ is employed to evaluate the state-action values for the entire system. The architecture is centralized one central agent is tasked with collecting all state and action data, updating a shared memory structure, and deciding the appropriate action policies for each agent. As a result, all agents operate under the same feedback signal. To move toward decentralization, a key modification involves enabling each agent to independently determine its own action-selection strategy. Although agents may still rely on a common memory for state or state-action information, their behaviour can diverge based on individual exploration or exploitation strategies influenced by local constraints. This independence may lead to the need for certain agents to offset the limitations of others. Figure 3 illustrates the centralized reinforcement learning architecture.

Advantages expected from a Centralized Approach are:

- A holistic view of the entire system, enabling the analysis of all scenarios and potential actions.
- Coordination issues among actors are addressed at a single decision-making level.

However, the approach shows several disadvantages, including:

- The entire system is vulnerable to failures at the central decision-making point.

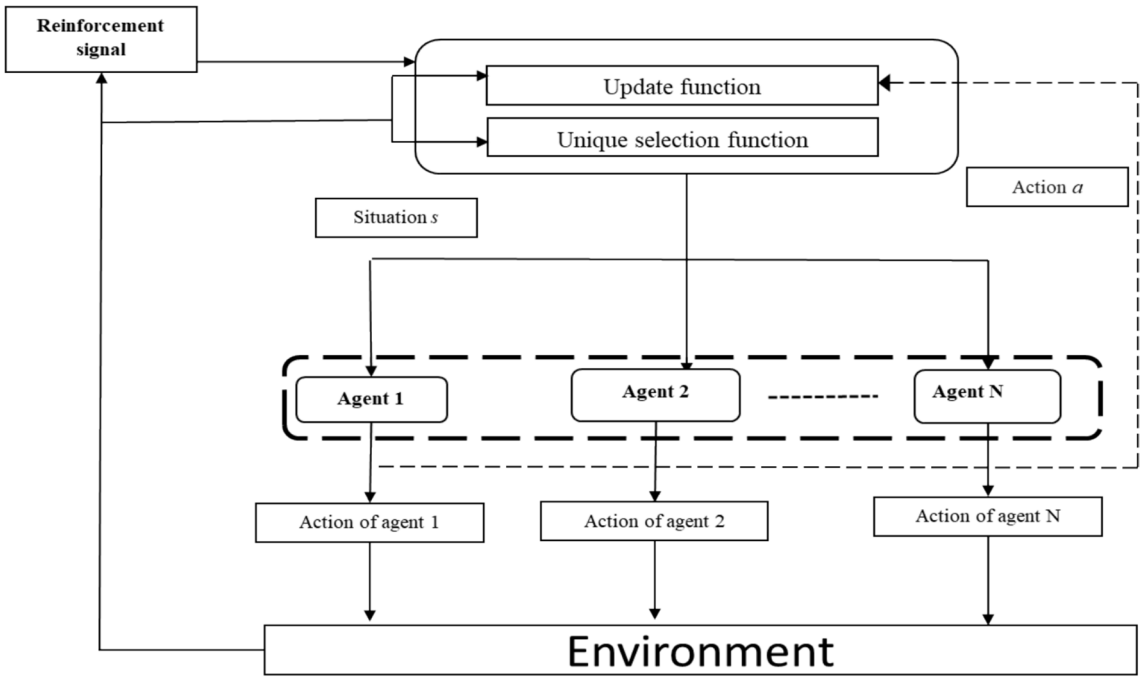

Fig. 3 Centralized architecture to reinforcement learning applied on N agents

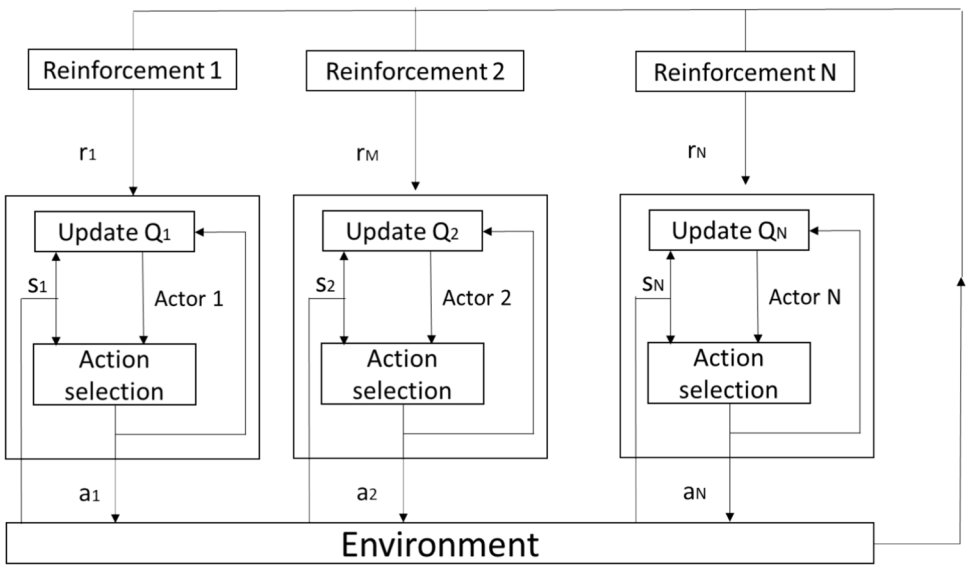

Fig. 4 Distributed approach to reinforcement learning applied on N agents

- As the number of agents increases, the number of state-action pairs (s, a) expands exponentially.
- In team scenarios, positive effects at the overall system level can be detrimental to an individual actor, as local pressures are not considered.

We have a set of agents (controllers) where each independent controller commands, for example, a leg of a robot. All these controllers are managed by a central agent, which represents an update mechanism for the controllers according to a reinforcement learning algorithm.

3.2.4 Distributed approach to Q-learning

With an approach distributed by learning, illustrated by Fig. 4, each agent operates independently, potentially receiving different state information and reinforcement signals, thereby conducting its own learning process. Assuming that all N agents share the same state space, the total number of state-action values $Q(s, a)$ to be updated becomes $N \times |S| \times |A|$. Let $|S|$ denote the number of states, and $|A|$ represent the number of possible actions.

Each leg is therefore considered an agent with status information on the robot and able to perform movements

allowing the robot to fulfil its mission. Each agent has its own learning algorithm.

The distributed approach to reinforcement learning is illustrated in Fig. 5. Each leg of the hexapod robot is controlled independently (Zennir et al. 2003a). Each controller agent "decides" the next movement based on the binary state (in the air or on the ground) of the other legs. Not all legs necessarily receive the same state information or the same reinforcement signal, and the learning is local for each leg (Amhraoui and Masrour 2024).

The distributed Q-Learning approach for control and learning is expected to offer several significant advantages. It provides greater flexibility, allowing the system to adapt more easily to unforeseen environmental changes. Additionally, it enhances reliability by tolerating individual errors, ensuring the overall performance remains unaffected by localized issues. Furthermore, it exhibits greater robustness as the problem-solving capacity emerges collectively from the entire system rather than relying on any single component. However, this approach poses several challenges (Zennir et al. 2003b). One of the primary difficulties lies in defining local objectives that align seamlessly with the global objective. Another critical challenge involves selecting appropriate modes of cooperation among operational entities or agents. These modes include synchronization, which ensures the sequencing of actions over time; collaboration, which enables effective Experiment sharing; and coordination, which focuses on conflict resolution and performance enhancement.

3.3 State-Action-Reward-State-Action (SARSA) and SARSA (λ) algorithms

SARSA is a fundamental RL algorithm that employs on-policy temporal difference learning to estimate action-value functions, enabling agents to select the most optimal course of action. It provides a structured framework for agents to interact with their environment, accumulate rewards, and iteratively enhance their decision-making process. This section focuses on the different types of SARSA and the implementation of the algorithm for controlling the locomotion of the hexapod robot to follow its desired trajectory (Sutton and Barto 2014). The SARSA algorithm is one from reinforcement learning algorithm type, which belongs to the category of on-policy temporal difference learning methods (Siciliano et al. 2016). It updates its action-value function, is presented as $Q(s, a)$, based on the actions it actually takes while interacting with the environment (Wenxia et al. 2021). The global SARSA update expression as defined from Sutton and Barto (Sutton and Barto 2018) is illustrated in Eq. 4:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_{t+1} + \gamma Q(s',a') - Q(s,a)] \tag{4}$$

where:

- s and s' are the current and next states respectively.
- a and $a'$are the current and next actions.
- r is the reward received for transitioning from s to s' via action a.
- α is the learning rate.
- γ is the discount factor.

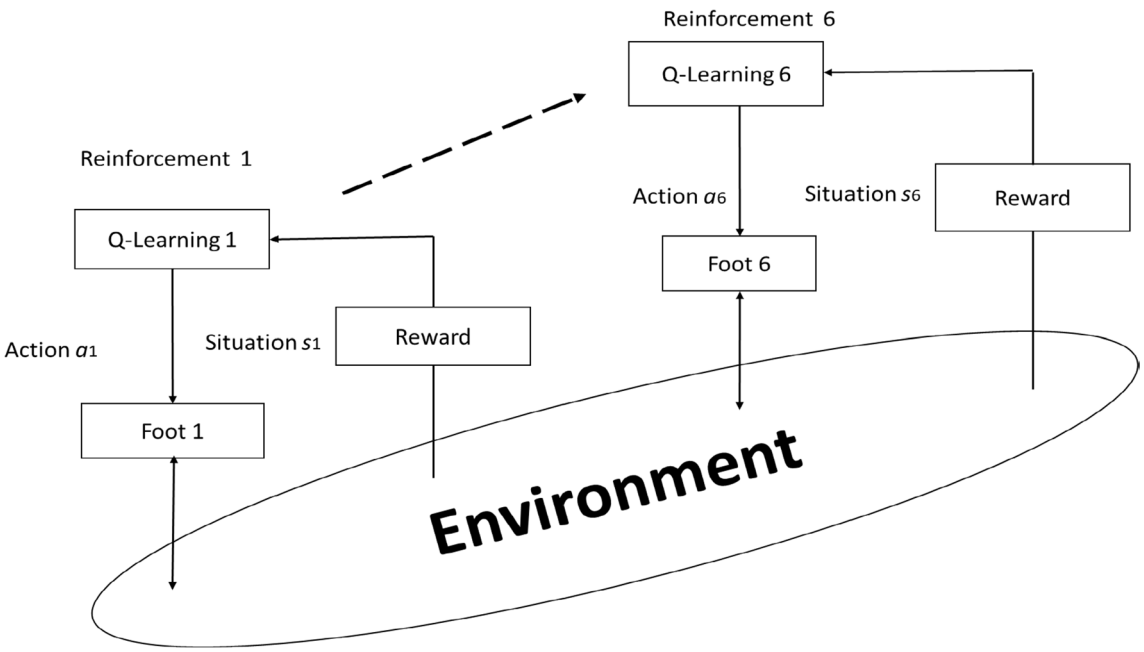

Fig. 5 Distributed Q-learning architecture for reinforcement learning in the hexapod robot

- Q(s, a) is the current value of the state-action pair (s, a)
- Q(s', a') is the estimated value of the next state-action pair (s', a').

Equation (5) defines the SARSA update rule for each leg i∈{1, …, 6}.

$$Q^i(s,a) \leftarrow Q^i(s,a) + \alpha.\left[r_{t+1} + \gamma.Q^i(s',a') - Q^i(s,a)\right] \quad (5)$$

where the variables *s, a, s', a', r_{t+1}* are defined in Eq. 4.

SARSA is particularly useful in environments where the safety and reliability of the actions taken are critical, since it updates the policy based on actual actions and observed rewards.

SARSA(λ) extends SARSA by incorporating eligibility traces (Sutton and Barto 2014), allowing it to perform more efficient updates.

Eligibility Traces (λ): Similar to Q(λ), SARSA(λ) uses eligibility traces (Singh and Sutton 1996) to keep track of the occurrences of state-action pairs. These traces help propagate updates back to previously visited state-action pairs, speeding up learning. Where $\lambda \in [0,1]$, which determines how far back the influence of rewards is distributed. Higher values of λ place more emphasis on longer sequences of actions, effectively blending Monte Carlo and TD-learning approaches(Sutton and Barto 2014).

The update rule with eligibility traces becomes:

Table 4 Input and output parameters of SARSA

Input	Output
▪ State space s ▪ Action space *a* ▪ Learning rate α ▪ Discount factor γ ▪ Exploration rate ϵ ▪ Eligibility trace λ	▪ Updated Q-value table *Q*(*s*, *a*)

$$e_t(s,a) \leftarrow \gamma\lambda e_{t-1}(s,a) + 1(s = s_t, a = a_t) \quad (6)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\delta_t \cdot e_t(s,a) \quad (7)$$

where:

- The eligibility trace $e_t(s,a)$ for pair (*s, a*)
- The temporal difference error δ_t:

$$\delta_t = r_{(t+1)} + \gamma Q(s',a') - Q(s_t,a_t) \quad (8)$$

- 1 is the indicator function.

3.3.1 SARSA (λ) algorithm for controlling a hexapod robot

The inputs and the output of the SARSA algorithm are presented on Table 4 and the Algorithm is shown in Algorithm 2.

Algorithm 2: SARSA (λ) Algorithm for controlling a hexapod robot.

Initialize $Q^i(s,a)$ arbitrarily for all $\boldsymbol{s} \in \boldsymbol{S}$ *and* $a \in A$. $\lambda \in [0, 1]$
Repeat (for each episode):
  Initialize state ***s*** (any stable configuration).
  Choose action ***a*** from ***s*** using an epsilon-greedy policy derived from **Q**.
  Repeat (for each step)
    Take action ***a***, observe reward **r** and next state **s′.**
    Choose next action ***a′*** from **s′** using an epsilon-greedy policy derived from **Q**.
    Update **Q-value:** $\boldsymbol{Q^i(s,a) \leftarrow Q^i(s,a) + \alpha.\left[r_{t+1} + \gamma.Q^i(s',a') - Q^i(s,a)\right]}$
    If falls:
      Reset to any stable configuration
    Else: Set s←s' and a←a'
  Until end step
Until correct operation or end of the episode

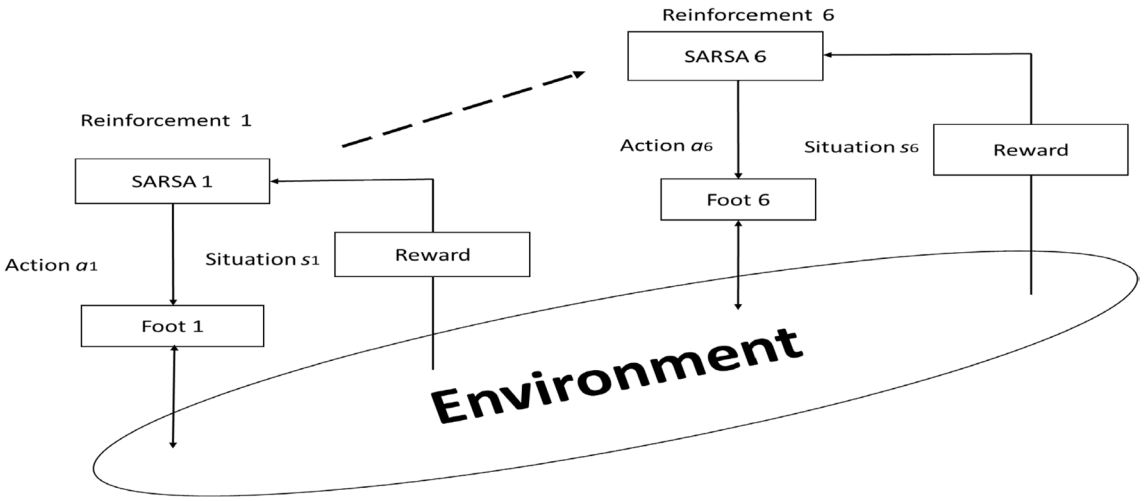

Fig. 6 Distributed SARSA architecture for reinforcement learning in the hexapod robot

The initialization procedure is described in Sect. 3.2.2.

3.3.2 Centralized approach to SARSA

In centralized SARSA, the decision method and exploration strategy are not specified, making it difficult for the robot to find the optimal trajectory, and the number of steps is greater compared to distributed SARSA. Its architecture is presented in Fig. 3.

3.3.3 Distributed approach to SARSA

The distributed SARSA algorithm estimates the value of the next state s' based on the action selected by the decision-making process. In state s, the action $a \leftarrow$ Decision (Q,s) is performed, and the agent receives the reward r and observes the new state s'. From state s', the next action $a' \leftarrow$ Decision (Q,s') is then chosen. The update of the state–action pair (s, a) by the SARSA algorithm for each leg i is given in Eq. (5), and its architecture is illustrated in Fig. 6.

Compared to distributed Q-learning, the only difference lies in the update rule: replacing $\gamma \cdot max_{a'} Q_i(s', a')$ with $\gamma \cdot Q_i(s', a')$. In other words, SARSA updates the Q-value based on the action that will actually be taken in the next step the action a' selected by the current policy making it an on-policy algorithm. This means SARSA learns the value of the policy being followed, rather than estimating the value of the optimal policy. However, this requires explicit knowledge of the next action a', which ties the learning process to the current behaviour policy and can constrain exploration during training.

3.4 The role of the learning parameters (α, γ, λ).

The learning parameters in R play a critical role in ensuring the stability of the robot's learning process and in achieving convergence toward the target. Sutton et al. (Sutton and Barto 2014) introduced a comprehensive overview of reinforcement learning methods, including Q-Learning, SARSA and Q-Learning(λ), SARSA(λ), highlighting the differences among them and emphasizing the importance of learning parameters. In another study, Xiaolin Zhou (2022) compared these parameters within the context of stochastic mazes; however, their analysis did not incorporate the use of eligibility traces (λ).

This subsection presents an overview of the learning parameters α (learning rate), γ (discount factor), and λ (eligibility trace decay), and their overall influence on algorithmic performance:

- **Learning Rate (α):** Controls how much new information influences the Q-table updates, gradually overriding previous estimates.
  - A **low α** leads to slower learning but improves convergence stability.
  - A **high α** accelerates learning but may introduce instability and oscillations.
- **Discount Factor (γ):** Reflects the importance of future rewards in current decision-making.
  - A **high γ** encourages the agent to prioritize long-term rewards.
  - A **low γ** leads the agent to focus on immediate, short-term gains.

Table 5 Tripod gait actions (three legs are moving)

Time	L1	L2	L3	L4	L5	L6
T1	1	0	1	0	1	0
T2	0	1	0	1	0	1

Table 6 Tetrapod gait actions (two legs are moving)

Time	L1	L2	L3	L4	L5	L6
T1	1	0	0	0	0	1
T2	0	1	0	1	0	0
T3	0	0	1	0	1	0

Table 7 Wave gait actions (one leg is moving)

Time	L1	L2	L3	L4	L5	L6
T1	1	0	0	0	0	0
T2	0	1	0	0	0	0
T3	0	0	1	0	0	0
T4	0	0	0	1	0	0
T5	0	0	0	0	1	0
T6	0	0	0	0	0	1

Table 8 Ripple gait actions (one leg is moving)

Time	L1	L2	L3	L4	L5	L6
T1	1	0	0	0	0	0
T2	0	1	0	0	0	0
T3	0	0	1	0	0	0
T4	0	0	0	1	0	0
T5	0	0	0	0	1	0
T6	0	0	0	0	0	1

- **Eligibility Trace Decay (λ):** Controls how past experiences influence current learning updates.
  - A **higher λ** accelerates learning by distributing credit across multiple preceding actions, facilitating sequence learning.
  - A **lower λ** (approaching 0) makes the algorithm behave similarly to traditional one-step Q-learning, where only the most recent action significantly impacts the Q-value update.

4 Path planning and locomotion control of the hexapod robot

Path planning is fundamental to achieving reliable control for guiding a robot along a designated trajectory. This section introduces a key approach to controlling the hexapod robot, ensuring it adheres to its optimal path.

4.1 Control architecture

To deal with the walking robot control problem, we based on the multi-level control model proposed by Brooks (Porta and Celaya 2000; Singh 2017). Each level of control manages a specific part of the robot's operation. The different levels are: (*i*) the first level, which holds segments in position; (*ii*) the second level, which controls the gait; and (*iii*) the final level, which handles navigation control. Each level adds a distinct functionality, and together, they enable the global control of the robot.

The lowest level manages the change and keeping in position of the segments. In our robot the control of each segment is independent from the others by a specialized circuit incorporating a digital filter type PID, generating the control necessary for the tracking and regulation around these set-points. The second level is that of the gait control which assumes to generate the instructions of positions necessary to move the body in the space according to a controlled trajectory, while ensuring the robot's balance and posture, and adapting to terrain that may be uneven or may present obstacles. Level three is navigation, which includes environmental perception and analysis, the choice of objectives to be achieved, and the choice of approach strategy (Porta and Celaya 2000).

