# Application of a Factorial Designed Experiment to Optimize Selection of Reinforcement Learning Observations for a Hexapod Trajectory Following Task

by

Alec Freeman

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
July 2024

# Table of Contents

# List of Tables

# List of Figures

# Abstract

This thesis describes the development of a hexapod simulator built in the MATLAB Simscape environment, with the goal of testing the potential for a designed experiment to be use in the selection of observations for a reinforcement learning controlled hexapod design. The hexapod is controlled using a novel combination of a central pattern generator consisting of six coupled Hopf oscillators, and mapping functions with parameters updated via a reinforcement learning agent. The reinforcement learning agent is trained to control the hexapod using the Deep Deterministic Policy Gradient (DDPG) algorithm on a trajectory following task. Through implementation of a designed experiment testing different combinations of observations, a model is formulated to estimate the observations required to maximize the hexapod training reward. The model is validated in the simulator and the capabilities of the hexapod are further demonstrated on more complex path following tasks.

# List of Abbreviations Used

| | |
|---|---|
| CPG | Central Pattern Generator |
| DDPG | Deep Deterministic Policy Gradient |
| IMU | Inertial Measurement Unit |
| PC | Personal Computer |
| PD | Proportional Derivative |
| PI | Proportional Integral |
| ReLU | Rectified Linear Unit |
| RL | Reinforcement Learning |
| TG | Trajectory Generator |

# Chapter 1: Introduction

Mobile robotics play an important role in the substitution of humans to complete dangerous tasks in the fields of surveillance, demining, inspection, rescue operations, and exploratory missions [1]. Such robots must be capable of overcoming difficult terrain in challenging environments without the need for human intervention. Walking hexapod robots are an ideal candidate for these roles as they offer significant advantages over wheeled robots in terms of both maneuverability and capabilities on rugged and unforeseen terrain.

Walking robots are often controlled via a central pattern generator – a system which generates rhythmic signals without requiring any rhythmic or changing inputs. Central patten generators are able to produce smooth oscillating signals to drive robot motion using a limited number of input variables. One of the most common types of central pattern generator applied to walking mobile robots is the Hopf oscillator, which consists of two coupled differential equations producing a stable cyclic pattern. The Hopf oscillator based central pattern generator used in this work is described in Chapter 3.

The current direction of the literature is for mobile robots to become increasingly independent of human operators through the use of machine learning. Recent works have demonstrated the application of reinforcement learning to train and control hexapod robots in complex environments for both path-planning navigation and complex locomotion tasks. Reinforcement learning allows a mobile robot to modify its behaviour and walking gait automatically in real time in response to changes in terrain, external stimulus, and other inputs. Reinforcement learning agents are trained through a repetitive process where the hexapod must repeat a similar task many times in order to iteratively modify its behaviour to obtain a maximum reward for the given task. One such reinforcement learning algorithm that will be used in this work is the Deep Deterministic Policy Gradient (DDPG). Combining reinforcement learning with a central pattern generator improves learning speed as the central pattern generator provides a base gait signal allowing the robot to be much more successful at the start of training.

There are numerous factors and parameters that influence the success of a reinforcement learning agent, with one of the most important being the selection of the observations, meaning

the measurements of the robot state and environment provided to the reinforcement learning agent. In many machine learning problems the maximum amount of data available is provided to the learning algorithm, however in this application the observations needed to successfully learn a particular locomotion task will affect the physical hardware required and the design of the robot itself. There may be limitations in hardware cost, physical size constraints or power requirements that necessitate the selection of a limited number of observations for a hexapod robot. The optimal combination of observations for a given reinforcement learning task must be determined through testing; however, this process can lead to issues with constraints caused by the length of agent training time. Ibarz *et al.* [2] discuss the importance of sample efficiency as many widely-used reinforcement learning algorithms require millions of interactions over the course of training. Training can take a considerable amount of time even in a simulation environment. The reinforcement learning agent used in this work was trained in simulation using 15 second episodes which were repeated 1000 times – which took roughly 8 hours of actual training time. It is clear that, for the complex task of hexapod locomotion, a systematic method for determining the optimal observations is required in order to gain maximal insight into their effects in the most time efficient manner.