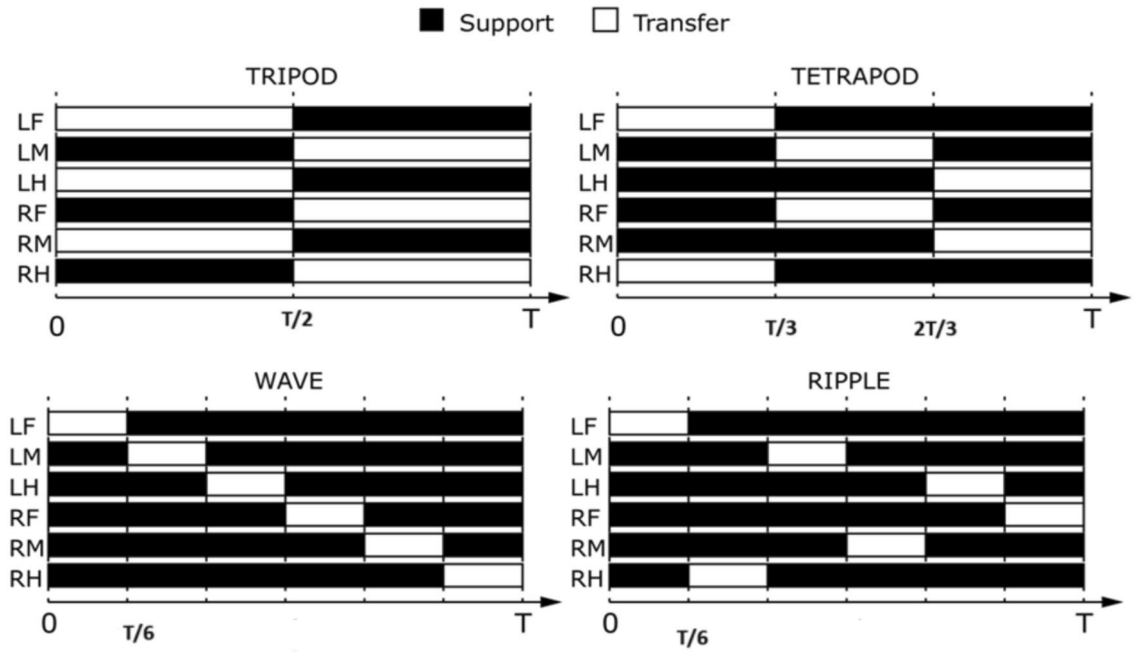

Fig. 7 Sequence diagrams of hexapod gaits: Tripod gait (top left), Tetrapod gait (top right), Wave gait (bottom left), and Ripple gait (bottom right)

Table 9 Eight initial poses of the robot's gravity centre

Poses	Angles and Y position
Pose 1	$\theta p = 0$ and $yp = 0$
Pose 2	$\theta p = \pi/4$ and $yp = 1.5$
Pose 3	$\theta p = \pi/2$ and $yp = 0.5$
Pose 4	$\theta p = 3\pi/4$ and $yp = 0.5$
Pose 5	$\theta p = \pi$ and $yp = 0$
Pose 6	$\theta p = -3\pi/4$ and $yp = -0.5$
Pose 7	$\theta p = -\pi/2$ and $yp = -0.5$
Pose 8	$\theta p = -\pi/4$ and $yp = -1.5$

4.2 Gait generation

The hexapod robot utilizes three stable gaits, with undercarriage stability being the primary requirement for effective walking control (Ivo and Patrik 2005). To maintain stability during locomotion, steady-state stability states were defined (Celaya and Porta 2000) as shown in Tables 5, 6, 7, 8 and the Fig. 7 is presented the sequence diagram of all gaits.

Where: L1 to L6 represent the robot's legs, corresponding to the same order shown in Fig. 1. T1, T2, indicate the time when each leg is either on the ground (stance phase), coded as 1, or in the air (transfer phase), coded as 0 like as presented in Fig. 9.

In our simulation, we used the tripod gait (Table 5) because it allowed the robot to move both effectively and faster compared to other gaits. The stability configuration of the robot using the tripod gait depends on the position of its centre of gravity (CoG) relative to the stability triangle. When the CoG lies within the centroid of the stability triangle (Fig. 9 a), the robot remains stable (see (Zennir and Couturier 2005)). Conversely, if the CoG falls outside the stability triangle (Fig. 9 b), the robot becomes unstable.

Foot 1, Foot 2, and Foot 5 are the end-effectors of legs 1, 3, and 5, respectively, which remain on the ground during the stance phase of the first step of the tripod gait. In the second step, legs 2, 4, and 6 are used instead (Fig. 1 illustrates the leg numbering).

4.3 Path planning

This paper addresses the challenge of enabling the robot to adjust its trajectory effectively. The objective is to guide the robot starting from various initial positions and orientations (eight poses) towards alignment with a predefined straight path, while ensuring that the lateral deviation error remains within the acceptable range Ey, and the rotational error around the vertical axis Gz stays within the threshold Eθ, as illustrated in Fig. 9.

In our simulation, we evaluated the robot's performance across eight distinct initial positions and orientations, as outlined in Table 9 and depicted in Fig. 10. The resulting robot trajectories, ranging from Trajectory 1 to Trajectory 8, correspond to the initial positions and orientations numbered 1 through 8 in Table 9. The maximum Y-coordinate (*Ymax*) was set to 2 m, while the minimum (*Ymin*) was −2 m. The acceptable distance error for lateral deviation, Ey (Eq. 9), was within the range [−0.1, 0.1] meters, and the acceptable orientation error, Eθ (Eq. 10), was constrained to [−0.1, 0.1] radians.

Table 10 The rewards values of the RL algorithms

Penalty: $r = -1$ if	Reward: $r = 10$ if
Following a proactive decision, the robot falls	The protraction movement occurs normally (no fall)
The protraction command is repeated twice in a row	The retraction movement has exceeded the minimum duration
The retraction movement has a duration that is either too long or too short	
All the legs are on the ground	

Table 11 The reward values of the RL algorithms

Parameter	Value	Parameter	Value
T (s)	0.5	L_{ci}(m)	0.05
β	0.5	L_{fi}(m)	0.1
L_S (m)	0.2	L_{ti}(m)	0.2
S_P (m)	0.6	O_i (m)	0.0
H_B (m)	0.4	VR (m/s)	0.4
F_C (m)	0.2		

Table 12 Variation of SARSA learning parameters across various experiments

Experiment	α	γ	λ
Experiment 1	0.01	0.5	0
Experiment 2	0.01	0.9	0.9
Experiment 3	0.05	0.5	0
Experiment 4	0.05	0.9	0.9
Experiment 5	0.1	0.3	0.1

$$E_y = Y_{des} - Y_{cur} \tag{9}$$

$$E_\theta = \theta_{des} - \theta_{cur} \tag{10}$$

where.

Y_{des} and θ_{des} represent the desired position on the Y-axis and the desired orientation of the robot's centre of gravity, respectively. In our case, both are equal to zero. Y_{cur} and θ_{cur} denote the current position on the Y-axis and the current orientation of the robot during motion.

The robot was placed in one of eight possible initial positions and orientations, and the task was to learn how to reach the target position. Two motion equations were used one for translation and one for rotation which defined the available actions in the Q-learning and SARSA algorithms. The control strategy for reaching the target consisted of applying these two actions: a rotation adjustment and a translation step, as formulated in Eq. 6. In each training episode, the robot began at one of the eight initial poses (Table 9) and aimed to reach the target position located at the origin with zero orientation error. Learning proceeded iteratively until the position and orientation errors satisfied the convergence criteria ($Ey \in [-0.1, 0.1]$ m and $E\theta \in [-0.1, 0.1]$ rad) or until the maximum number of episodes was reached. These thresholds defined successful convergence to the target. Once the robot completed a pose scenario (e.g., starting from Pose 1 until reaching the target), it was reset to another initial pose, and the learning process continued until all eight poses were completed. This procedure was repeated for all five experiments (Tables 10 and 11). Further details of the simulation method are provided in Sect. 5

$$\mathbf{Tg} = \begin{bmatrix} \cos(\theta g) & -\sin(\theta g) & 0 \\ \sin(\theta g) & \cos(\theta g) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

Each agent seeks to maximize its own objective function, which involves repeating a locomotion cycle consisting of a transfer phase followed by a stance phase. The agent is penalized if a protraction lasts more than one time unit. The agent is also penalized if the retraction duration is too short or too long. The retraction duration is dictated by the higher control level (level 3), where the gait is selected. In the event of a fall, only the agents responsible are penalized. If two consecutive legs lift simultaneously, the robot tips to that side, triggering the fall detection sensor and imposing a local penalty on the legs that caused the fall. If all legs are on the ground, this represents a resting position, which we have chosen to penalize (at a local level, this situation could be detected by each leg through a force sensor, with each leg bearing the least load). Thus, the rules outlined only reflect local behaviours or phenomena that can be perceived by the agents and do not result from a global analysis of the movement sequence resembling walking. We assume that maximizing the gains of each agent in achieving their own objectives is compatible with the overall objective of making the robot walk. The reward function was designed to penalize undesirable events (e.g., falls, improper leg timing) and to reward successful forward movement. Table 12 summarizes these rules. In essence, a large positive reward (+10) is given when a leg's protraction completes without causing a fall, and smaller penalties (−1) are assigned for events like the robot tipping over, legs colliding (two consecutive legs lifted), or the robot entering a rest state with all legs on ground.

5 Simulation results and discussion

We can divide the parameters used in our simulation into two categories: those for the robot model and those specifically for the algorithms. In our simulation, we applied distributed Q-learning (λ) and distributed SARSA (λ). When we refer to Q-learning and SARSA below, we are specifically referring to distributed Q-learning (λ) and distributed SARSA (λ). The simulation results were obtained using MATLAB.

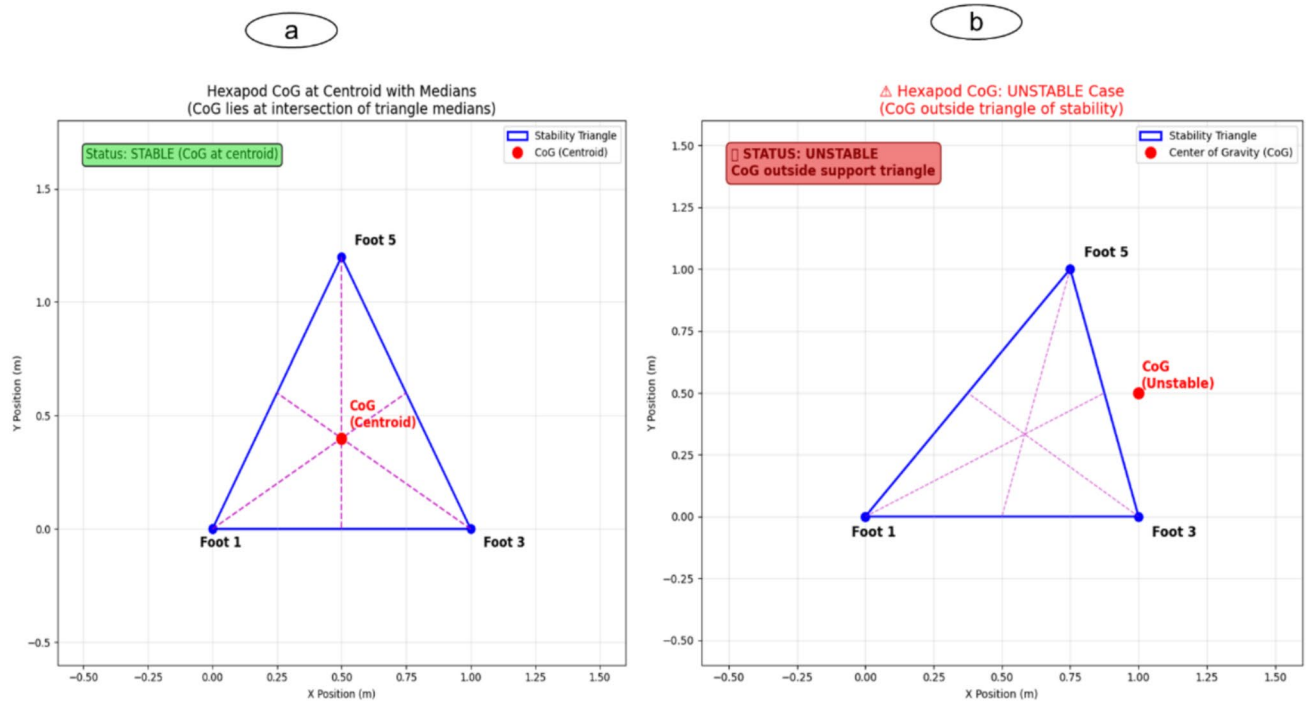

Fig. 8 Stability margin of the robot when using the tripod gait: (**a**) stable case and (**b**) unstable case

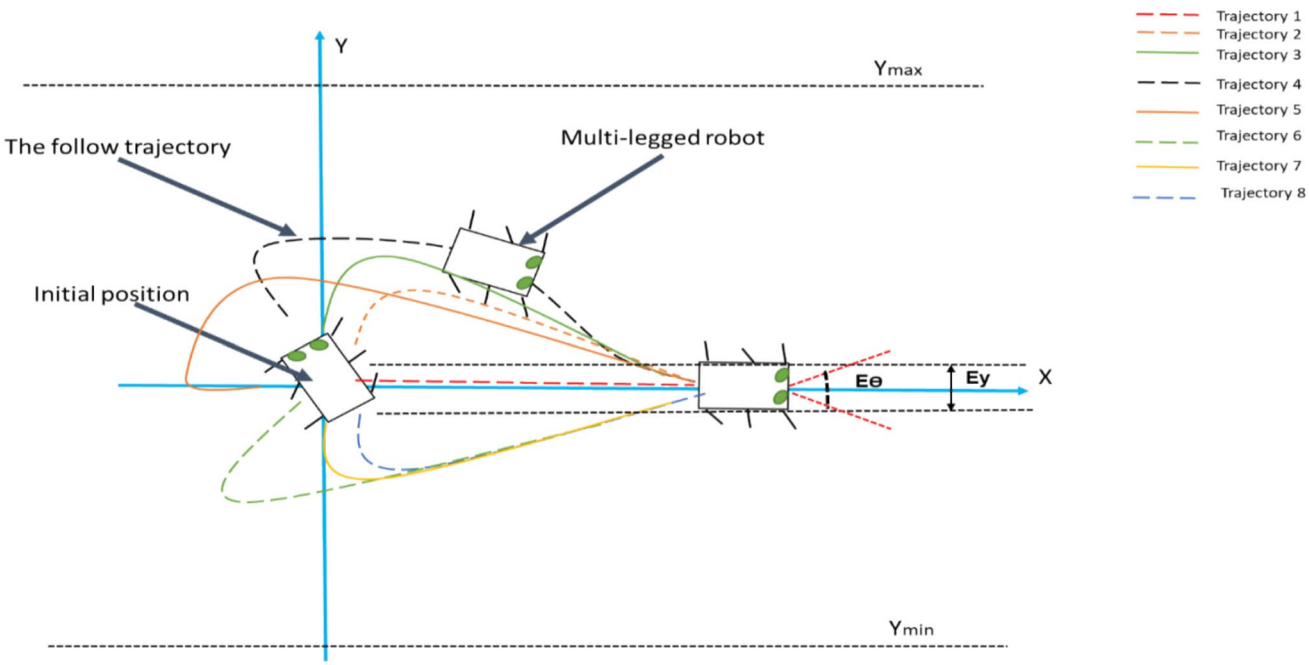

Fig. 9 The robot learning to change its trajectory from different initial poses

The parameter values of the robot used in this simulation, described in Sect. 2 (Table 2 and Fig. 2), are summarized in Table 11.

The simulation environment was configured by placing the robot in eight different initial positions and orientations, as detailed in Table 9 and illustrated in Fig. 10. The Q-learning and SARSA reinforcement learning (RL) algorithms were employed to guide the robot's trajectory toward a designated target position, as shown in Fig. 8 and described in Sect. 4. A series of experiments were conducted, each using distinct parameter sets for the SARSA and Q-learning algorithms, presented in Tables 12 and 13,

Table 13 Variation of learning parameters across various experiments using with Q-Learning algorithm

Experiment	α	γ	λ
Experiment 1	0.05	0.5	0
Experiment 2	0.05	0.9	0.3
Experiment 3	0.05	0.9	0.9
Experiment 4	0.1	0.3	0.1
Experiment 5	0.1	0.5	0.9

respectively. In these experiments, key learning parameters, including α, γ, and λ, were systematically varied, as illustrated in Fig. 11 within the MATLAB control interface. The selection of the learning parameters was based on empirical tuning, as clearly described by Zennir et al. [ref(a)] (Zennir 2004).

The training phase of the Q-learning and SARSA algorithms was implemented in C+ +, and the resulting pose curves and robot walking trajectories were subsequently visualized and analysed in MATLAB (Fig. 11).

Simulation Method: In each experiment, the robot learns to reach the target from a given initial pose. After completing the scenario from Pose 1, the robot is reset to another initial pose (e.g., Pose 2) and repeats the task to reach the target. This procedure is continued sequentially through Pose 8. Once all eight poses are completed, the learning parameters are reset for Experiment 2, and the same process is repeated for each initial pose. This cycle is carried out until all experiments are completed. The methodology is first applied using SARSA and then repeated with Q-learning.

5.1 Simulation results

The results from various experiments using the SARSA and Q-learning algorithms are presented as follows. These include the individual performance outcomes of the SARSA and Q-learning algorithms, as well as a comparative analysis highlighting the key differences between them.

5.1.1 Results of the SARSA algorithm

This section presents, in Sect. 5.1.1.1, the results obtained from using the SARSA algorithm for the position and orientation of the robot's centre of gravity during all experiments with the eight initial poses, along with the number of iterations for each experiment cycle. Additionally, Sect. 5.1.1.2 illustrates the joint angle positions when the robot completed the cycle for each of the eight initial poses.

5.1.1.1 Positions and orientations results Figures 12 through 21 depict the trajectories of the robot's centre of gravity from various initial positions and orientations to the target position over multiple iterations, using the SARSA algorithm. The red curve corresponds to Pose 1, the blue curve to Pose 2, while the yellow, green, olive green, black, pink, and light blue curves represent Poses 3 through 8, respectively. These initial poses are detailed in Fig. 10 and Table 9. This color-coded scheme is consistently applied across all experimental results. Figs. 12 13, 14, 15, 16, 17, 18, 19, 20, 21 present the outcomes of the SARSA algorithm under different experimental configurations.Specifically, Fig. 12, Fig. 14, Fig. 16, Fig. 18, and Fig. 20 illustrate

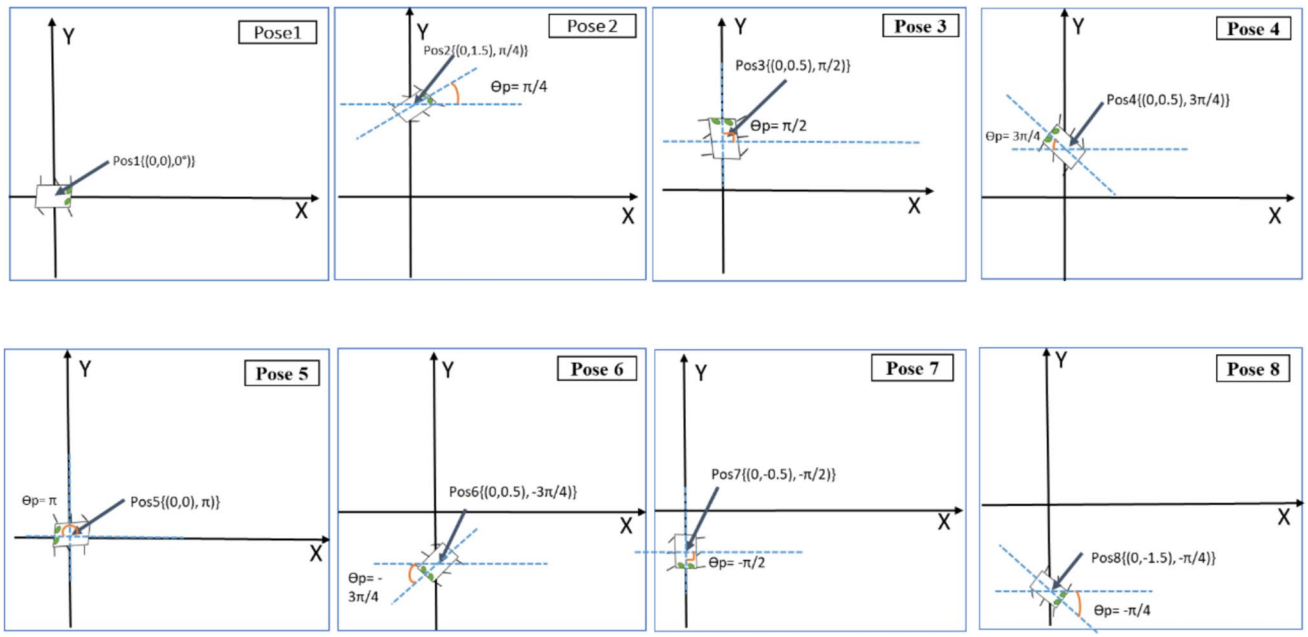

Fig. 10 Different initial positions and orientations of the robot

the evolution of the robot's centre of gravity position, while Fig. 13, Fig. 15, Fig. 17, Fig. 19, and Fig. 21 show changes in orientation, both plotted against the number of iterations. These results correspond to the five experiments described in Table 12.

In general, using the SARSA algorithm, the robot was able to reach the target from all initial poses smoothly across different experiments. Although oscillations were observed at the end of some poses, they were very small and remained within the acceptable interval [−0.1, 0.1]. We also observed slight variations in the position and orientation curves between different experiments. A more detailed explanation is provided in the Discussion section.

5.1.1.2 Joints angle of the robots The joint angle trajectories of the robot's legs during the complete cycle from each of the eight initial poses to the target are shown in Figs. 22, 23, 24, 25, 26, 27, 28, 23, corresponding to Poses 1 to 8, respectively. In all figures, the red curve denotes the coxa angle, the blue curve the femur angle, and the yellow curve the tibia angle, as illustrated in Figs. 1 and 2 of Sect. 2. Since all experiments produced identical joint angle patterns, only the results from Experiment 2 are presented.

Across eight locomotion poses, the joint angle trajectories of a hexapod leg—coxa, femur, and tibia—reveal distinct modulation patterns during forward, turning, and lateral movements. During straight walking (Pose 1), all joints exhibit stable, periodic oscillations at ≈10 Hz, with consistent amplitudes and minimal transients, reflecting a smooth gait. As rotational and lateral manoeuvres are introduced (Poses 2–8), increasing complexity emerges: the coxa shows irregular oscillations, phase shifts, pauses, and spikes, particularly during larger turns (e.g., 135° and 180°), indicating adaptive steering control. The femur maintains a near-constant oscillation frequency (~10 Hz) across all conditions but exhibits transient amplitude drops during gait transitions (e.g., at $t = 60$, 120, or 140 s), suggesting dynamic load redistribution. The tibia displays the most variability, with segmented activity, frequency modulation, and periods of zero motion especially during turns indicating selective foot engagement for balance and directional control. Overall, the coxa adapts most to orientation changes, while the femur provides rhythmic stability, and the tibia fine-tunes ground contact, demonstrating a robust, context-dependent locomotion strategy.

A more detailed discussion is provided in Sect. 5.2.3.

5.1.2 Results of the Q-learning algorithm

This section, in Sect. 5.1.2.1, presents the results of applying the Q-Learning algorithm to analyse the position and orientation of the robot's centre of gravity across all experiments with the eight initial poses, as well as the corresponding number of iterations for each experiment cycle. Section 5.1.2.2 then illustrates the joint angle variations observed when the robot completed the cycle for each initial pose.

5.1.2.1 Positions and orientations results Figures 30, 31, 32, 33, 34, 35, 36, 37, 38, 39 present the trajectory results of the gravity centre of the robot using the Q-learning algorithm, starting from eight different initial positions and orientations and progressing toward the target position over multiple iterations. The red and blue curves represent trajectories from Poses 1 and 2, respectively, while the yellow, green, olive green, black, pink, and light blue curves correspond to Poses 3 through 8. The details of these initial configurations are provided in Fig. 10 and Table 9, while Table 13 presents the learning parameters of Q-learning for each experiment.

This colour scheme is consistently used across all experiments.

Specifically, Fig. 30, Fig. 32, Fig. 34, Fig. 36, and Fig. 38 show the progression of the centre of gravity position, whereas Fig. 31, Fig. 33, Fig. 35, Fig. 37, and Fig. 39 illustrate the evolution of orientation, both as functions of the number of iterations. These figures correspond to the five experimental setups outlined in Table 13.

Overall, the Q-Learning algorithm enabled the robot to reach the target from all initial poses with smooth trajectories across different experiments. Minor variations in the position and orientation curves were observed between experiments. Further analysis of these results is presented in the Discussion section.

In summary, Q-Learning trials generally converged faster and achieved slightly better trajectory accuracy than SARSA trials under equivalent settings, although exceptions

Table 14 The value of "*Ey*" of each pose crossing all experiments using SARSA algorithm

	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8
Experiment 1	−0.0404	0.0017	−0.0020	0.0219	−0.0195	0.0779	0.1165	−0.0058
Experiment 2	0.0077	0.0178	0.0662	0.0699	−0.0041	0.0014	−0.0064	-eq0.0005
Experiment 3	−0.0433	0.0072	−0.0320	−0.0484	−0.0293	−0.050	−0.0517	−0.0046
Experiment 4	0.0136	0.0095	0.0051	0.0092	0.0078	−0.0161	−0.0030	−0.0028
Experiment 5	0.0078	0.0033	0.009	0.0002	0.028	0.0483	0.0411	0.087

Table 15 The value of "$E\Theta$" of each pose crossing all experiments using SARSA algorithm

	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8
Experiment 1	0.003	−0.0063	−0.0061	−0.0367	0.0197	0.0093	0.0157	0.0238
Experiment 2	−0.0076	−0.0013	0.0245	−0.0087	−0.0072	−0.007	0.0057	−0.0061
Experiment 3	0.0013	−0.003	0.027	0.0209	−0.0222	0.0025	0.001	−0.0373
Experiment 4	0	0.0128	0.009	0.0039	0.0105	−0.0215	0.0037	0.0093
Experiment 5	−0.0111	0.0027	−0.0171	0.0371	0.0065	0.0047	0.0023	−0.0323

Table 16 The number of iterations of each pose crossing all experiments using SARSA algorithm

	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8
Experiment 1	51	107	149	166	166	160	126	87
Experiment 2	51	110	147	164	151	147	114	83
Experiment 3	51	137	201	203	200	145	101	78
Experiment 4	51	238	226	196	100	122	94	76
Experiment 5	51	141	173	223	152	180	118	88

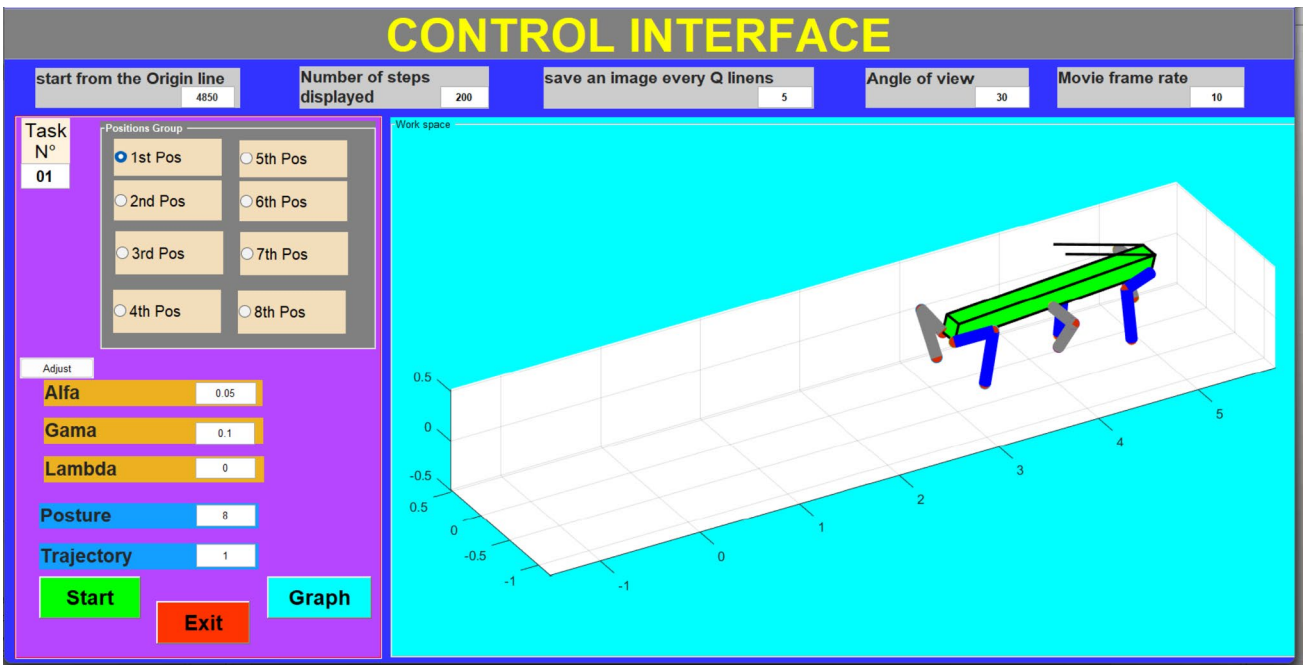

Fig. 11 MATLAB control interface for hexapod robot locomotion simulation

occurred in certain configurations (detailed comparisons are provided inSect. 5.2.3).

5.1.2.2 Joint angles results When the joint angles results of SARSA algorithm is the same results of Q-Learning, we get enough by one results in Sect. 5.1.1.2.

5.2 Discussion

This section discusses the results obtained from applying the SARSA (State-Action-Reward-State-Action) and Q-learning algorithms to control the trajectory of a hexapod robot. The performance of both algorithms was evaluated through a series of experiments involving variations in key learning parameters, including α, γ, and λ, as detailed in Tables 12, 13. The corresponding results are presented in Tables (14, 15, 16) and Figs. 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, which illustrate the robot's behaviour under different experimental conditions.

The analysis focuses on three primary performance metrics:

- **Ey**: the lateral deviation of hexapod gravity centre from the desired trajectory,
- **E0**: the orientation error relative to the desired heading,
- **Number of iterations**: the steps required to reach the target position or achieve convergence.

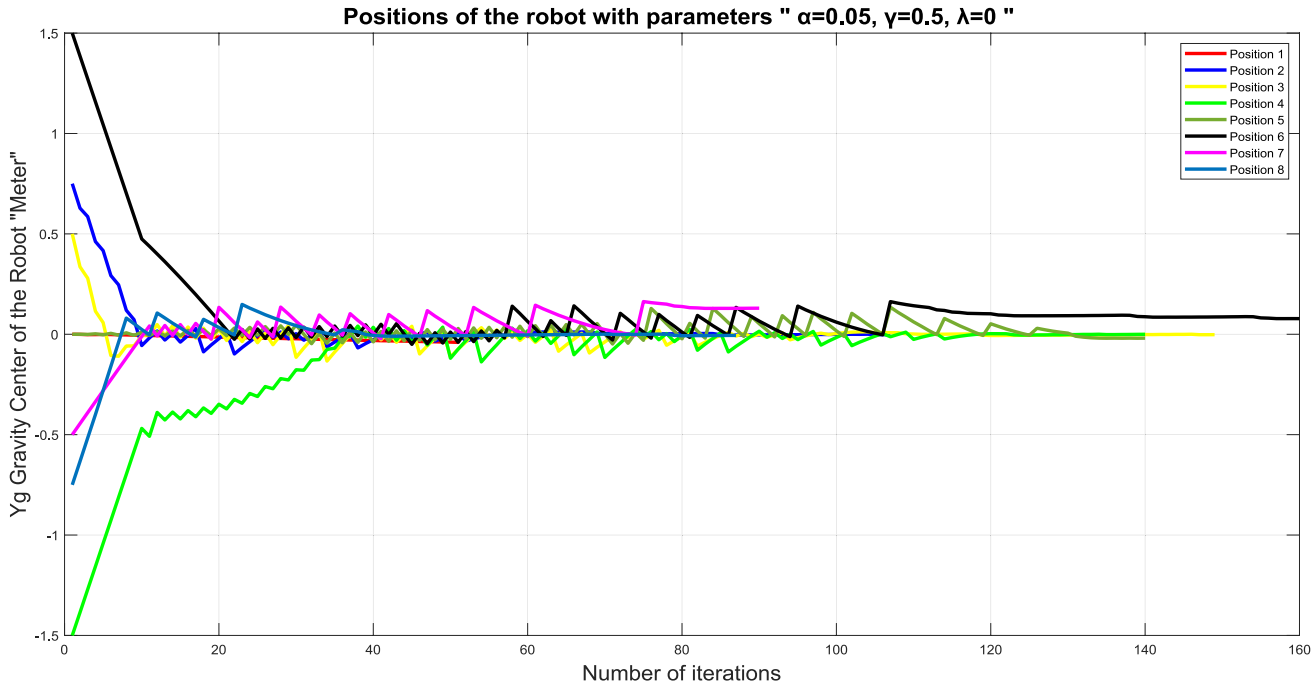

Fig. 12 Position Changes in the gravity centre of the robot using the SARSA algorithm for Experiment 1

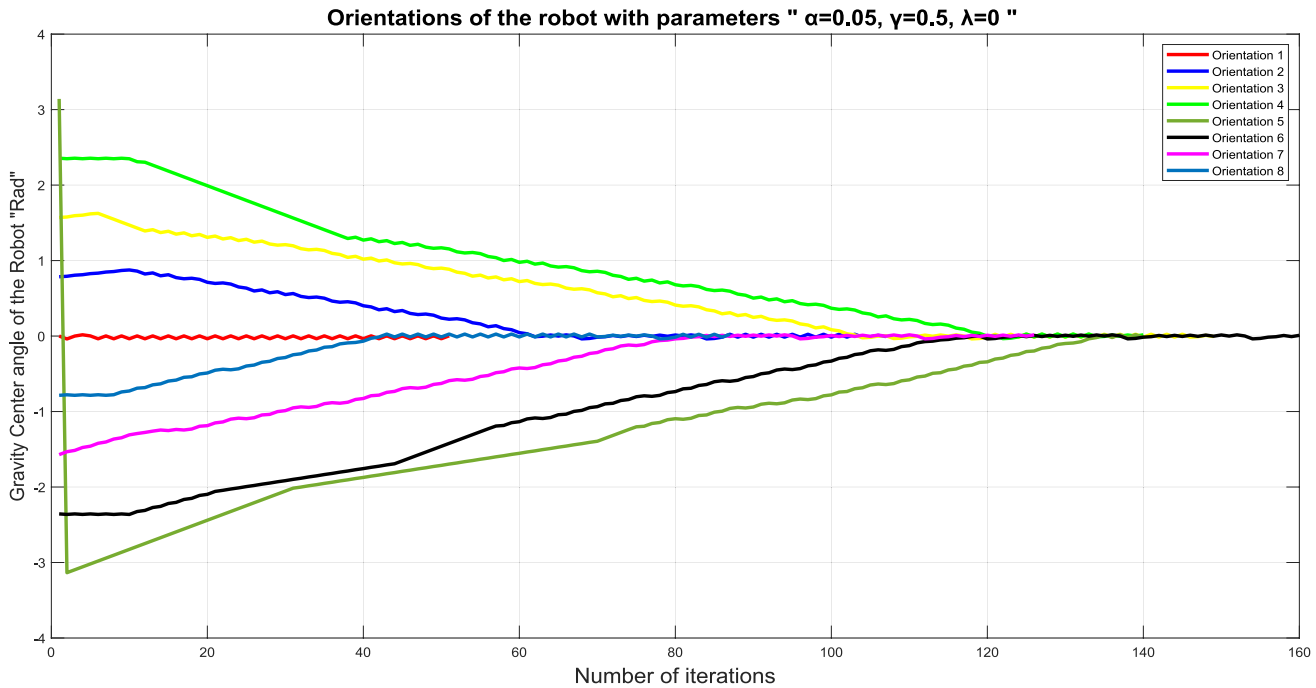

Fig. 13 Orientationchanges in the gravity centre of the robot using the SARSA algorithm for Experiment 1

These metrics, described in Sect. 4.3, serve as the basis for assessing the accuracy, stability, and efficiency of the SARSA and Q-learning algorithms in guiding the robot's locomotion.

5.2.1 Discussion of SARSA results

This section analyses the performance of the SARSA algorithm in controlling the trajectory of a hexapod robot. The evaluation is based on three key performance metrics assessed across eight target poses and five experimental

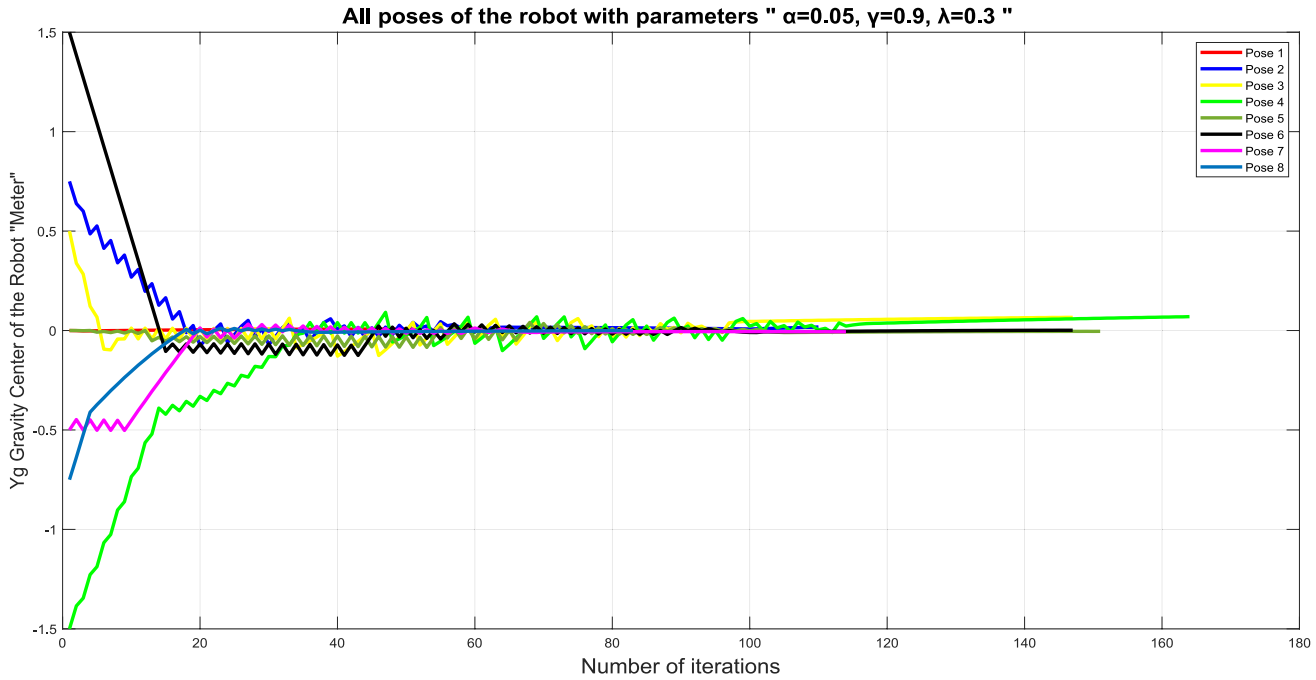

Fig. 14 Position Changes in the gravity centre of the robot using the SARSA algorithm for Experiment 2

SARSA

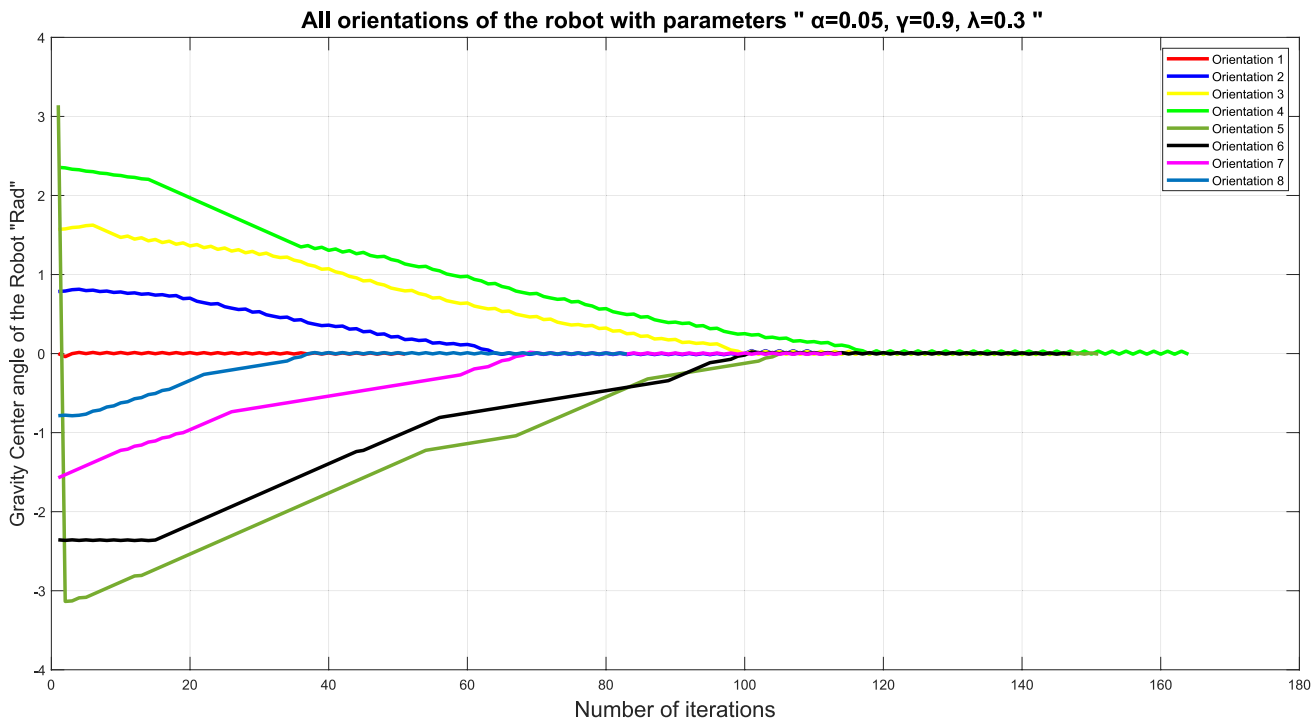

Fig. 15 Orientation changes in the gravity centre of the robot using the SARSA algorithm for Experiment 2

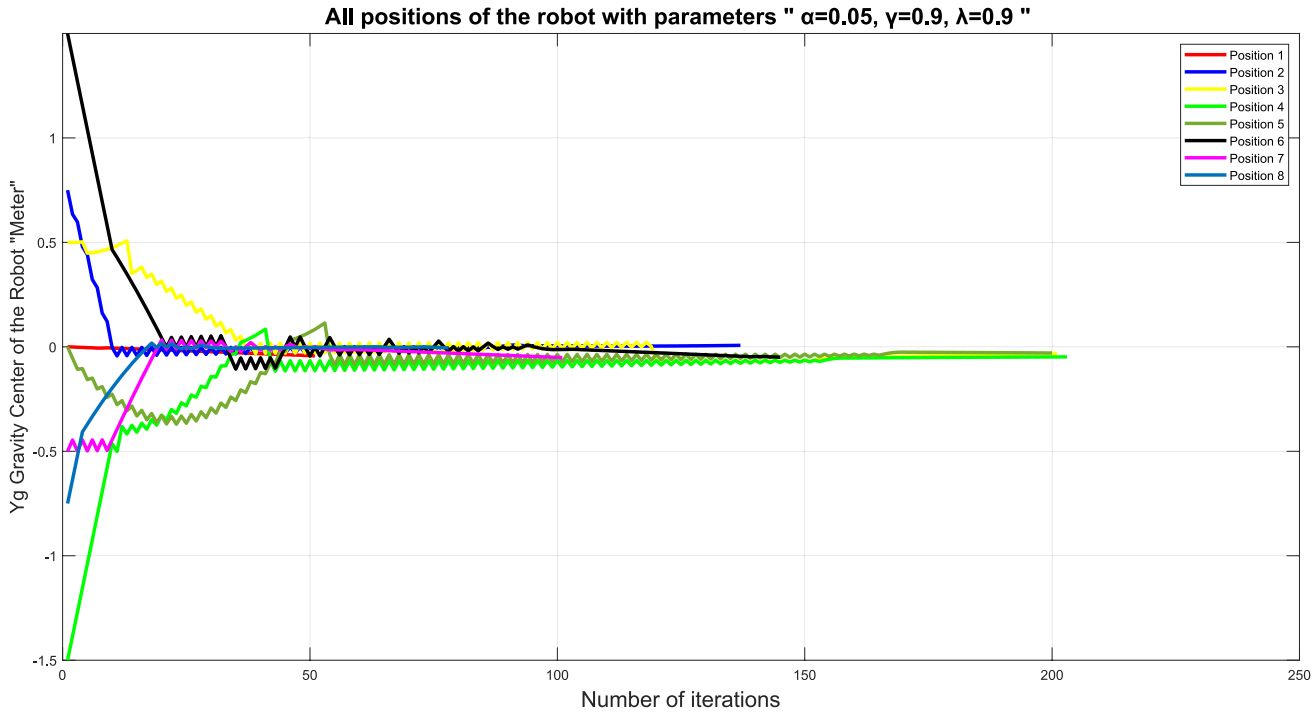

Fig. 16 Position changes in the gravity centre of the robot using the SARSA algorithm for Experiment 3