In the field of machine learning there exists several methods of feature selection whose purpose is to detect the relevant features of a set of training data and discard those which are irrelevant. Feature selection techniques can be grouped into three main categories: filter methods, wrapper methods, and embedded methods [3]. Filter methods of feature selection are used for supervised machine learning where a model is trained on a large existing dataset, and thus cannot be used for a reinforcement learning problem where no data exists before the training begins. Wrapper methods can be used for reinforcement learning but are computationally expensive. Embedded methods such as the technique presented by Whiteson [4] incorporate feature selection into the learning algorithm itself, updating the number of observations automatically to optimize performance. In the context of applying reinforcement learning to physical robot hardware, both wrapper and embedded feature selection methods have the limitation that, although they result in a selection of observations to maximize performance, they do not provide a model of how each observation affects the robot performance. The selection of observations can impact not only the hexapod performance, but additional constraints such as hardware cost, form factor, and both power and computational requirements. It would be beneficial to use a

method of observation (or feature) selection that can provide a model of how different observations affect hexapod performance to better evaluate these potential engineering trade-offs. The use of a designed experiment as a method of feature selection to select observations for a hexapod reinforcement learning problem will be explored in this thesis.

Originally developed for use in the fields of agriculture and industrial manufacturing, design of experiments is a statistical methodology used to produce experiments able to provide insight into the effects of various factors on a process result in the most efficient way possible [5]. If the reinforcement learning training routine is considered to be the process, the observations treated as process factors, and the final reward taken as measurable process result, then a designed experiment could potentially be used to optimize the combination of observations for maximum final reward. With the overall goal of maximizing the performance of a mobile robot, this thesis will explore the potential application of a fractional factorial designed experiment in the selection of observations for a hexapod locomotion task.

With these considerations in mind, the objectives for this thesis are as follows:

1. Develop a hexapod robot simulation platform which is built upon a central pattern generator-based control approach.
2. Apply Deep Deterministic Policy Gradient (DDPG) reinforcement learning to the central pattern generator of the simulated hexapod to enable the hexapod to learn a robust behaviour that allows the robot to follow a desired path.
3. Investigate the effect on hexapod performance of using different combinations of observations (sensor measurements) to train the reinforcement learning agent.
4. Determine the ability of a fractional factorial designed experiment to predict the best possible combination of observations to use for the hexapod path-following locomotion task.
5. Demonstrate in simulation that the complex behaviour of path following can be achieved through training where the hexapod is rewarded for traveling along a straight trajectory.

The remainder of this thesis is structured as follows. Chapter 2 presents a literature review of the field of hexapod robot control with a specific focus on central pattern generators and reinforcement learning. An extensive survey of relevant works is presented to explore the

different combinations of observations used to train reinforcement learning agents for walking robot tasks in the current literature. Chapter 3 details the development of the hexapod simulation environment, the Hopf oscillator-based central pattern generator used in this work, as well as the setup of the reinforcement learning agent and its integration into the hexapod control architecture. The setup of the designed experiment used in this work is presented in Chapter 4. Presentation and discussion of the results from the designed experiment along with corresponding analysis to produce the final mathematical model are detailed in Chapter 5. Predictions generated using the model are also validated in this chapter. Finally, Chapter 6 includes recommendations for future work and potential improvements, as well as conclusions about the main contributions of this work.

# Chapter 2: Literature Review

Mobile robots will play an increasingly important role in the replacement of humans in remote, hazardous, and extreme environments. Hexapod robots offer several advantages over other mobile robotics platforms including excellent maneuverability, versatility over complex terrain, better stability, redundancy to limb faults or failures, and adaptability to specific tasks or environments [6]. Research in the control of hexapod robots has progressed from traditional kinematics- and dynamics-based controllers, to biologically-inspired controllers using central pattern generators, optimization of gaits using genetic algorithms, and finally to the current cutting edge of control which incorporates machine learning – specifically reinforcement learning [1].