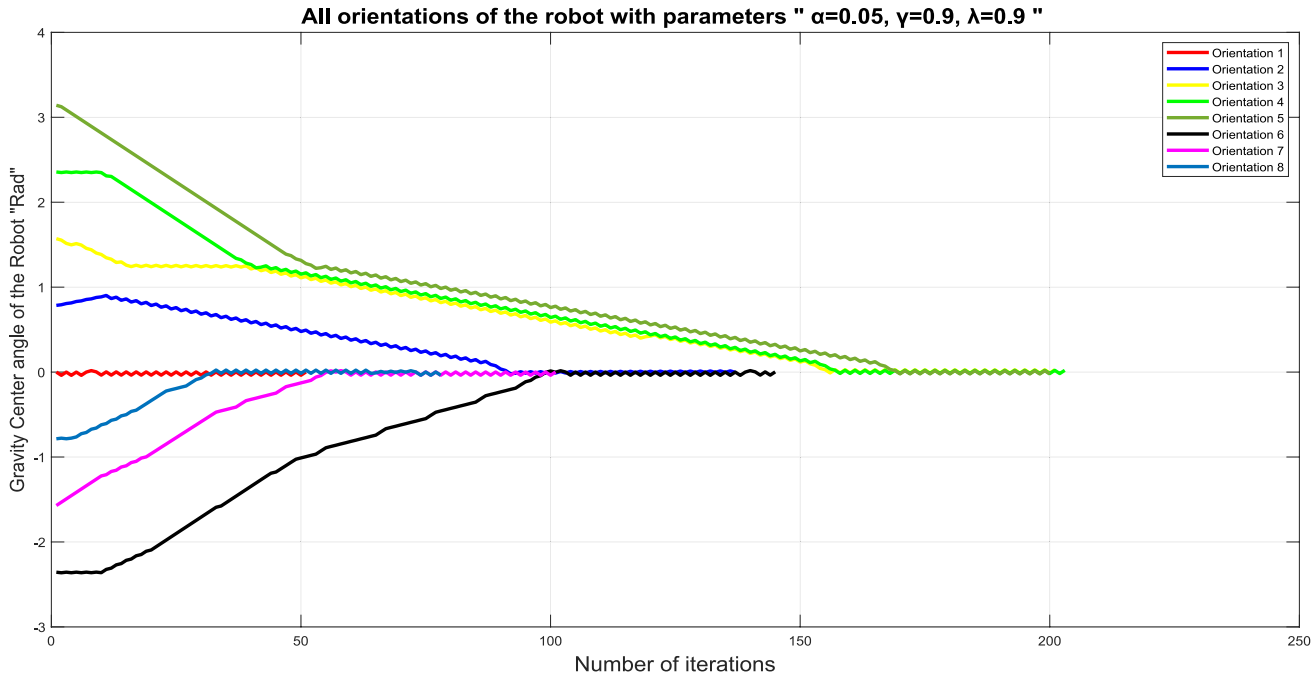

Fig. 17 Orientation changes in the gravity centre of the robot using the SARSA algorithm for Experiment 3

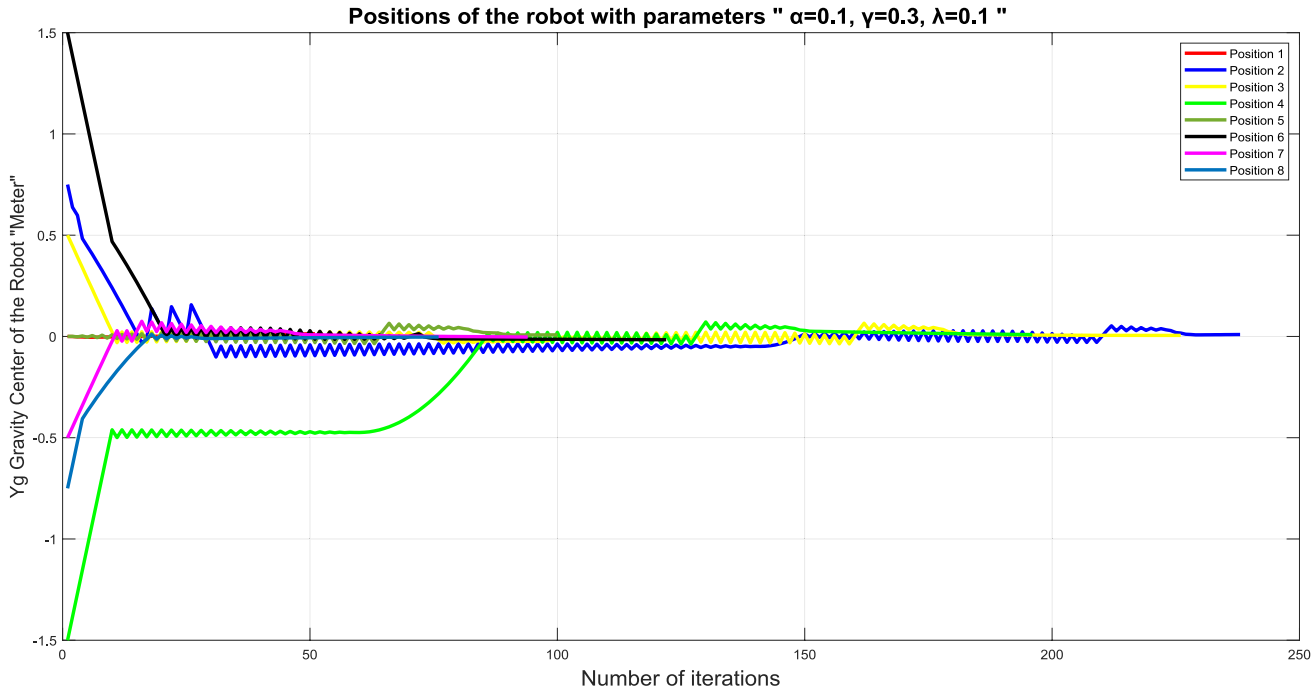

Fig. 18 Changes in the gravity centre of the robot trajectory using the SARSA algorithm for Experiment 4

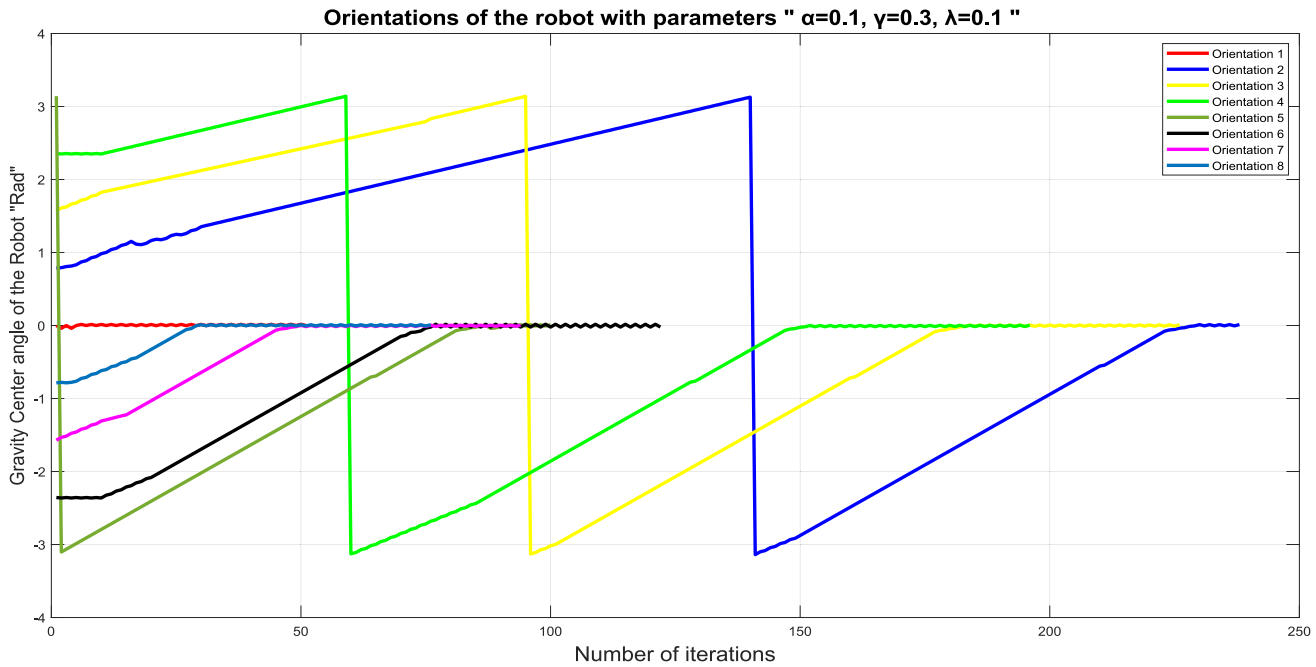

Fig. 19 Orientation changes in the gravity centre of the robot using the SARSA algorithm for Experiment 4

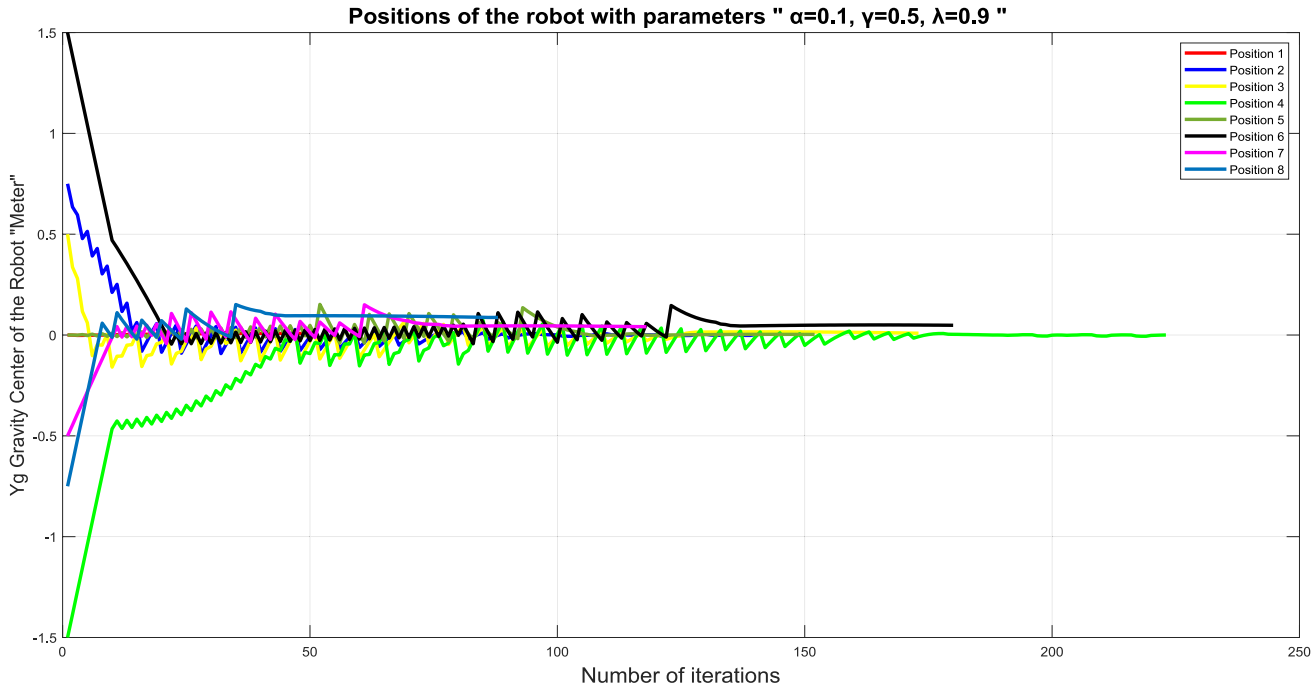

Fig. 20 Changes in the gravity centre of the robot trajectory using the SARSA algorithm for Experiment 5

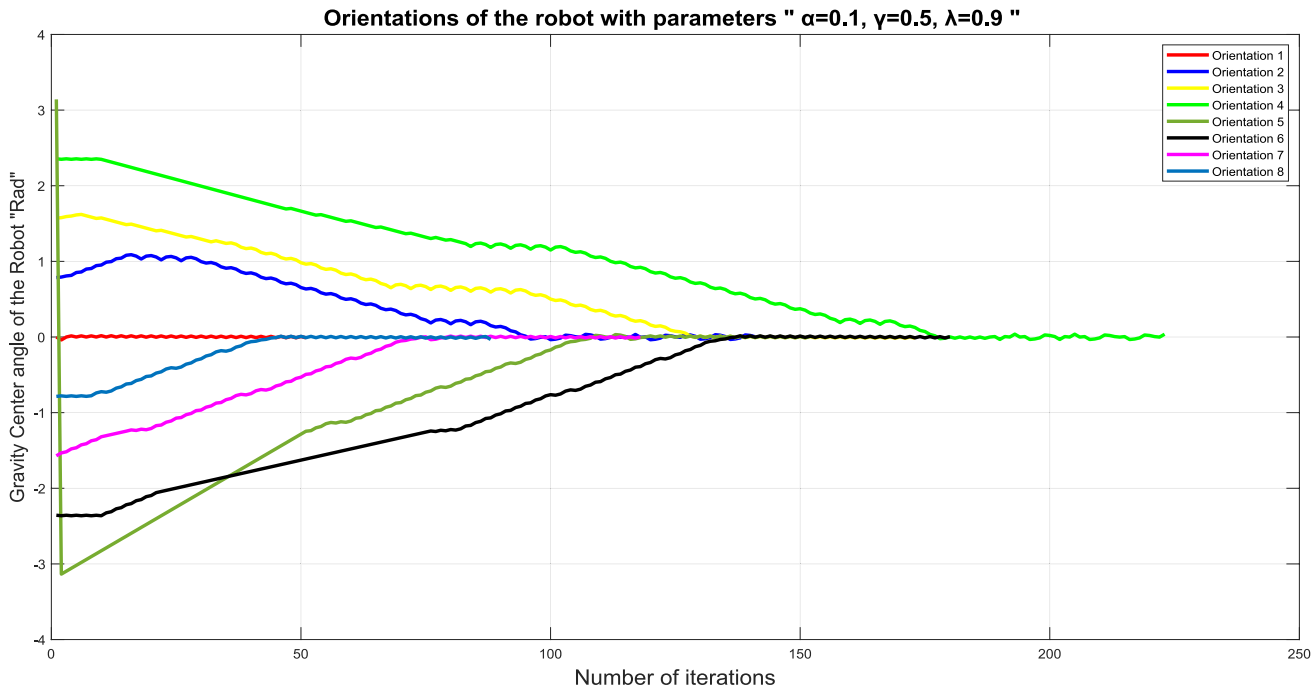

Fig. 21 Orientation changes in the gravity centre of the robot using the SARSA algorithm for Experiment 5

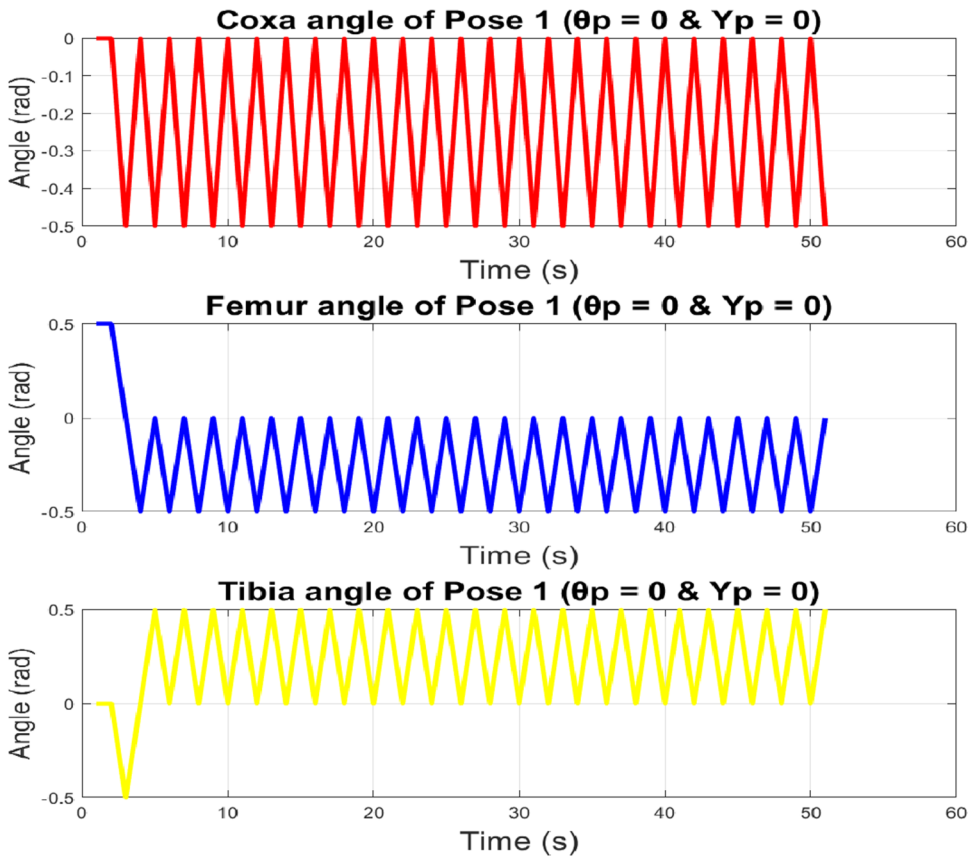

Fig. 22 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 1

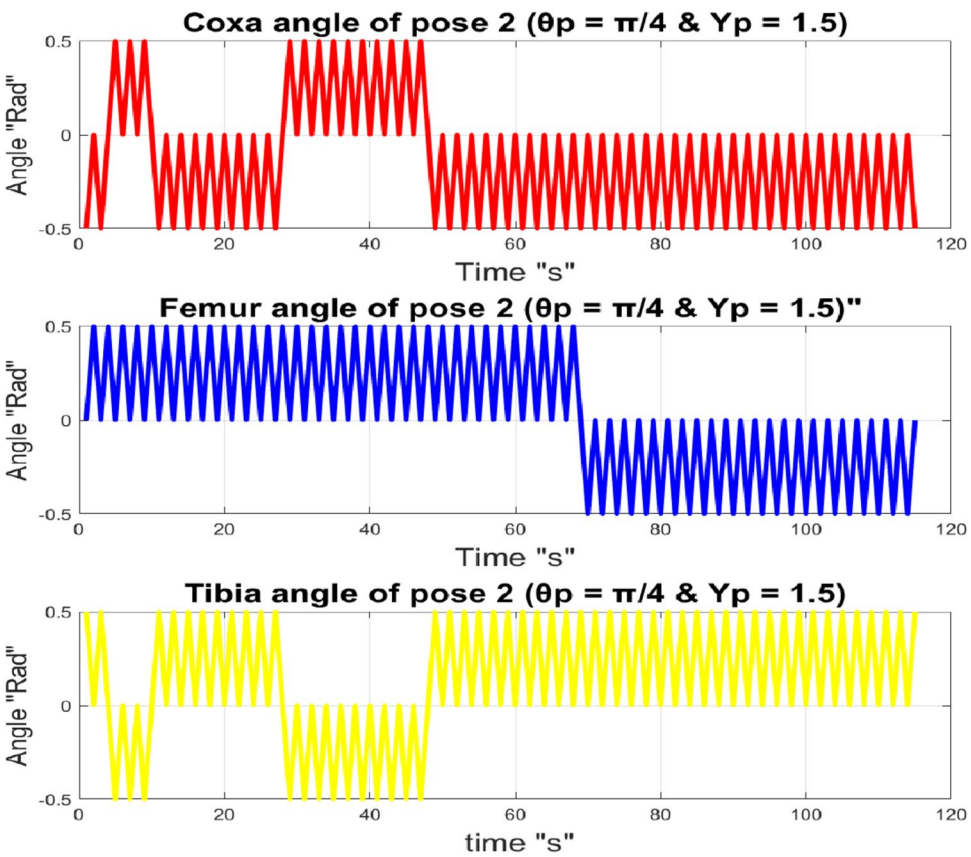

Fig. 23 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 2

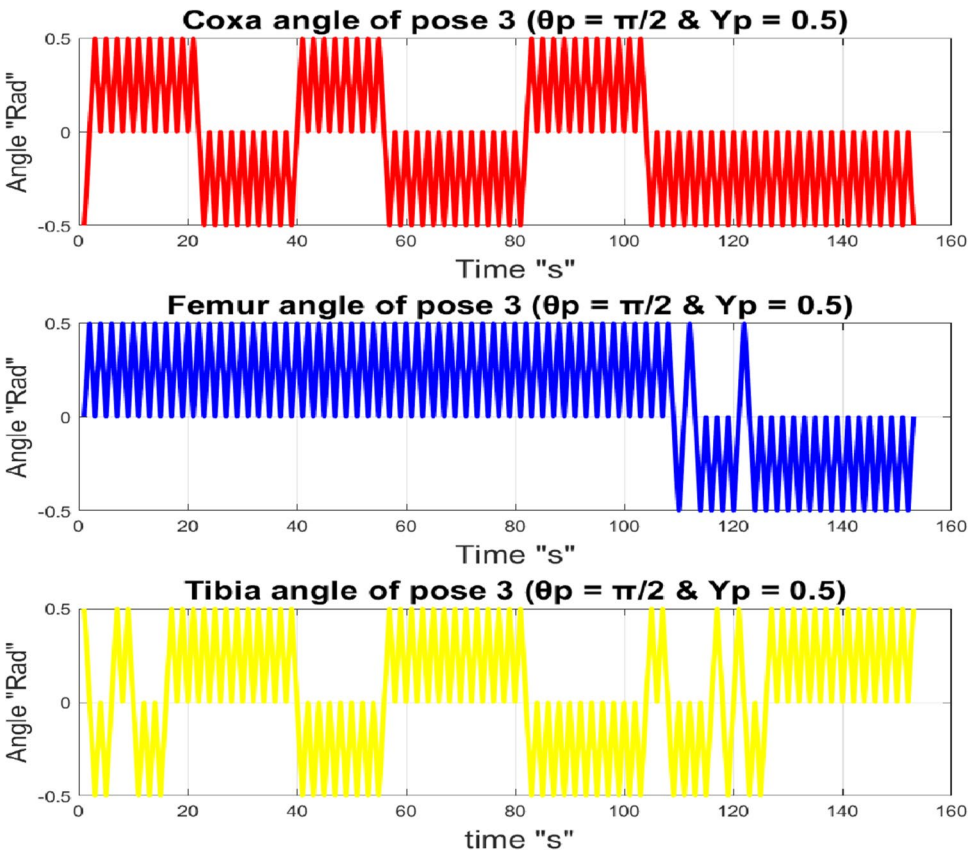

Fig. 24 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 3

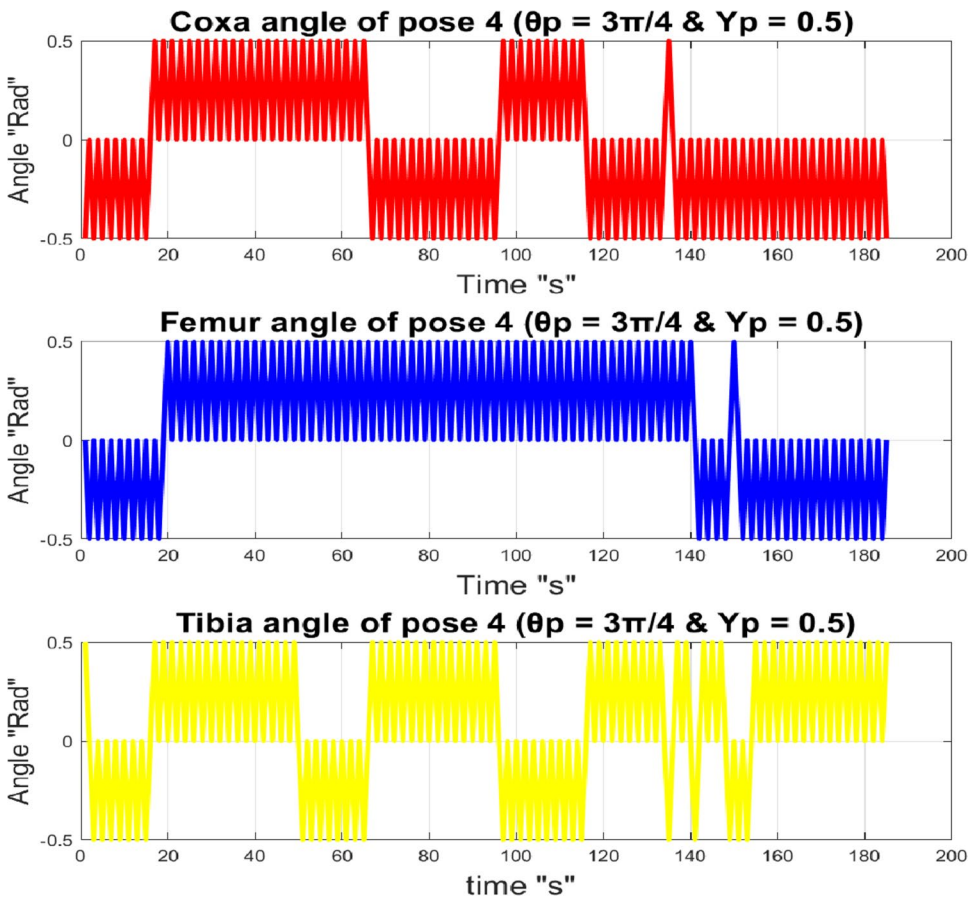

Fig. 25 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 4

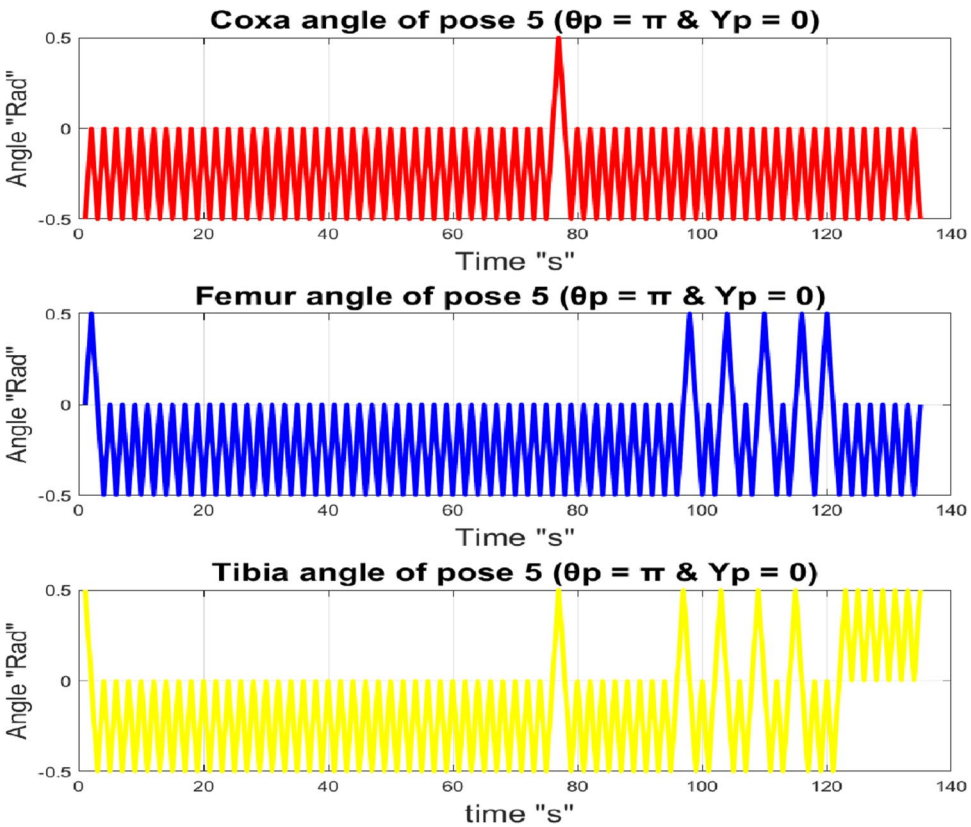

Fig. 26 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 5

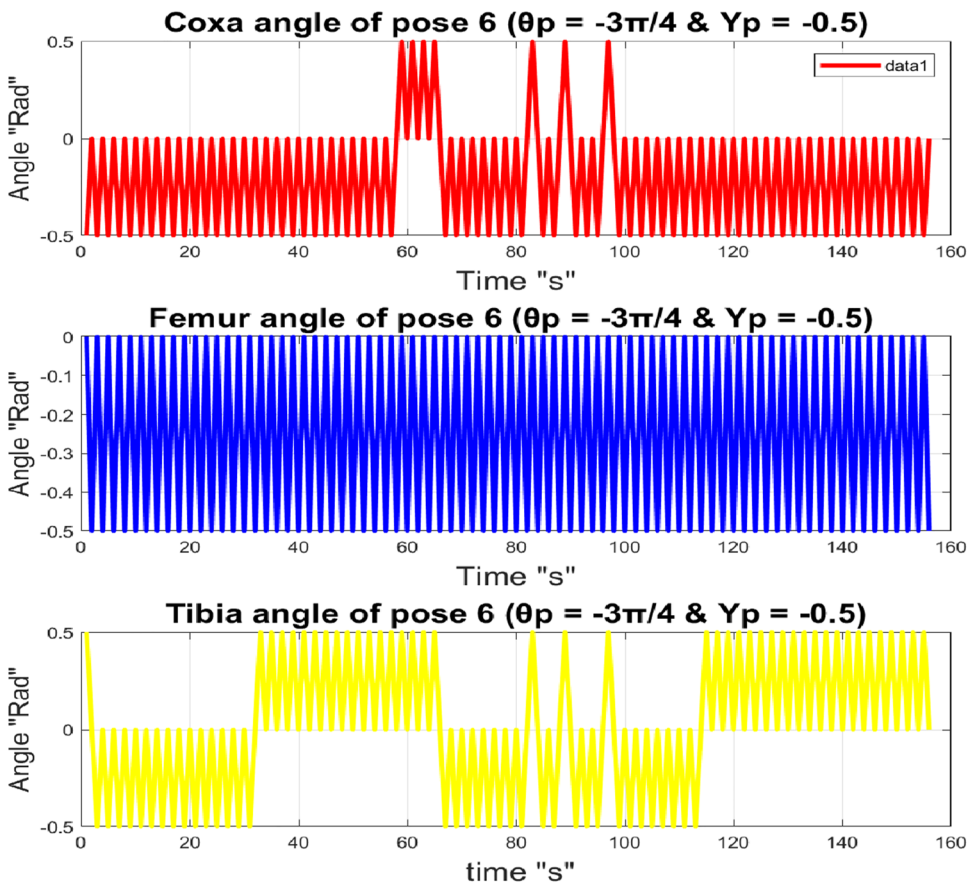

Fig. 27 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 6

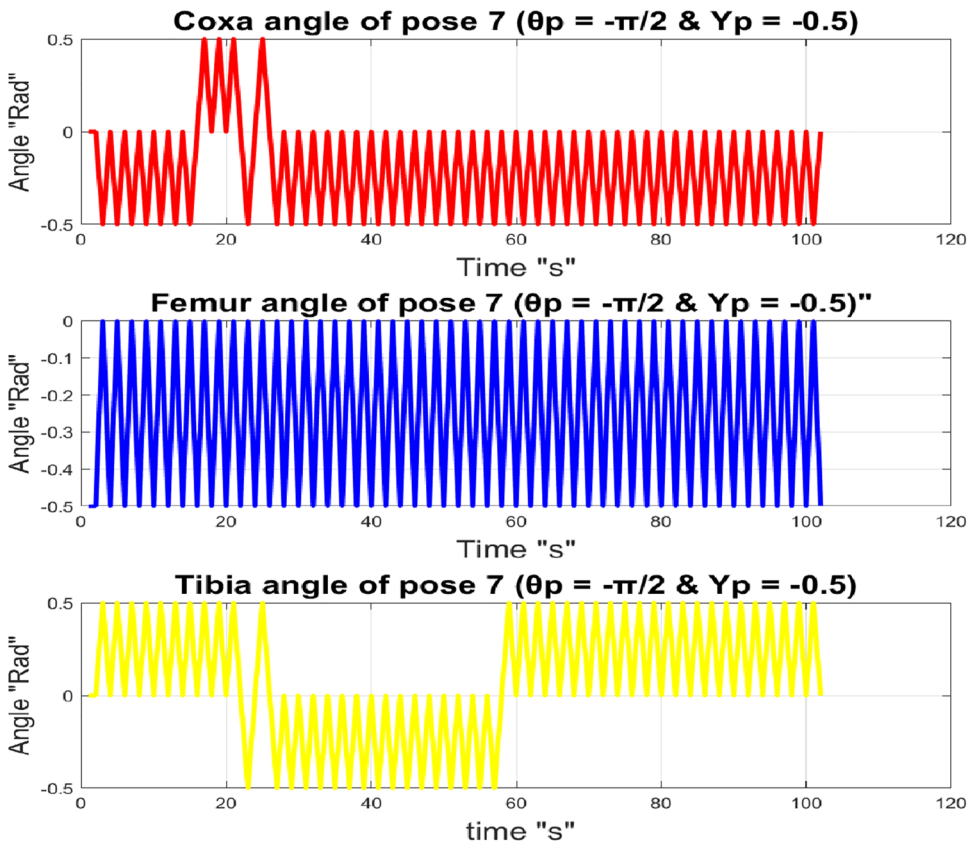

Fig. 28 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 7

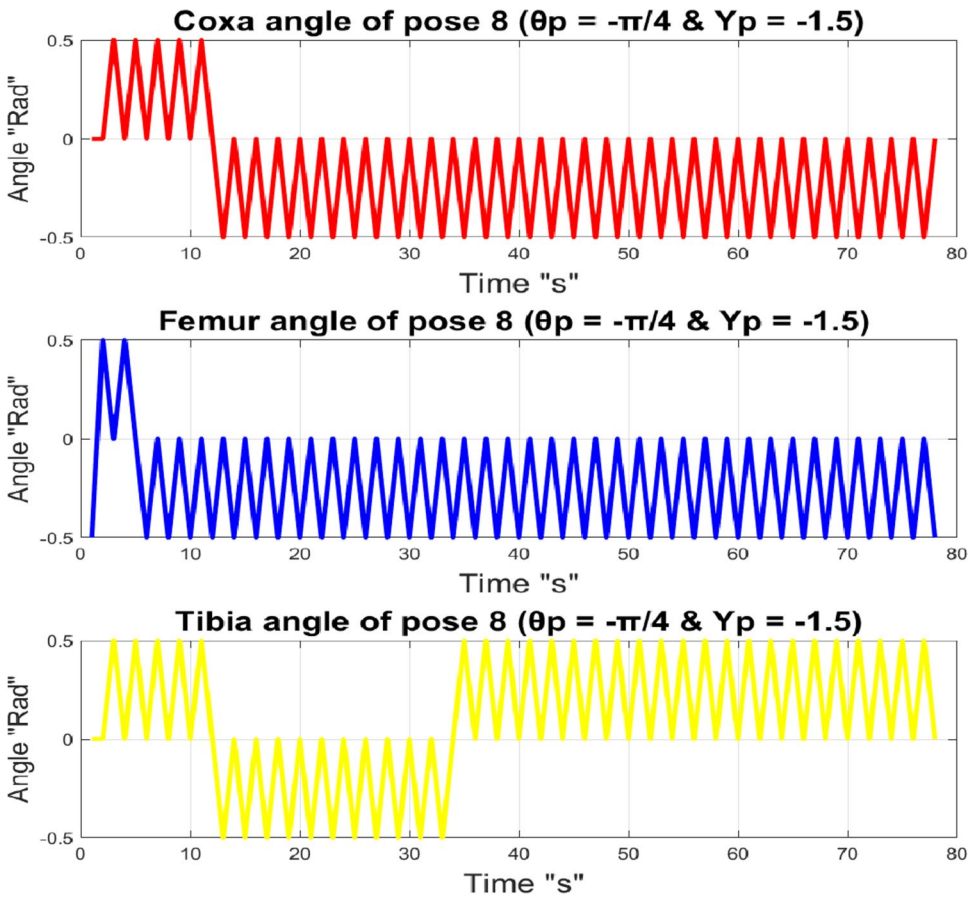

Fig. 29 Variation of the coxa, femur, and tibia joint angles of the hexapod robot's leg during the experiment starting from initial pose 8

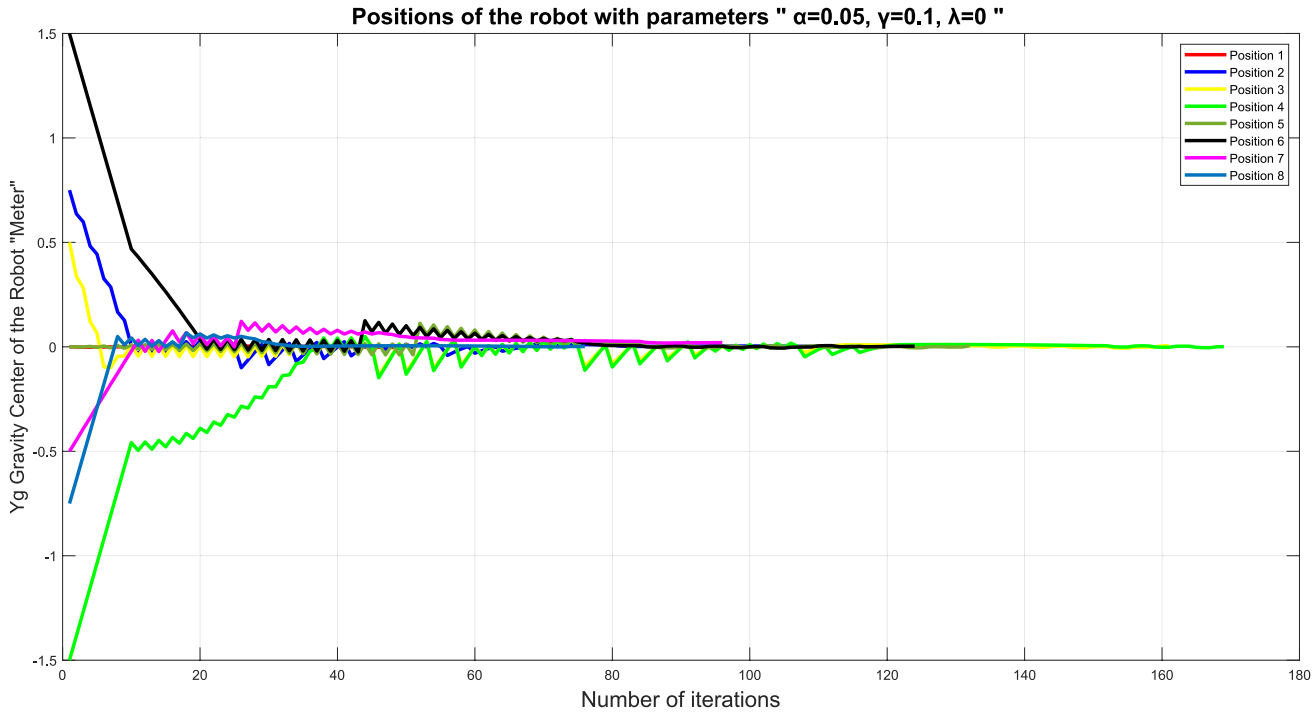

Fig. 30 Orientation changes in the gravity centre of the robot using the Q-Learning algorithm for Experiment 1

trials: (Ey), (Eθ), along with the number of steps taken to arrive at the target position. The lateral deviation errors are presented in Table 14 and visualized through the robot's centre of gravity trajectories in Figs. 11, 13, 15, 17, and 19. Orientation errors are reported in Table 15 and illustrated in Figs. 12, 14, 16, 18, and 20. The number of iterations needed for convergence in each case is summarized in Table 16.

These experiments investigate the influence of varying SARSA parameters (α, γ, λ) on the robot's trajectory control performance across all eight initial poses.

5.2.1.1 Influence of Learning Parameters on lateral deviation error (Ey) Table 14 presents the lateral deviation error (Ey) for each pose across all SARSA experiments. Experiment 3 (α=0.05, γ=0.5, λ=0) exhibited the highest overall error, particularly at Pose 4 (−0.0484) and Pose 7 (−0.0517). This suggests that a lower discount factor combined with the absence of eligibility traces impairs the agent's ability to perform long-term planning, resulting in reduced spatial accuracy. In contrast, Experiment 2 (α=0.01, γ=0.9, λ=0.9) demonstrated consistently low and positive error values, notably at Poses 2 to 4. The higher values of γ and λ appear to enhance long-term reward optimization, contributing to better trajectory correction and stability. Additionally, Experiment 5 (α=0.1, γ=0.3, λ=0.1) achieved one of the lowest errors at Pose 4 (0.0002) and showed high accuracy at Pose 8 (0.087). The elevated learning rate may have facilitated faster adaptation, although the relatively low discount factor likely limited its effectiveness in handling more complex trajectory transitions.

5.2.1.2 Orientation Error (Eθ) As shown in Table 15, Experiments 1 and 3 both configured with λ=0 exhibited greater variation and generally poorer orientation control. For instance, Pose 4 in Experiment 1 recorded an error of −0.0367, while the same pose in Experiment 3 showed+0.0209. These findings suggest that the absence of eligibility traces hinders the stability of angular corrections. In contrast, Experiment 2, which employed high values for both γ and λ, demonstrated improved orientation accuracy, with most error values remaining close to zero. This further supports the advantage of incorporating long-term policy refinement through eligibility traces. Although Experiment 5 presented a few notable deviations (e.g., Pose 3: −0.0171 and Pose 4:+0.0371), the orientation errors remained within acceptable bounds. The high learning rate in this case may have contributed to rapid correction of angular deviations despite a relatively low discount factor.

5.2.1.3 Iteration count Table 16 summarizes the iteration counts required for the robot to reach each target pose during the experiments conducted using the SARSA algorithm. Experiment 1 consistently required the fewest iterations across all poses. While this may initially appear advanta-

Table 17 The value of "Ey" of each pose crossing all experiments using Q-Learning algorithm

	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8
Experiment 1	0.0053	0	0.0059	0.0011	0.0049	0.003	0.0201	0.0012
Experiment 2	0.0062	0.0057	0.0063	0.0002	0.0031	0.0179	0.001	−0.0071
Experiment 3	0	0.0015	0	0.0018	−0.0015	−0.0004	−0.0034	0.0049
Experiment 4	0.0130	0.0123	0.0091	0.013	0.0178	0.0165	0.0173	0.0203
Experiment 5	−0.0219	−0.0075	0.0272	−0.0244	0.005	−0.0254	0.03	0.0017

Table 18 The value of "Ey" of each pose crossing all experiments using Q-Learning algorithm

	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8
Experiment 1	0.0133	−0.0166	0.0088	0.0023	0.0277	0.0144	0.0109	−0.0331
Experiment 2	0.0062	0.0055	0.0063	0.0002	0.0031	0.0179	0.001	−0.0071
Experiment 3	−0.0245	0.0076	0.0036	0.0250	−0.0084	0.0323	−0.0054	−0.0335
Experiment 4	−0.0006	0.0131	−0.0321	−0.0046	−0.0319	−0.0306	−0.0303	−0.0051
Experiment 5	0.0219	0.0075	0.0272	0.0244	0.0050	−0.0254	0.0300	0.0017

Table 19 The number of iterations of each pose crossing all experiments using Q-Learning algorithm

	Pose 1	Pose 2	Pose 3	Pose 4	Pose 5	Pose 6	Pose 7	Pose 8
Experiment 1	51	115	161	169	132	124	96	76
Experiment 2	51	115	153	175	135	156	102	78
Experiment 3	51	85	111	136	140	131	98	82
Experiment 4	57	154	281	257	161	177	147	131
Experiment 5	51	170	135	169	155	168	101	80

geous, it coincided with higher spatial and angular errors, suggesting premature convergence to suboptimal policies. In contrast, Experiment 4 required the most iterations, with the highest counts observed at Pose 2 (238) and Pose 3 (226). However, this longer training duration yielded notably low error values, indicating that a well-tuned Q-table supported by eligibility traces can prioritize precision over convergence speed. Experiment 5 achieved a favourable trade-off, combining relatively low iteration counts with acceptable Ey and Eθ values. This performance highlights the potential of a higher learning rate (α) to accelerate convergence, even when paired with a low discount factor (γ).

5.2.1.4 General observations The inclusion of eligibility traces ($\lambda > 0$) consistently improved trajectory accuracy, particularly in terms of orientation. This outcome supports the conclusion that temporal credit assignment enhances learning performance in dynamic, multi-legged robotic systems. Higher discount factors (γ) were also associated with better long-term performance, as demonstrated in Experiments 2 and 4. This is expected, as a higher γ encourages the agent to consider future rewards more effectively, leading to smoother and more goal-oriented trajectories. Conversely, a low learning rate ($\alpha = 0.01$) generally resulted in slower adaptation, as observed in Experiment 1. However, when combined with high γ and λ values as in Experiment 2, this limitation was mitigated, yielding stable and accurate performance.