## 2.1. Central Pattern Generators

An important aspect of control for walking mobile robots is the use of central pattern generators (CPG) to drive the rhythmic gait patterns. A CPG is a system which generates a stable oscillating cycle without requiring any changing inputs. Central pattern generators have been observed in the field of neurobiology as a driving force for rhythmic behaviours in both vertebrae and invertebrates [7]. Central pattern generators are well suited to walking robot control as they produce smooth control signals, generate stable rhythmic patterns by exhibiting a stable limit-cycle behaviour, and the output signals can be collectively modified by adjusting a few key parameters [8]. Central pattern generators typically consist of an oscillator created using a set of differential equations which generates a stable limit cycle, with the most prevalent example from the literature being the Hopf-type oscillator. Recently, central pattern generators have proven an effective control method when applied to hexapod and quadruped robots performing complex locomotion tasks.

The works of both Ma *et al.* [9] and Wang *et al.* [10] apply coupled Hopf oscillators as central pattern generators to a quadruped robot and produce trot-type gaits in a simulation environment. Zhong *et al.* [11] utilizes a central pattern generator to control a novel 24 degree of freedom hexapod, with successful application to real-world hardware. Work by Wang *et al.* [12]

uses Hopf oscillators to generate control signals for the joints of a hexapod and demonstrates the stability of the resulting motion while walking on flat ground and up a slope.

Central pattern generators have been shown to produce smooth transitions between different gait types. Chen *et al.* [13] employ a Hopf oscillator as central pattern generator in the control of a hexapod robot and demonstrate the ability to smoothly transition between different gaits including a wave gait, tetrapod gait, and tripod gait. Smooth gait transitions for a hexapod were also demonstrated by Campos *et al.* [14]. The proposed control method was proven in both simulation and experimentally on a hexapod robot with the switch between gaits achieved by changing the phase lag in the Hopf oscillator. Work by Bal [15] also utilizes Hopf oscillators as central pattern generators, using six coupled oscillators to produce smooth gait transitions for a hexapod robot while exploring how the coupling relationships between the six oscillators affects the hexapod's performance.

The performance of a centrally-controlled hexapod or quadruped has been further improved in the literature through the addition of reactionary mechanisms to the robot control. Tran *et al.* [16] utilize a recurrent neural network as a central pattern generator which was applied to a quadruped robot with added reflexive pitch adaptation modules. The quadruped is able to react to changes in pitch of the robot body caused by either stepping on uneven ground or on inclines by feeding reflex signals calculated based on sensor measurements back to the central pattern generator to modify its output. Chung *et al.* [17] also demonstrated the ability for a hexapod controlled using a central pattern generator to use body tilt measurements to modify its walking gait while stepping over obstacles. A dual oscillator is employed to generate the joint signals for the robot, while a Proportional Integral (PI) controller is used to modify the robot's gait in response to changes in measured body tilt. Sartoretti *et al.* [18] incorporate inertial feedback into central pattern generator control for stable locomotion and climbing in unstructured terrain. Their results are experimentally validated on a hexapod robot in challenging natural terrain. Liu *et al.* [19] utilize coupled Hopf oscillators to control a quadruped robot and demonstrate the ability to transition between walking and trotting gait as well as using sensory feedback of body attitude to modify the gaits for walking on a slope. Yu *et al.* [20] increased the capabilities of a hexapod controlled with a central pattern generator by adding reactionary mechanisms to each leg independently. Force feedback from the tip of each leg allows the

hexapod to trigger a reactionary movement if the leg does not contact the ground in the expected location. The addition of these reactions to feedback about the environment allows the hexapod to traverse terrain with raised obstacles and lowered ditches [20].

The use of a central pattern generator is a proven foundation of control for a hexapod robot, and for that reason a Hopf oscillator was selected as the driving force of the central pattern generator in this work with full details found in Chapter 3.

## 2.2. Genetic Algorithms for Hexapod Optimization

While central pattern generators are an improvement on simple parameterized gaits, they have traditionally been optimized using manual hand tuning for each specific scenario or environment. Genetic algorithms were introduced to robotic control to automate the optimization process and have been applied in the literature to walking mobile robots. A genetic algorithm, inspired by the process of natural selection, is an iterative process where a set of gait parameters can be optimized to maximize a given goal. Trivun *et al.* [21] utilised a genetic algorithm to optimise a parameterized gait for a hexapod robot, so the hexapod is able to reach an optimal gait without manual tuning. The genetic algorithm was also tested as a method to deal with faults in the hexapod robot, and the demonstrated case was adaptation to a broken leg. Kon and Sahin [22] implemented a genetic algorithm to generate gaits for a hexapod robot where the genetic algorithm optimizes parameters of sinusoidal functions which drive the motion of each joint. The genetic algorithm was shown to be able to cope with motor faults (locked joints) much more effectively than maintaining a standard tripod gait.