Overall, Experiment 2 ($\alpha = 0.01$, $\gamma = 0.9$, $\lambda = 0.9$) achieved the best balance between spatial and angular accuracy, while maintaining a reasonable number of iterations. This makes it the most effective configuration for precise trajectory tracking in the hexapod robot. In contrast, Experiment 3, which excluded eligibility traces and used a lower γ, underperformed significantly, particularly in spatial accuracy. These findings highlight the critical role of both temporal credit assignment and future-oriented decision-making in enhancing the SARSA algorithm's effectiveness for robotic gait control.

While theoretical foundations of the SARSA algorithm highlight its on-policy nature, ensuring that value estimates reflect the actual behavior of the agent during training (Sutton and Barto 2018), our results extend this understanding by applying SARSA in the context of hexapod robot trajectory control. To the best of our knowledge, there is no prior published work that specifically investigates the use of SARSA for controlling the trajectory of a hexapod robot. This makes our study one of the first attempts to explore its effectiveness in such a domain. The findings show that SARSA can maintain stable orientation and distance errors within the accepted tolerance range ([−0.1, 0.1]), thereby validating its theoretical advantage of producing smoother and safer behavior, even though its convergence speed is slower than Q-Learning.

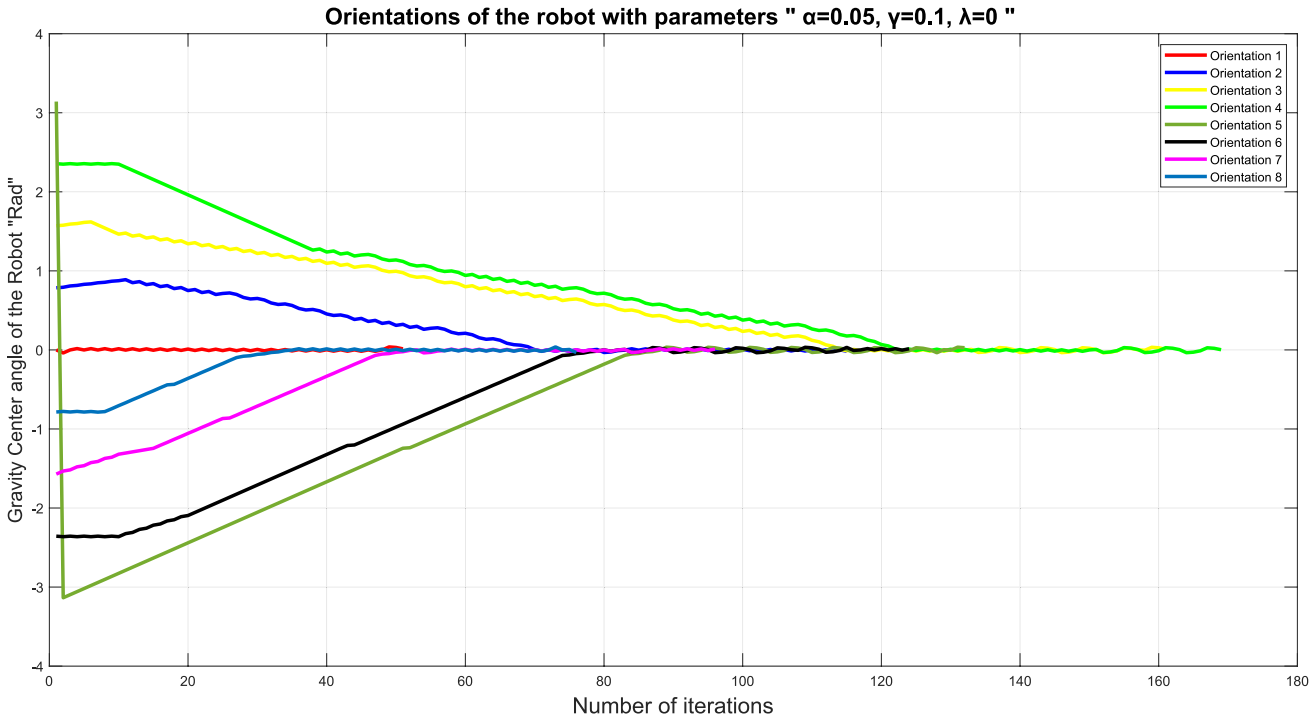

Fig. 31 Orientation changes in the gravity centre of the robot using the Q-Learning algorithm for Experiment 1

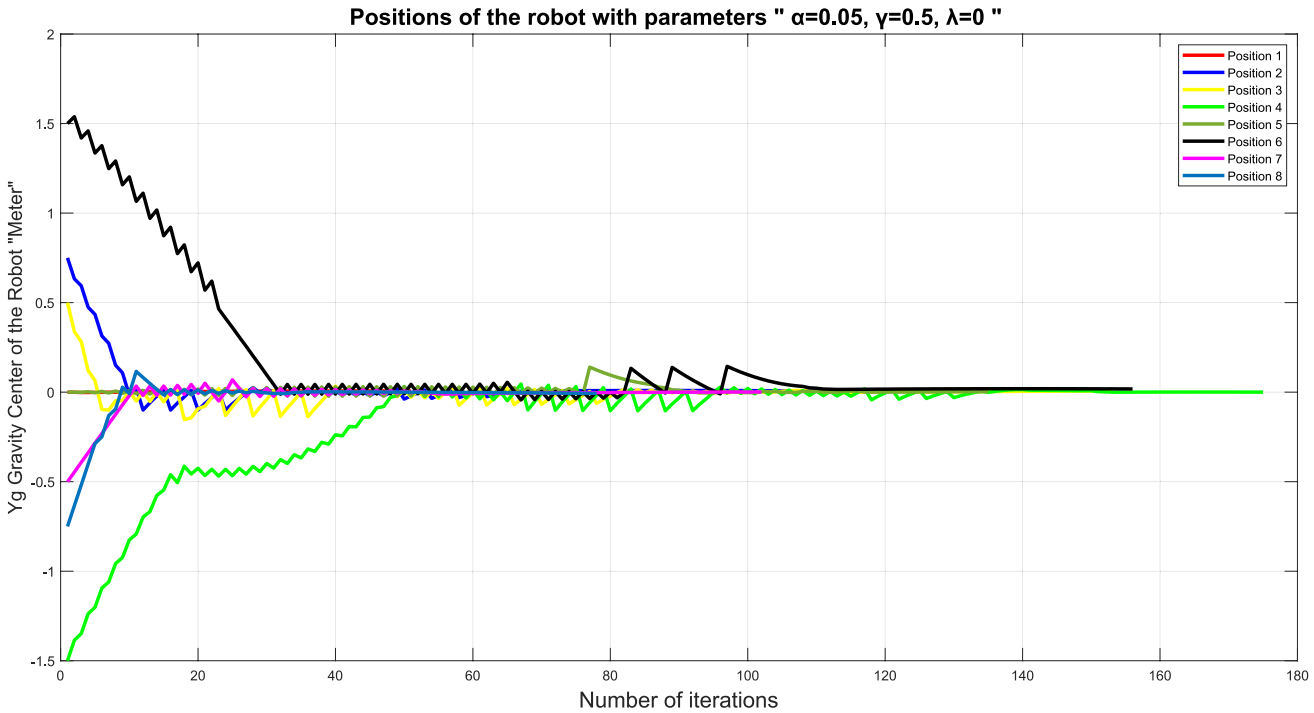

Fig. 32 Changes in the gravity centre of the robot trajectory using the Q-Learning algorithm for Experiment 2

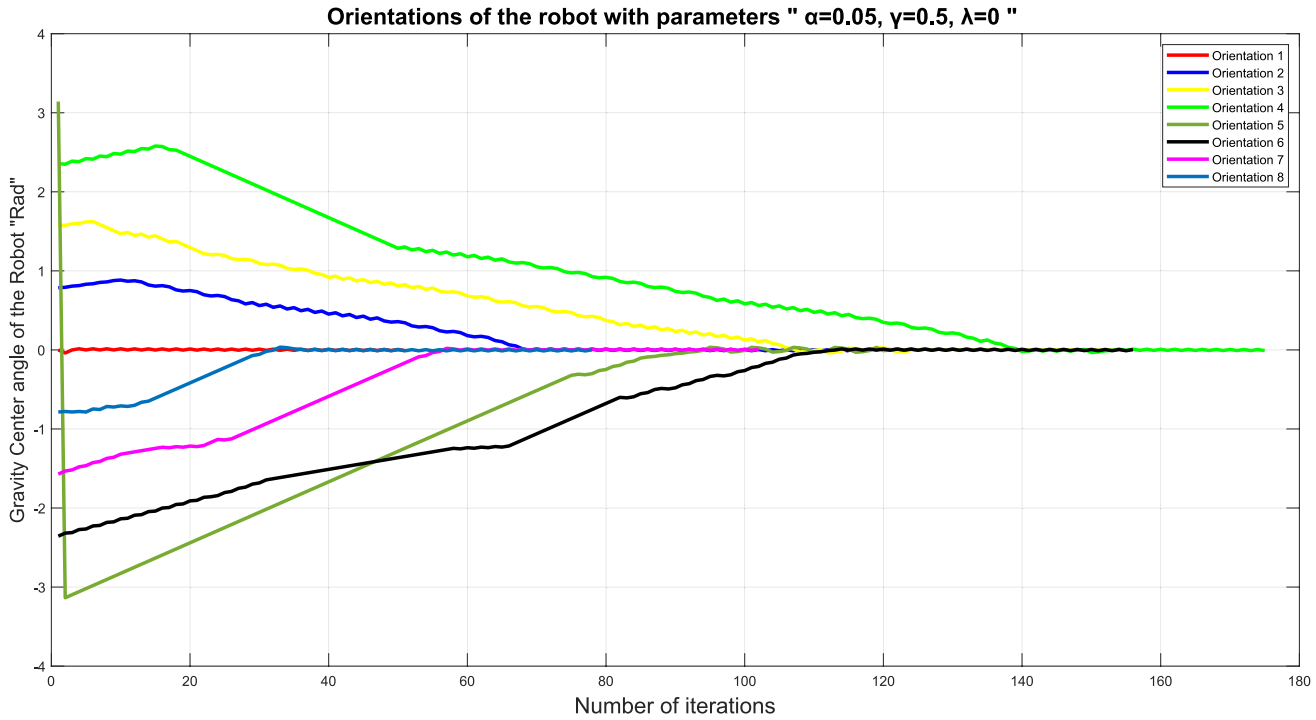

Fig. 33 Orientation changes in the gravity centre of the robot using the Q-Learning algorithm for Experiment 2

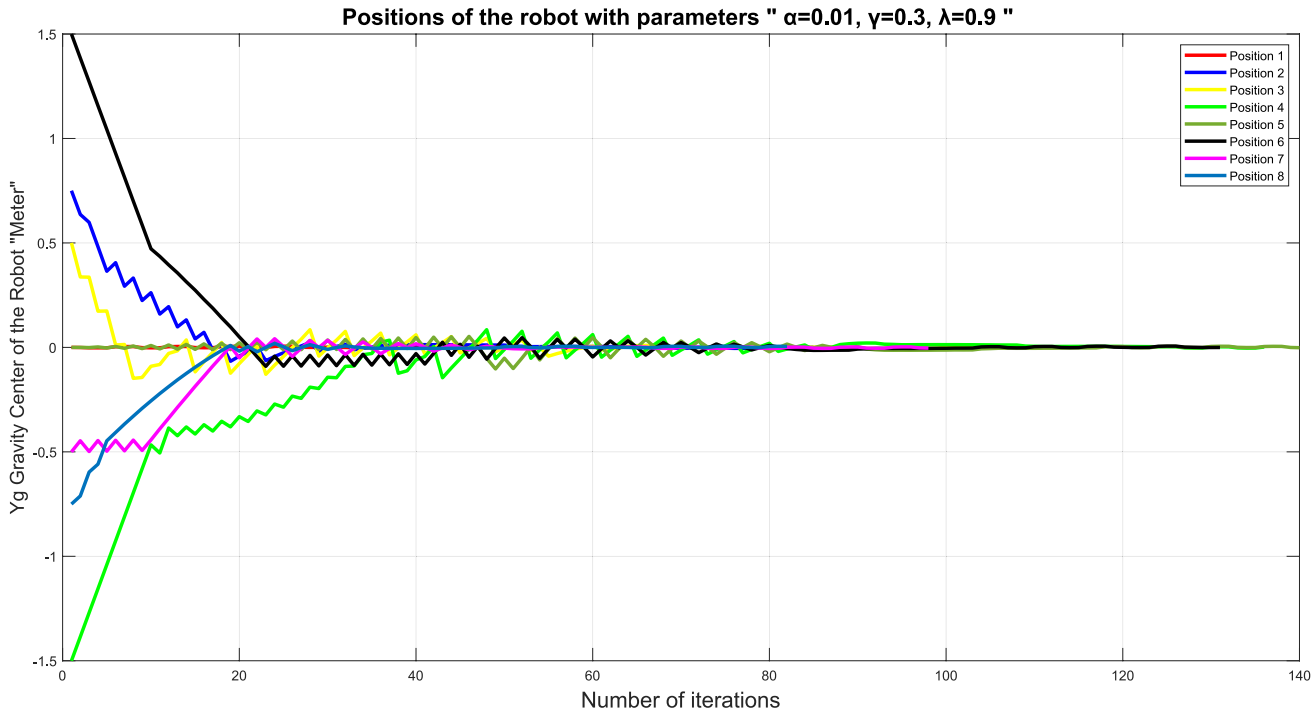

Fig. 34 Changes in the gravity centre of the robot trajectory using the Q-Learning algorithm for Experiment 3

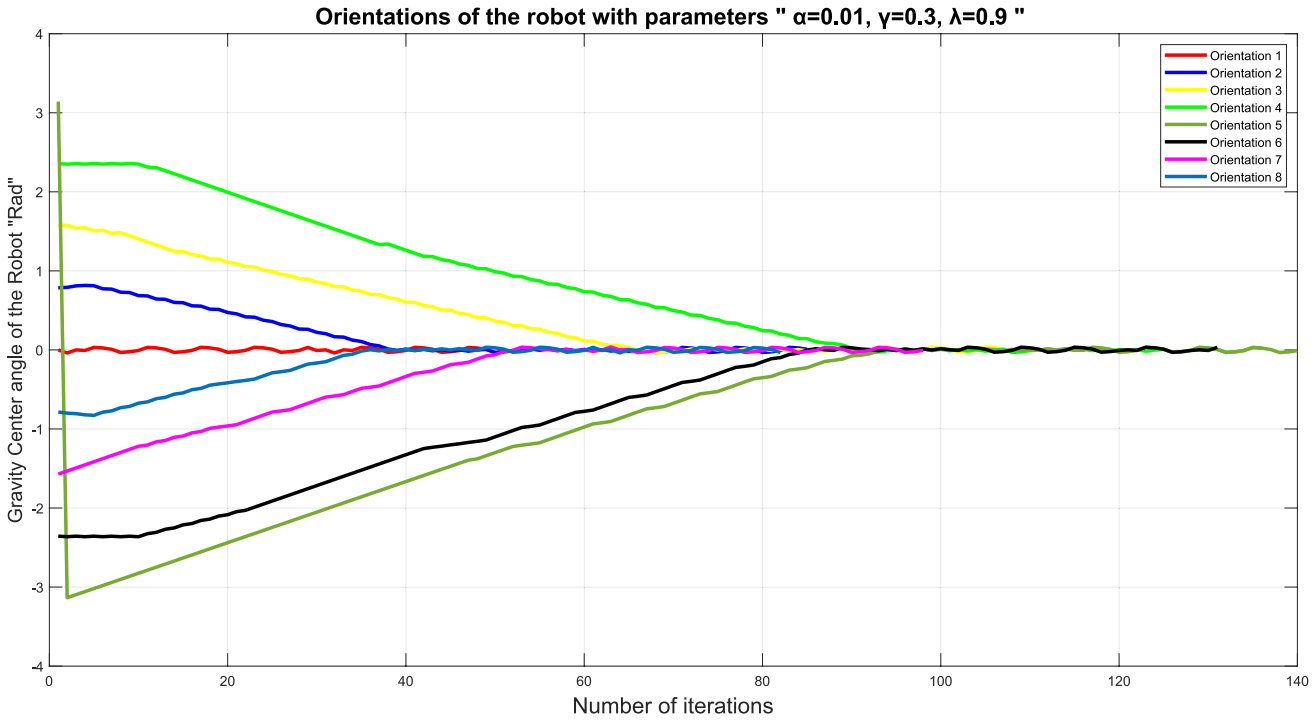

Fig. 35 Orientation changes in the gravity centre of the robot using the Q-Learning algorithm for Experiment 3

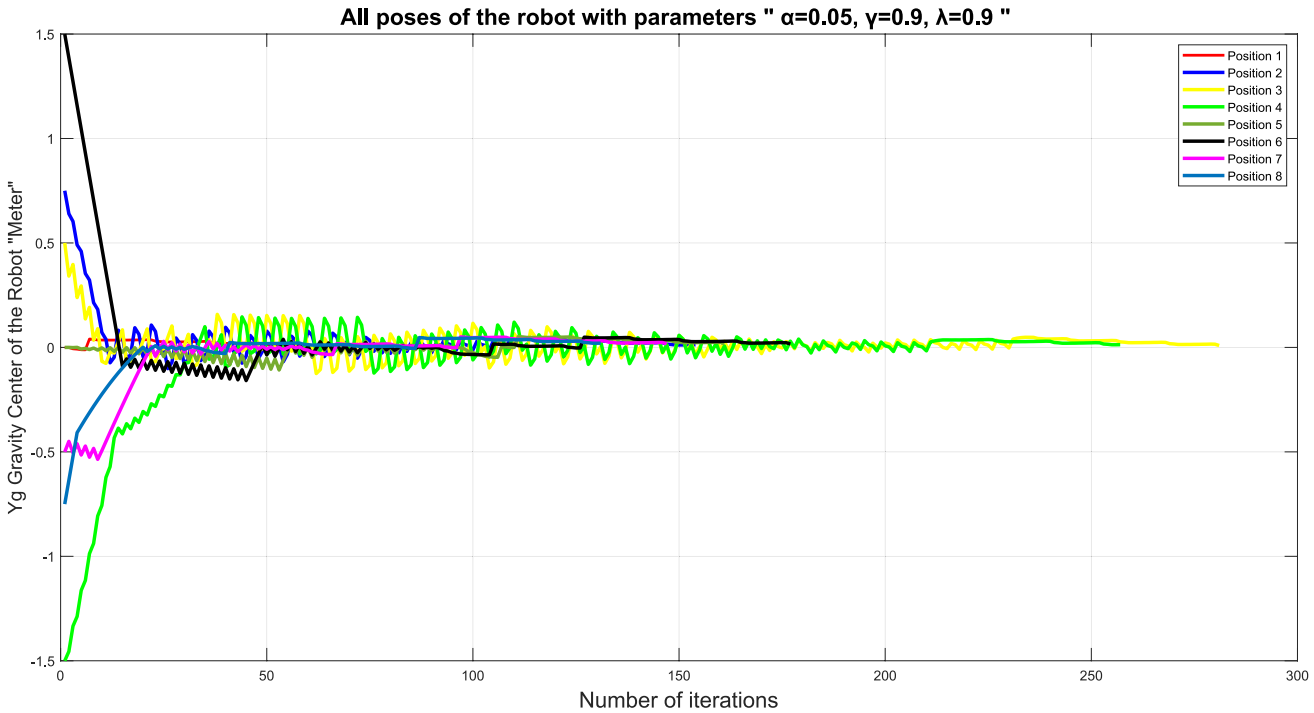

Fig. 36 Changes in the gravity centre of the robot trajectory using the Q-Learning algorithm for Experiment 4

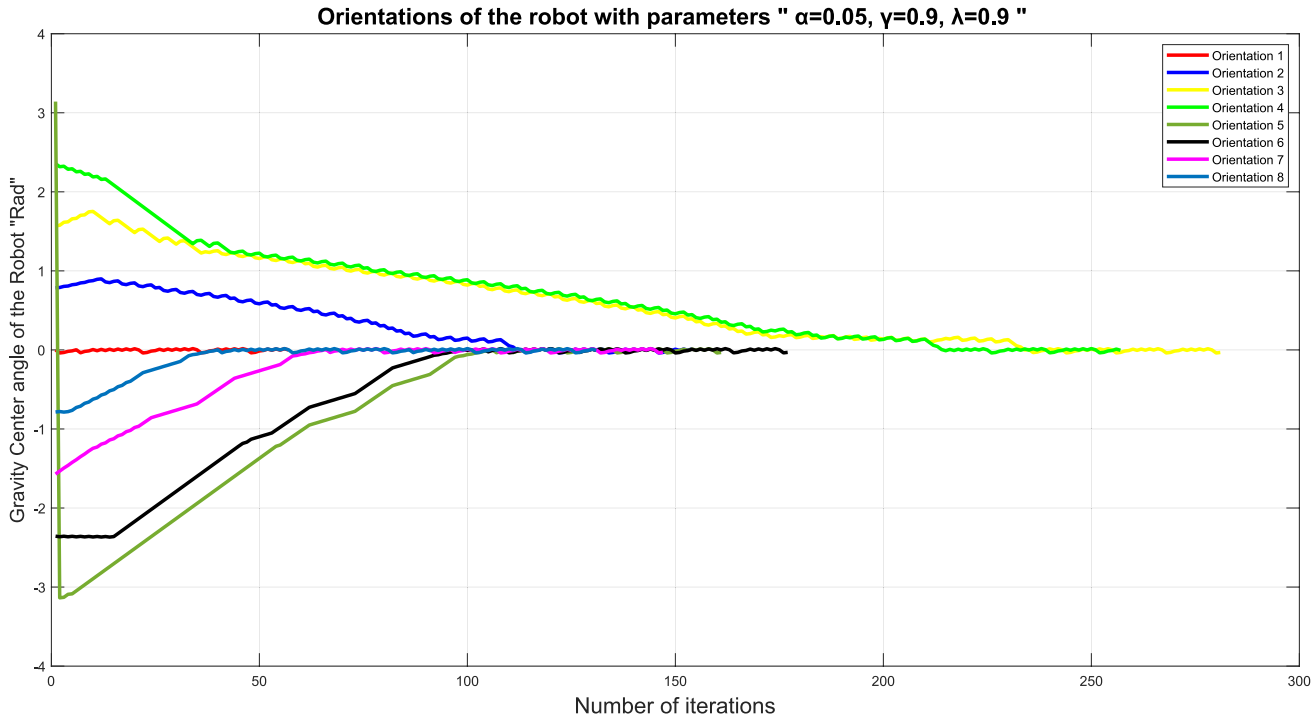

Fig. 37 Orientation changes in the gravity centre of the robot using the Q-Learning algorithm for Experiment 4