Genetic algorithms have been successfully applied in combination with a central pattern generator. The work of Wang *et al.* [23] combines a central pattern generator and a genetic algorithm with application to a hexapod robot. A central pattern generator based on coupled Hopf oscillators generates output signals which are converted into joint-angle signals through a set of mapping functions. A genetic algorithm is used to optimize the parameters of the mapping functions with the goal of producing a fast gait while limiting the vertical oscillation. Borrett and Beckerleg [24] demonstrated a genetic algorithm for evolving two types of controllers: a novel evolvable hardware controller based on a virtual field programmable gate array, and an artificial

neural network. The genetic algorithm was shown to be able to optimize the hexapod gait in both simulation and real-world testing.

The main drawback of optimization using a genetic algorithm is that, once the gait parameters have been optimized, they remain fixed during deployment to the hexapod. The hexapod is still limited in its reactivity to external disturbances. As before, any reactions must be programmed into the underlying gait ahead of deployment, limiting the performance of the hexapod when new unseen disturbances are applied.

## 2.3. Reinforcement Learning in Hexapod Control

As noted by Coehlo *et al.* [1], the recent trend in controlling hexapod robots shows the emergence and increasing use of reinforcement learning in the literature over the last half-dozen years. Reinforcement learning has significant advantages over the other aforementioned control methods in terms of reactions to external disturbances and changes to the environment. A central pattern generator can successfully produce smooth walking gaits for a hexapod, select reflexive responses can be added to modify the gait based on sensor feedback, and further tuning and optimization of the gait can be performed using a genetic algorithm. However, these methods are limited in that, once the gait has been determined, it remains fixed throughout deployment and cannot react to disturbances or environmental changes outside of those set in the predetermined reflexes. Reinforcement learning agents, however, are able to react in real time to external stimuli and, through varied and extensive training, have been shown to be able to generalize their behaviours to previously-unseen circumstances. Heess *et al.* [25] demonstrated the emergence of complex locomotion behaviours for both a quadruped and a humanoid if provided sufficient sensory data during training in a complex and diverse simulation environment.

Reinforcement learning has been applied to hexapod and quadruped robots both as a standalone control method and in tandem with a central pattern generator-based control architecture. Hexapod and quadruped robots controlled using reinforcement learning have been proven to accomplish complex locomotion tasks in the literature. The following section details in table form many examples of reinforcement learning applied to hexapod robots.

## 2.4. Survey of Observations in Reinforcement Learning Research Applied to Hexapods in Combination with Central Pattern Generators

The work described in this thesis focusses on the observations used for reinforcement learning when applied to a hexapod robot walking problem. To demonstrate the variation in observations used in the current literature, a systematic review of the observations used by different authors in the literature is conducted and the results are presented in Tables 1 and 2. Note that, while these tables present an extensive overview of the state of the art in this field, they do not contain all possible relevant articles. To narrow the scope of the literature search, these tables only include work which uses a hexapod or quadruped robot platform as these robot configurations are similar in both performance and control requirements (bipedal robots are excluded from this review as they are not statically stable and thus may require different observations/measurements). Of key interest for this thesis work is to keep the sensor and processing power requirements of the robot platform to a minimum, so no complex exteroceptive sensors such as cameras or LiDAR are included. An exteroceptive sensor, such as an ultrasonic distance sensor, takes measurements and/or readings of the surroundings external to the robot body. On the other hand, a proprioceptive sensor takes measurements internal to the robot, such as the torque applied at a given joint. The review of the current literature focusses solely on work that utilises only proprioceptive sensors and exteroceptive sensors requiring little processing power as observations. The hexapod can also be provided with observations which are not measurements but commands to the reinforcement learning agent – such as a desired walking speed or direction. The scope of this thesis focusses on the use of reinforcement leaning for walking and gait modification in reaction to the environment or control commands such as walking speed or direction. In cases where a multi-level reinforcement learning control scheme is implemented, this review focuses on the lower-level leg control network rather than any higher-level path-planning networks.