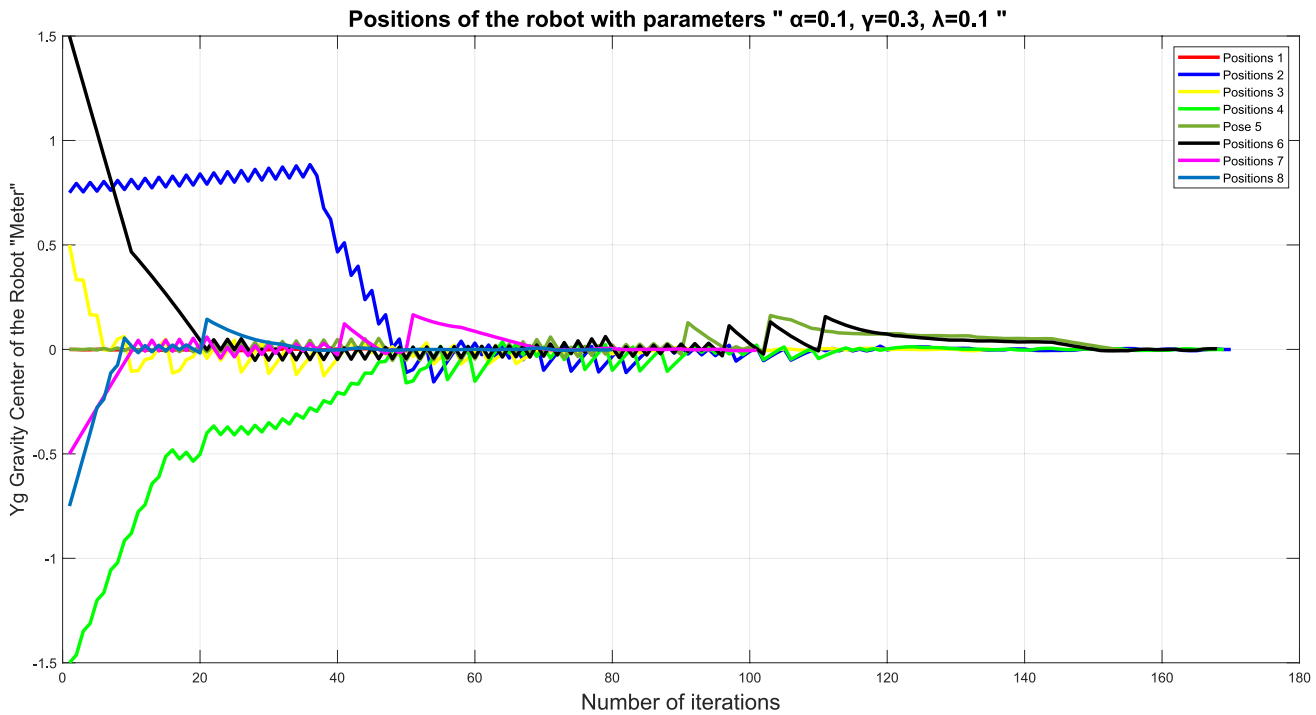

Fig. 38 Changes in the gravity centre of the robot trajectory using the Q-Learning algorithm for Experiment 5

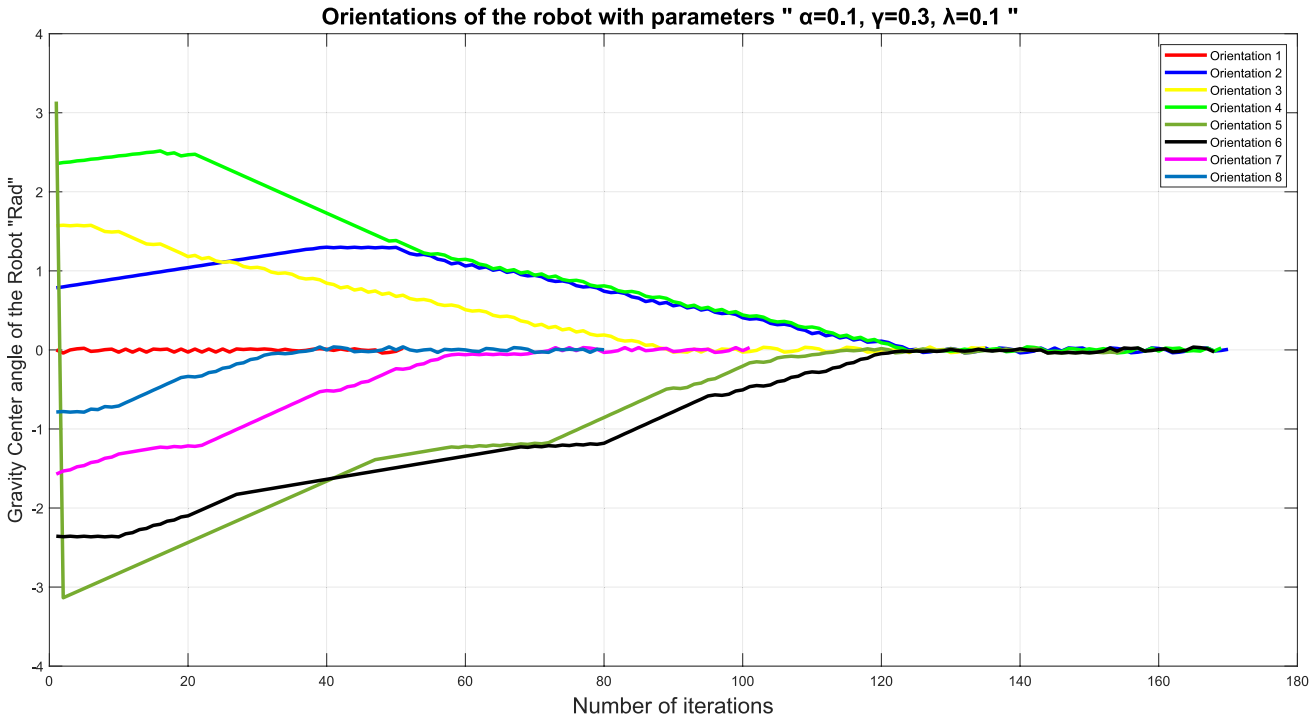

Fig. 39 Orientation changes in the gravity centre of the robot using the Q-Learning algorithm for Experiment 4

5.2.2 Discussion of Q-Learning results

This section analyses the performance of the Q-Learning algorithm in controlling the trajectory of a hexapod robot. The evaluation is based on three key metrics: lateral deviation error (Ey), as presented in Table 17; orientation error (Eθ), shown in Table 18; and the number of iterations required to reach each pose, also detailed in Table 19. These metrics were measured across five experimental trials, as outlined in Table 13. Figures 30, 32, 34, 36, and 38 illustrate the trajectories of the robot's center of gravity for Experiments 1 through 5, respectively, while Figs. 31, 33, 35, 37, and 39 depict the corresponding orientation curves. Together, these visualizations provide insight into the spatial and angular accuracy of the robot's movement under different Q-Learning configurations.

5.2.2.1 Lateral Deviation Error (Ey) As shown in Table 17, Experiment 3 achieved zero or near-zero lateral deviation (Ey) across several poses (e.g., Poses 1, 2, and 3), indicating high spatial accuracy. This configuration used a high discount factor ($\gamma = 0.9$) and a full eligibility trace ($\lambda = 0.9$), which facilitated long-term policy optimization. Experiment 5 displayed slightly higher variability, including some negative deviations (e.g., Pose 1: −0.0219), but still maintained acceptable accuracy. The higher learning rate ($\alpha = 0.1$) likely contributed to faster convergence, albeit with minor overshooting. Experiments 1 and 2 also performed well, showing minimal Ey values across all poses. These results suggest that Q-Learning effectively manages local transitions and spatial alignment, even with lower or no eligibility trace values, in contrast to SARSA.

5.2.2.2 Orientation error (Eθ) Table 18 presents the orientation error (Eθ) across all experiments. Experiment 3 again exhibited superior orientation control, with consistently low or negative Eθ values (e.g., Pose 7: −0.0054; Pose 8: −0.0335). This reinforces the positive impact of combining high γ and λ values for maintaining directional stability. In contrast, Experiment 4, which had the lowest γ (0.3), experienced significantly poorer orientation control (e.g., Pose 5: −0.0319; Pose 6: −0.0306). This suggests that a low γ leads to short-sighted policy behaviour. Experiment 1 ($\gamma = 0.5$, $\lambda = 0$) demonstrated moderate performance but was clearly

outperformed by configurations with eligibility traces, confirming λ's role in stabilizing angular adjustments.

5.2.2.3 Iteration count Table 19 also outlines the steps counts required for the robot to reach each target pose, reflecting the convergence speed and computational demand of each Q-Learning setup.

- **Pose 1 Stability**: Similar to SARSA, Pose 1 consistently required a fixed or near-fixed number of iterations (~51), indicating a well-learned and easily replicable initial condition.
- **Iteration Peaks**: **Experiment 4** showed substantial spikes at Pose 3 (281 iterations) and Pose 4 (257 iterations), suggesting difficulty in state transitions. These results may reflect insufficient exploration or a need for more refined policy updates in complex navigation zones.
- **General Trend**: Iteration counts in earlier experiments were generally lower and more consistent than those observed in SARSA, pointing to faster convergence in less complex scenarios. However, the increased variability in later experiments suggests that Q-Learning's generalization ability may diminish in highly dynamic or ambiguous environments.

Summary, The Q-Learning algorithm demonstrated strong overall performance in trajectory control, with key advantages including:

- Minimal lateral and orientation errors across most configurations.
- Lower or comparable iteration counts relative to SARSA, reflecting reduced computational overhead.
- Stable performance in early trajectory segments and robust adaptability in more complex transitions

The superior performance of Q-Learning in achieving reduced trajectory and orientation errors aligns with theoretical foundations asserting that off-policy temporal-difference methods tend to converge more quickly toward optimal value functions, given sufficient exploration and learning rate tuning. This theoretical trait is empirically affirmed in hexapod robot locomotion research, such as in the emergency-response hexapod gait adaptation study using heuristic Q-Learning, which demonstrated effective real-time locomotion response under joint failure conditions (Yang et al. 2019). Additionally, distributed Q-Learning frameworks for hexapod trajectory control further validate that Q-Learning variants can manage decentralized control tasks with promising convergence and performance (Zennir et al. 2003b; Zennir and Couturier 2005a, b). Together, these findings substantiate your results by showing that faster convergence and reduced error bounds via Q-Learning are consistent with both theory and prior practical implementations.

5.2.3 Discussion of joint angles results

Pose 1 (Fig. 22**)**: This pose represents straightforward, stable locomotion along the X-axis with no rotational or lateral deviation (θp = 0, Yp = 0). The joint angle trajectories exhibit highly regular, periodic oscillations across all three joints coxa, femur, and tibia over 50 s, indicating a steady-state gait. The coxa joint shows symmetric lateral motion between −0.4 and 0 rad, reflecting consistent hip articulation for forward propulsion. The femur and tibia joints both oscillate at ~ 10 Hz within ± 0.5 rad, demonstrating synchronized high-frequency stepping. The brief transient at t = 0 quickly damps out, confirming rapid stabilization of the control system. The uniform amplitude and frequency across all joints highlight excellent gait regularity and mechanical efficiency, typical of optimized tripod gaits in hexapods on flat terrain. This serves as a baseline for normal walking, where minimal energy is lost to disturbances or corrections.

Pose 2 (Fig. 23**):** In this manoeuvre, the robot performs a moderate rotational turn (45°) while simultaneously correcting a significant lateral offset (1.5 m). The joint dynamics reveal complex coordination and transitional behaviour over a 120-s interval. The coxa joint exhibits irregular oscillations with phase shifts and reduced activity before stabilizing at approximately t = 60 s, suggesting an initial gait reconfiguration to accommodate both turning and lateral movement. The femur maintains ~ 10 Hz oscillations but undergoes a temporary reduction in amplitude and frequency between 60–70 s, indicating a gait transition phase caused by weight shifting and stance-leg reassignment during rotation. The tibia shows intermittent activity, with two active phases separated by a low-motion interval, reflecting asymmetric foot engagement likely the result of certain legs lifting or adjusting during lateral correction. This pose illustrates how the robot dynamically redistributes leg function to manage multi-axis motion, prioritizing stability during combined manoeuvres.

Pose 3 (Fig. 24**):** This 90° turn, combined with a smaller lateral correction, results in segmented and adaptive joint behaviour over 160 s. The coxa joint alternates between high-frequency oscillations and brief pauses, indicating intermittent stance-swing cycles, to legs being used as pivot points during the turn. The femur maintains a stable ~ 10 Hz rhythm until a sharp amplitude drop at t = 120 s, suggesting a leg retraction or gait re-synchronization as the turn

concludes. The tibia shows highly variable activity with periods of zero motion, reflecting selective foot placement and ground contact modulation some legs may be lifted or dragged depending on their role in the turn. This pose demonstrates increased complexity in leg coordination, where certain limbs act as anchors while others generate turning torque, showcasing the robot's ability to modulate gait per leg for rotational control.

Pose 4 (Fig. 25**)**: The 135° turn requires greater articulation, producing highly segmented and dynamic joint behaviour over a 200-s interval. The coxa joint exhibits multiple oscillatory phases with pauses and a sharp spike at t = 140 s, indicating active hip repositioning to complete the turn. The femur maintains its ~ 10 Hz stepping rhythm but shows a sudden amplitude drop around t = 140 s, coinciding with the coxa spike suggesting a coordinated leg lift or reconfiguration to avoid interference or adjust stance geometry. The tibia displays intermittent, frequency-modulated activity with extended periods of inactivity, implying that certain legs disengage from propulsion to serve as pivot points. This pose highlights advanced gait plasticity, in which the robot temporarily sacrifices continuous stepping in favour of precise rotational control, emphasizing task-dependent leg specialization.

Pose 5 (Fig. 26**)**: The full 180° reversal is a challenging maneuver requiring complete reorientation without lateral movement. Over 140 s, the coxa shows small-amplitude oscillations (± 0.3 rad) with a sharp spike at t = 80 s, likely marking a key repositioning event, such as a leg swing or body pivot. The femur and tibia maintain high-frequency (~ 10 Hz) stepping but show irregularities between t = 100–130 s, suggesting instability or gait disruption during the final phase of the turn. These disturbances may arise from increased torque demands or leg interference when reversing direction. Despite this, the overall periodicity indicates successful completion of the turn through coordinated leg motion, though with reduced gait smoothness compared to smaller turns. This pose demonstrates the limits of continuous gait during extreme manoeuvres, requiring temporary compromises in rhythm for reorientation.

Pose 6 (Fig. 27**)**: This clockwise 135° turn mirrors Pose 4 in magnitude but in the opposite direction, combined with a lateral shift from the negative Y-axis. The coxa joint shows periodic oscillations with sharp spikes at t = 60, 80, and 100 s, indicating repeated hip adjustments to manage rotational torque and foot placement. The femur maintains consistent high-frequency oscillations (~ 10 Hz) with minimal amplitude variation, reflecting continuous propulsive effort and balance maintenance. In contrast, the tibia displays segmented activity with frequency modulation and pauses, suggesting selective foot engagement some legs may be used for pivoting while others step. The symmetry in femur behaviour across turns (Poses 4 and 6) suggests robust mid-leg control, while tibia modulation highlights adaptive distal control. This pose confirms the robot's directional symmetry in control strategy, capable of handling large turns in either direction with similar coordination.

Pose 7 (Fig. 28**):** This 90° clockwise turn is executed over a 100-s interval with a lateral correction from the negative Y-axis. The coxa exhibits regular motion with a sharp spike at t = 20 s, indicating an early hip adjustment to initiate the turn. The femur maintains uninterrupted high-frequency oscillations (~ 10 Hz), providing consistent leg drive and stability throughout. The tibia displays segmented activity with a notable pause between t = 30–60 s, suggesting that one or more legs temporarily disengage from stepping to act as pivot points during lateral translation. This pause is shorter than in larger turns, reflecting less disruption during moderate manoeuvres. Overall, Pose 7 illustrates efficient coordination between continuous propulsion (femur) and adaptive distal control (tibia), enabling smooth execution of moderate directional changes.

Pose 8 (Fig. 29**):** This manoeuvre involves a 45° clockwise turn and a large lateral shift ($Yp = -1.5 \rightarrow 0$) over 80 s. The coxa shows initial high-frequency motion stabilizing after t = 10 s, indicating rapid adjustment. The femur maintains ~ 10 Hz oscillations with a brief amplitude drop around t = 10–15 s, likely due to load redistribution. The tibia exhibits a pause between t = 20–35 s, reflecting selective foot disengagement during lateral repositioning. Despite the small rotation, the large lateral displacement dominates the control strategy, requiring significant leg reconfiguration and temporary stepping interruptions**.**

The comparative analysis of joint angle trajectories across eight locomotion poses reveals a strong dependence of gait modulation on task complexity. During straight walking (Pose 1), the coxa, femur, and tibia exhibit stable, periodic oscillations, reflecting a highly regular and efficient gait. In contrast, rotational and lateral manoeuvres (Poses 2–8) introduce significant gait adaptations, including intermittent activity, phase shifts, and transient amplitude reductions. Larger turns (e.g., 135° and 180° in Poses 4–6) are marked by pronounced coxa spikes and tibia pauses, indicating active leg repositioning and selective foot engagement. The femur maintains consistent ~ 10 Hz oscillations across all conditions, highlighting its role as the primary rhythmic driver, while the coxa and tibia show greater adaptability to directional changes. Clockwise manoeuvres (Poses 6–8) exhibit smoother transitions and faster stabilization, suggesting directional asymmetry in control or mechanics. Notably, large lateral displacements—even with small rotations (e.g., Pose 8) induce significant gait segmentation, demonstrating that lateral translation imposes greater locomotors demands than rotation alone. Overall, the hexapod

Table 20 SARSA and Q-learning key comparison

Aspect	SARSA	Q-Learning	Remarks
Update function	Equation 5	Equation 3	In Q-Learning using $max_{a\prime}.Q^i\ (s\prime, a\prime)$ instead $Q^i\ (s\prime, a\prime)$ which is in SARSA algorithm
Best Ey (Distance Error)	Experiment 2: (α = 0.01, γ = 0.9, λ = 0.9)	Experiment 3: (α = 0.05, γ = 0.9, λ = 0.9)	Both favor long-term reward and use eligibility traces
Best Eθ (Orientation Error)	Experiment 2: Smallest orientation drift across poses	Experiment 3: Most consistent angular control across poses	SARSA showed slightly better orientation stability in dynamic poses
Convergence Speed	Experiment 1: Fastest (lowest iterations overall)	Experiment 1: Fastest for SARSA, but higher than Q-Learning	Q-Learning converges faster overall, but may sacrifice stability
Stability Across Poses	Moderate—some spikes in error (e.g., Pose 7 in Exp 1)	High—especially in Exp 3 with minimal error fluctuations	SARSA shows smoother performance across all poses
Effect of Eligibility Traces (λ)	High λ improves performance significantly	High λ also improves accuracy and stability	Eligibility traces are critical for both algorithms' success
Parameter Sensitivity	Sensitive to γ and λ	Sensitive to α and γ	Parameter tuning is essential for both; SARSA is more robust with λ

Table 21 Limitations of Q-Learning vs. SARSA in Hexapod Trajectory Control

Aspect	SARSA (On-Policy)	Q-Learning (Off-Policy)
Convergence speed	May converge prematurely to suboptimal policies (fast but inaccurate in some cases)	Generally faster in simple scenarios, but can show iteration spikes in complex state transitions
Spatial accuracy (Ey)	More dependent on eligibility traces; poor accuracy without λ	Maintains low Ey even with little or no λ, but overshooting can occur with high α
Orientation stability (Eθ)	Strongly reliant on λ for angular stability; unstable when λ = 0	Achieves good orientation accuracy with γ,λ high; unstable with low γ (short-sighted)
Discount factor sensitivity (γ)	Low γ leads to poor long-term planning and high error	Low γ causes short-sighted policies and unstable orientation control
Eligibility traces (λ)	Essential for stable performance; without λ, errors increase significantly	Improves stability with λ, but less dependent than SARSA for lateral accuracy
Learning rate sensitivity (α)	Low α = very slow adaptation; high α helps but may need strong γ, λ to stabilize	High α accelerates learning but risks overshooting and variability
Generalization ability	More stable but slower to adapt; struggles less with dynamic environments but may sacrifice speed	Strong in early/simple segments, weaker in highly dynamic or ambiguous transitions
Trade-off	Prioritizes safety/stability but at cost of slower learning	Prioritizes speed/aggressiveness but may lose precision in complex cases

employs a task-dependent gait strategy, balancing rhythmic stability with adaptive joint control to achieve robust and versatile locomotion.

5.2.4 Comparison between the results of Q-Learning and SARSA algorithms

This section presents a comparative evaluation of the SARSA and Q-Learning algorithms for trajectory control of a hexapod robot, based on multiple experimental trials, along with an assessment of their limitations. The key differences in performance and behaviour are summarized in Table 20.

The results show that Q-Learning generally outperforms SARSA in terms of trajectory stability and orientation accuracy, particularly when high eligibility trace values (λ) and long-term reward emphasis (γ = 0.9) are applied. Q-Learning converges more rapidly, especially in early learning stages, making it more suitable for scenarios where training time is constrained. However, under certain conditions, it exhibits slightly higher orientation drift. The distinction between the two update strategies—Q-Learning's off-policy learning versus SARSA's on-policy updates—had a clear impact on the exploration–exploitation trade-off. Q-Learning's updates toward the greedy action encouraged more aggressive exploitation of high-value actions, accelerating convergence but occasionally amplifying errors in stochastic transitions. By contrast, SARSA's reliance on the actual next action under its ε-greedy policy implicitly incorporated exploration, producing more conservative behavior. This slowed convergence in some cases but improved stability in environments requiring frequent exploratory moves, such as obstacle-rich or dynamically varying trajectories.

Both algorithms achieved trajectory control within the predefined acceptable error bounds of [−0.1, 0.1] for orientation error (Eθ) and lateral distance error (Ey). From a feasibility perspective, this confirms that both methods can generate valid trajectories. Nevertheless, Q-Learning consistently produced smaller deviations within this interval. While SARSA occasionally yielded Eθ and Ey values approaching ± 0.08, Q-Learning typically maintained errors closer to zero, often between ± 0.02 and ± 0.04, representing up to a 60–70% reduction in residual error. Furthermore, Q-Learning retained 80–85% of trials within the tighter interval [−0.05, 0.05], compared to 55–65% for SARSA.

It also exhibited smoother convergence across trials, with fewer oscillations in trajectory corrections, whereas SARSA showed greater fluctuations despite remaining within acceptable limits. These findings indicate that although both methods satisfy error constraints, Q-Learning offers greater robustness by operating closer to the center of the tolerance band, thereby enhancing trajectory precision and reducing the risk of boundary violations.

To contextualize their applicability, the limitations of both algorithms in hexapod trajectory control were analysed (Table 21). While both enabled effective path following, they differed notably in convergence behaviour, stability, parameter sensitivity, and adaptability to dynamic environments.

In summary, the comparative analysis highlights a fundamental trade-off. As an on-policy method, SARSA generates value estimates that closely reflect the robot's actual behavior, yielding more stable and conservative policies. This robustness, however, comes at the cost of slower convergence and greater dependence on eligibility traces and discount factors. In contrast, Q-Learning, being off-policy, achieves faster convergence and generally superior spatial and orientation accuracy, particularly under high γ and λ. Yet, its aggressive pursuit of optimal returns increases sensitivity to parameter tuning and introduces potential instability in complex or dynamic navigation scenarios. Overall, SARSA is more suitable for tasks requiring safe, steady adaptation, while Q-Learning is better suited to applications prioritizing rapid convergence and high precision.

For real-time robotic control, Q-Learning may be advantageous due to its faster convergence and reduced oscillatory behaviour, whereas SARSA may be preferable in simulation-driven studies that emphasize stable policy development.

6 Conclusion

This study presented a comprehensive evaluation of the SARSA and Q-Learning algorithms for controlling the trajectory of a hexapod robot across multiple experimental configurations. Both methods were tested under varying parameter settings for learning rate (α), discount factor (γ), and eligibility trace (λ), with the objective of optimizing trajectory tracking and minimizing errors in position and orientation.

The results show that Q-Learning outperforms SARSA in trajectory stability, orientation precision, and convergence speed, particularly with high λ and γ values, making it more suitable for real-time control despite slightly higher orientation drift. SARSA, while slower and more sensitive to eligibility traces, provided stable performance and faster policy adaptation in simulation-intensive scenarios. Both algorithms benefited considerably from eligibility traces, which enhanced temporal credit assignment and improved trajectory accuracy over time. Overall, Q-Learning appears preferable for real-time applications where responsiveness and stability are essential, whereas SARSA may offer advantages in simulation environments that prioritize convergence speed.

Despite these advantages, both Q-Learning and SARSA face inherent limitations. As tabular methods, their scalability is constrained in high-dimensional or continuous state–action spaces, often requiring discretization that reduces precision. Their performance is also sensitive to hyper parameter selection, which can lead to slow convergence or suboptimal policies if not carefully tuned. Furthermore, in complex or partially observable environments, both algorithms may struggle to capture long-term dependencies, making them less suitable without additional extensions.

Future work will focus on integrating Simultaneous Localization and Mapping (SLAM) for terrain mapping and developing a Deep Reinforcement Learning–based control framework for hexapod navigation in obstacle-rich and unstructured environments. These enhancements are expected to improve the robot's ability to operate in unknown areas, accurately map uneven terrains, and enhance its locomotion capabilities for traversing highly irregular surfaces.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s41315-025-00496-6.

Author contribution A.Benyoucef and Y. Zennir wrote the main manuscript text and prepared figures and tables. A.Belatreche, M.F.Silva and M. Benghanem reviewed the manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

Aissaoui, A., Mahfoudi, C., Djeffal, S.: 'A General Kinematic-Based Walking Algorithm of a Hexapod Robot on Irregular Terrain' (2024).

AlMahamid, F., Grolinger, K.: 'Reinforcement learning algorithms: An overview and classification', in *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–7. (2021). https://doi.org/10.1109/CCECE53047.2021.9569056.

Amhraoui, E., Masrour, T.: Expected Lenient Q-learning: a fast variant of the Lenient Q-learning algorithm for cooperative stochastic Markov games. Int. J. Mach. Learn. Cybern. **15**(7), 2781–2797 (2024). https://doi.org/10.1007/s13042-023-02063-6

Benyoucef, A., Amrane, A., Zennir, Y., Belatreche, A.: Autonomous obstacle avoidance for a hexapod robot using proximity sensors. Int. J. Autom. Saf **02**(01), 1–6 (2024)

Benyoucef, A., Zennir, Y.: 'Enhancing Hexapod Robot Locomotion Control Through PID Controller Optimization Using Genetic Algorithm', in *2023 IEEE 11th International Conference on Systems and Control (ICSC)*, Sousse, Tunisia: IEEE, pp. 556–561 (2023). https://doi.org/10.1109/ICSC58660.2023.10449774.

Bjelonic,M., Homberger, T., Kottege, N., Borges, P., Chli, M., Beckerle, P.: 'Autonomous navigation of hexapod robots with vision-based controller adaptation', in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore: IEEE, pp. 5561–5568 (2017). https://doi.org/10.1109/ICRA.2017.7989655.

Boaventura, T., Medrano-Cerda, G. A., Semini, C., Buchli, J., Caldwell, D. G. (2013). Stability and performance of the compliance controller of the quadruped robot HyQ. In *2013* IEEE/RSJ international conference on intelligent robots and systems (pp. 1458-1464). IEEE.

Brooks, R.A.: 'A robot that walks; emergent behaviors from a carefully evolved network', in *Proceedings, 1989 International Conference on Robotics and Automation*, Scottsdale, AZ, USA: IEEE Comput. Soc. Press, pp. 692–4+2 (1989). https://doi.org/10.1109/ROBOT.1989.100065.

Cai, Z., Gao, Y., Wei, W., Gao, T., Xie, Z.: Model design and gait planning of hexapod climbing robot. J. Phys. Conf. Ser. **1754**(1), 012157 (2021). https://doi.org/10.1088/1742-6596/1754/1/012157

Celaya, E., Porta, J.M.: 'Force-based control of a six-legged robot on abrupt terrain using the subsumption architecture', pp. 1–10, (2000).

Chen, C., et al.: Attitude trajectory optimization to ensure balance hexapod locomotion. Sensors **20**(21), 1–31 (2020). https://doi.org/10.3390/s20216295

Chen, L., Wang, Q., Deng, C., Xie, B., Tuo, X., Jiang, G.: Improved double deep Q-network algorithm applied to multi-dimensional environment path planning of hexapod robots. Sensors **24**(7), 2061 (2024). https://doi.org/10.3390/s24072061

Coelho, J., Ribeiro, F., Dias, B., Lopes, G., Flores, P.: Trends in the control of hexapod robots: a survey. Robotics **10**(3), 1–22 (2021). https://doi.org/10.3390/robotics10030100

Ding, Z., Huang, Y., Yuan, H., Dong, H.: 'Introduction to Reinforcement Learning.' In: Dong, H., Ding, Z., Zhang, S. (eds.) *Deep Reinforcement Learning*, pp. 47–123. Springer Singapore, Singapore (2020). https://doi.org/10.1007/978-981-15-4095-0_2

Djeffal, S., Morakchi, M.R., Ghoul, A., Kargin, T.C.: DDPG-based reinforcement learning for controlling a spatial three-section continuum robot. Frankl. Open **6**, 100077 (2024). https://doi.org/10.1016/j.fraope.2024.100077

Dobrevski M., Skocaj, D.: 'Adaptive dynamic window approach for local navigation', in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, pp. 6930–6936 (2020). https://doi.org/10.1109/IROS45743.2020.9340927.

Dulac-Arnold, G., et al.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Mach. Learn. **110**(9), 2419–2468 (2021). https://doi.org/10.1007/s10994-021-05961-4

Fu H.,*et al.*: 'Deep Reinforcement Learning for Multi-contact Motion Planning of Hexapod Robots', in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, Montreal, Canada: International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 2381–2388. https://doi.org/10.24963/ijcai.2021/328.

Fuchs, 'Intersegmental coordination of cockroach locomotion: adaptive control of centrally coupled pattern generator circuits', *Front. Neural Circuits*, 2010, https://doi.org/10.3389/fncir.2010.00125.

Huang, Z., Liu, Q., Zhu, F.: Hierarchical reinforcement learning with adaptive scheduling for robot control. Eng. Appl. Artif. Intell. **126**, 107130 (2023). https://doi.org/10.1016/j.engappai.2023.107130

Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. Neural Netw **21**(4), 642–653 (2008). https://doi.org/10.1016/j.neunet.2008.03.014

Ivo, H., Patrik, K.: 'Reinforcement Learning In Control Systems for Walking Hexapod Robots.', *ResearchGate*, pp. 1–13, (2005)

Ji, Q., Qian, Z., Ren, L., Ren, L.: Simulation analysis of impulsive ankle push-off on the walking speed of a planar biped robot. Front. Bioeng. Biotechnol. **8**, 1–11 (2021). https://doi.org/10.3389/fbioe.2020.621560

Kingsley, D., Quinn, R., Ritzmann, R.: 'A Cockroach Inspired Robot With Artificial Muscles', in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China: IEEE, Oct. pp. 1837–1842. (2006) https://doi.org/10.1109/IROS.2006.282229.

Kirchner, F.: 'Q-learning of complex behaviours on a six-legged walking machine', in *Proceedings Second EUROMICRO Workshop on Advanced Mobile Robots*, Brescia, Italy: IEEE Comput. Soc, pp. 51–58 (1997). https://doi.org/10.1109/EURBOT.1997.633565.

Lagaza, K.P., Pandey, A.: 'Review Article Mechanical Engineering', (2018).

Li, H., Qi, C., Gao, F., Chen, X., Zhao, Y., Chen, Z.: Mechanism design and workspace analysis of a hexapod robot. Mech. Mach. Theory **174**, 104917 (2022). https://doi.org/10.1016/j.mechmachtheory.2022.104917

Lin, J.-L., Hwang, K.-S., Jiang, W.-C., Chen, Y.-J.: Gait balance and acceleration of a biped robot based on Q-learning. IEEE Access **4**, 2439–2449 (2016). https://doi.org/10.1109/ACCESS.2016.2570255

López-Lozada, E., Rubio-Espino, E., Sossa-Azuela, J.H., Ponce-Ponce, V.H.: Reactive navigation under a fuzzy rules-based scheme and reinforcement learning for mobile robots. PeerJ Comput. Sci. **7**, 1–25 (2021). https://doi.org/10.7717/peerj-cs.556

Ma, J., Qiu, G., Guo, W., Li, P., Ma, G.: Design, Analysis and Experiments of Hexapod Robot with Six-Link Legs for High Dynamic Locomotion. Micromachines **13**(9), 1–20 (2022). https://doi.org/10.3390/mi13091404

Margolis, G.B., Yang, G., Paigwar, K., Chen, T., Agrawal, P.: 'Rapid Locomotion via Reinforcement Learning', *Robot. Sci. Syst.*, p. 12, (2022).

Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., Hutter, M.: Learning robust perceptive locomotion for quadrupedal robots in the wild. Sci. Robot. **7**(62), eabk2822 (2022). https://doi.org/10.1126/scirobotics.abk2822

Nishigai, K., Ito, K.: 'Control of multi-legged robot using reinforcement learning with body image and application to a real robot', in *2011 IEEE International Conference on Robotics and Biomimetics*, Karon Beach, Thailand: IEEE 2011, pp. 2511–2516. https://doi.org/10.1109/ROBIO.2011.6181682.

Peng, X.B., Berseth, G., Yin, K., Van De Panne, M.: Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning. ACM Trans. Graph. **36**(4), 1–13 (2017). https://doi.org/10.1145/3072959.3073602

Porta, J., Celaya, E.: 'Body and Leg Coordination for Omni-Directional Walking in Rough Terrain', *Inst. Robòtica Informàtica Ind. UPC-CSIC Barc. SPAIN*, pp. 1–8, (2000).

Qiu, Z., Wei, W., Liu, X.: Adaptive Gait Generation for Hexapod Robots Based on Reinforcement Learning and Hierarchical Framework. Actuators **12**(2), 1–15 (2023). https://doi.org/10.3390/act12020075

Qu, T., Li, D., Zakhor, A., Yu, W., Zhang, T.: 'Versatile Locomotion Skills for Hexapod Robots'.

Ribeiro, T., Goncalves, F., Garcia, I., Lopes, G., Ribeiro, A.F.: 'Q-Learning for Autonomous Mobile Robot Obstacle Avoidance', in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Porto, Portugal: IEEE, pp. 1–7. (2019) https://doi.org/10.1109/ICARSC.2019.8733621.

Rummery, G., Niranjan, M.: 'On-Line Q-Learning Using Connectionist Systems', *Camb. Univ. Eng. Dep. Trumpington*, p. 19, (1994).

Schilling, M., Melnik, A.: 'An Approach to Hierarchical Deep Reinforcement Learning for a Decentralized Walking Control Architecture', in *Biologically Inspired Cognitive Architectures 2018*, vol. 848, A. V. Samsonovich, Ed., in Advances in Intelligent Systems and Computing, vol. 848. Cham: Springer International Publishing, pp. 272–282. (2019) https://doi.org/10.1007/978-3-319-99316-4_36.

Siciliano, B., Khatib, O., Eds., *Springer Handbook of Robotics*. in Springer Handbooks. Cham: Springer International Publishing, (2016). https://doi.org/10.1007/978-3-319-32552-1.

Silva, M.F., Machado, J.A.T.: Kinematic and dynamic performance analysis of artificial legged systems. Robotica **26**(1), 19–39 (2008). https://doi.org/10.1017/S0263574707003554

Silva, M.F., Machado, J.T.: A literature review on the optimization of legged robots. J. Vib. Control **18**(12), 1753–1767 (2012). https://doi.org/10.1177/1077546311403180

Silva, M.F., Machado, J.A.T., Lopes, A.M.: Modelling and simulation of artificial locomotion systems. Robotica **23**(5), 595–606 (2005). https://doi.org/10.1017/S0263574704001195

Singh, D.S.K.: A literature survey of hexapod robots. Int. J. Eng. Res. Comput. Sci. Eng. **4**(6), 95–104 (2017)

Singh, O., Barodiya, V.K., Paridwal, A., Makhija, A.: Reinforcement learning in robotics: challenges and applications. Turk. J. Comput. Math. Educ. TURCOMAT **11**(2), 743–745 (2020). https://doi.org/10.61841/turcomat.v11i2.14417

Singh, S.P., Sutton, R.S.: 'Reinforcement learning with replacing eligibility traces', *1996 Kluwer Acad.*, pp. 123–158, 196AD.

Socha, V., Kutilek, P., Stefek, A., Socha, L., Schlenker, J., Hana, K.: Decision Making Process of Hexapods in a Model of Complex Terrains. Acta Polytech. Hung. **13**(4), 141–157 (2016). https://doi.org/10.12700/APH.13.4.2016.4.9

Sutton R. S., Barto A. G., *Reinforcement Learning: An Introduction*. 2014.

Sutton R. S., Barto A. G, *Reinforcement learning: an introduction*, Second edition. in Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.

Tan J *et al.*, 'Sim-to-Real: Learning Agile Locomotion For Quadruped Robots' 2018, *arXiv*: arXiv:1804.10332. https://doi.org/10.48550/arXiv.1804.10332.

Thor, M., Manoonpong, P.: Versatile modular neural locomotion control with fast learning. Nat. Mach. Intell. **4**(2), 169–179 (2022). https://doi.org/10.1038/s42256-022-00444-0

Travers, M., Ansari, A., Choset, H.: 'A dynamical systems approach to obstacle navigation for a series-elastic hexapod robot', in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Las Vegas, NV, USA: IEEE, pp. 5152–5157 (2016). https://doi.org/10.1109/CDC.2016.7799057.

Wang, X., Fu, H., Deng, G., Liu, C., Tang, K., Chen, C.: Hierarchical free gait motion planning for hexapod robots using deep reinforcement learning. IEEE Trans. Ind. Inform **19**(11), 10901–10912 (2023a). https://doi.org/10.1109/TII.2023.3240758

Wang, Z., Gao, F., Zhao, Y., Yin, Y., Wang, L.: Improved A* algorithm and model predictive control- based path planning and tracking framework for hexapod robots. Ind. Robot. Int. J. Robot. Res. Appl **50**(1), 135–144 (2023b). https://doi.org/10.1108/IR-01-2022-0028

Wang T, Taghvaei A, Mehta PG, 'Q-learning for POMDP: An application to learning locomotion gaits', in *2019 IEEE 58th Conference on Decision and Control (CDC)*, Nice, France: IEEE,2019, pp. 2758–2763. https://doi.org/10.1109/CDC40024.2019.9030143.

Wenxia, X., Yu, B., Cheng, L., Li, Y., Cao, X.: Multi-fuzzy Sarsa learning-based sit-to-stand motion control for walking-support assistive robot. Int. J. Adv. Robot. Syst **18**(5), 17298814211050190 (2021). https://doi.org/10.1177/17298814211050190

Yang, Q., Gao, Y., Li, S.: 'Terrain-adaptive Central Pattern Generators with Reinforcement Learning for Hexapod Locomotion' (2023), *arXiv*: arXiv:2310.07744. https://doi.org/10.48550/arXiv.2310.07744.

Yang, M.-C., Samani, H., Zhu, K.: 'Emergency-Response Locomotion of Hexapod Robot with Heuristic Reinforcement Learning Using Q-Learning', in *Interactive Collaborative Robotics*, vol. 11659, A. Ronzhin, G. Rigoll, and R. Meshcheryakov, Eds., in Lecture Notes in Computer Science, vol. 11659. , Cham: Springer International Publishing, pp. 320–329 (2019). https://doi.org/10.1007/978-3-030-26118-4_31.

Zennir, Y.: Apprentissage par renforcement et systèmes distribués : application à l'apprentissage de la marche d'un robot hexapode. Phd these, INSA de Lyon (2004)

Zennir, Y., Couturier, P.: Approche distribuée de l'apprentissage. Application au contrôle de la trajectoire d'un robot hexapode. Journal Européen des Systèmes Automatisés **39**(8), 965–993 (2005a). https://doi.org/10.3166/jesa.39.965-993

Zennir, Y.. Couturier, P.: 'Multiactor approach and hexapod robot learning', in *2005 International Symposium on Computational Intelligence in Robotics and Automation*, Espoo, Finland: IEEE, pp. 665–671 (2005b). https://doi.org/10.1109/CIRA.2005.1554353.

Zennir, Y., Coutourier, P., Bétemps, M.: 'Distributed Reinforcement Learning of a Six-Legged Robot to Walk', in *The Fourth International Conference on Control and Automation 2003 ICCA Final Program and Book of Abstracts ICCA-03*, Montreal, Que., Canada: IEEE, pp. 896–900 (2003a). https://doi.org/10.1109/ICCA.2003.1595152.

Zennir, Y., Coutourier, P., Bétemps, M.: 'Apprentissage De La Marche D'un Robot Hexapode Selon une Approche Distribée du Q-Learning', *Conférence Int. En Sci. Electron. Technol. Inf. Télécommunications Bouhlel MS Solaiman B Kamoun Eds Sousse Tunis. 17–21 Mars 2003b*, pp. 1–8.

Zhou, X.: Optimal values selection of Q-learning parameters in stochastic mazes. J. Phys. Conf. Ser. **2386**(1), 012037 (2022). https://doi.org/10.1088/1742-6596/2386/1/012037

Ahmed Benyoucef is preparing his Ph.D degree in Automation from Skikda University. His research interests include mobile robots, adaptive control, and Reinforcement Learning algorithms.

Zennir Youcef Professor in electrical engineering 2021 at Skikda University. Phd in industrial automation 2004 at INSA of Lyon. DEA in industrial automation in 2000 at University of Savoie and Engineer in automation 2008 at Annaba University. Field of interest: robotics, autonomous systems, safety system, multi-agents systems, control system, artificial intelligent.

Ammar Belatreche is an Associate Professor of Computer Science at Northumbria University, UK. He received his PhD in Computer Science from Ulster University, UK, where he began his academic career as a Research Associate in the Intelligent Systems Research Centre before taking up a lectureship in the School of Computing and Intelligent Systems, Derry. He joined Northumbria University in May 2016. At Northumbria, Dr Belatreche serves as Postgraduate Taught (PGT) Director and Deputy Lead of the Data Science and Artificial Intelligence (DSAI) research group. He previously led the MSc Advanced Computer Science with Advanced Practice program.

Manuel F. Silva received his Licentiate, Master's, and Ph.D. degrees in Electrical and Computer Engineering from the Faculty of Engineering, University of Porto, Portugal, in 1997 and 2005, respectively. He is currently a Coordinating Professor at the School of Engineering of the Polytechnic of Porto, Department of Electrical Engineering, and a Principal Researcher at the Centre for Industrial Robotics and Intelligent Systems (CRIIS) of INESC TEC. In addition, he serves as a Trustee of the CLAWAR Association Ltd. and has been President of the Portuguese Robotics Society. He has been actively involved in several R&D projects.

Mohamed Benghanem is currently a Professor at the Faculty of Science, Physics Department, Islamic University of Madinah, Saudi Arabia. He previously served as a Professor at Taibah University (2004–2017) and has been a Regular and Senior Associate at the International Centre for Theoretical Physics (ICTP), Trieste, Italy, since 2004. He earned his BE (1987), MSc (1991), and Ph.D. (2000) in Electrical Engineering from the Polytechnic School of Algiers and USTHB University of Algiers with the collaboration of the Solar Energy Institute, University Polytechnic of Madrid (UPM), Spain. His research areas include modelling, simulation, control & smart systems and the use of artificial intelligent such as ANN for solar energy applications.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com