To present the data extracted from the reviewed papers more clearly, the data is split into two tables. Table 1 is used to list how the reinforcement learning agent controls the robot and the performance goal of the robot in the context of the work, while Table 2 focusses on comparing the types of observations used between works. Table 1 lists the papers which fit the search criteria of reinforcement learning applied to a hexapod or quadruped robot using only sensors

which measure aspects of the hexapod's dynamics for the observations. Each entry in the table refers to the work of separate authors and they are listed in descending order of the date of publication with the most recent works at the top of the table, and with the first column providing the reference numbers. The third column lists the type of robot used in the study, be it a hexapod or quadruped platform. Column four describes how the reinforcement learning agent provides control to the robot, including if the work uses some form of central pattern generator and if the control signals are provided to the robot joints as input positions or torques. The fifth column in Table 1 describes the main goal of the reinforcement learning in the given paper, taken either from a statement in the paper and/or determined through analysis of the reward function used to train the RL agent. The final column indicates whether the described method was implemented on a real-world robot, if the work was simulation based, or both.

*Table 1: Comprehensive literature review of recent works involving hexapod mobile robots and reinforcement learning*

| Reference | Year Published | Robot Type | Control Method and How the RL Agent is Implemented | Main Goal of Learning (from Reward Function) | Exp / Sim |
|---|---|---|---|---|---|
| [26] | 2023 | hexapod | Hopf oscillator CPG with sinusoidal mapping functions whose parameters are adjusted by the RL agent | Track linear velocity commands for motion in x and y directions as well as yaw rate over different terrain | sim |
| [27] | 2023 | hexapod | RL agent outputs directly correlated to motion trajectories for robot joints | Walk straight at constant speed over rough terrain | sim |
| [28] | 2023 | hexapod | Cycloidal function defines robot feet trajectories, the RL agent outputs parameters of this function to modify the gait | Track commanded velocity while maintaining balance (level robot body) | both |
| [29] | 2023 | hexapod | Hopf oscillator CPG outputs are converted to leg tip trajectories using mapping functions, the RL agent then provides adjustments to these trajectories | Walk forward while keeping the robot body level across uneven terrain | sim |
| [30] | 2022 | quadruped | Leg trajectories generated by interpolating between set points, RL agent modifies TG parameters | Walk forward at a target velocity | both |

| Reference | Year Published | Robot Type | Control Method and How the RL Agent is Implemented | Main Goal of Learning (from Reward Function) | Exp / Sim |
|---|---|---|---|---|---|
| [31] | 2022 | quadruped | RL agent actions are scaled to input joint torques for the robot | Walk forward as quickly as possible | sim |
| [32] | 2021 | hexapod (damaged) | RL agent actions are scaled to input joint torques for the robot | Robot with damage (missing leg(s)) walks along straight line | sim |
| [33] | 2021 | quadruped | Joint position commands are the outputs from the RL agent | Maintain balance (level robot body) on moving ground/platform | both |
| [34] | 2021 | hexapod | RL agent outputs coupling parameters for a Hopf oscillator-based CPG which provides positions control for the robot joints | Fastest forward velocity with low energy consumption | both |
| [35] | 2021 | hexapod | RL agent actions are scaled to input joint torques for the robot | Fastest forward velocity with emphasis on efficiency (low energy consumption) | sim |
| [36] | 2021 | quadruped | RL agent outputs desired joint positions which are converted to input torques using a PD controller | Move forward in a straight line smoothly at set speed | both |
| [37] | 2021 | quadruped | Sinusoidal CPG with parameters adjusted by RL agent, with outputs converted to robot joint torques using a PD controller | Walk at commanded velocity using different base gaits | sim |
| [38] | 2021 | hexapod | RL agent actions are scaled to input joint angles of the robot | Walk in specified direction at target velocity over flat or stairs terrain | sim |
| [39] | 2021 | quadruped | RL agent outputs desired joint positions which are converted to input torques using a PD controller | Walk forward while dealing with changing environment conditions | sim |
| [40] | 2020 | hexapod (+ quadruped) | RL agent actions are scaled to input joint angles of the robot (or torques for quadruped) | Walk forward as fast as possible with known categorized damage | sim |
| [41] | 2020 | hexapod | Decentralized lower-level RL agents for each leg output actions which are scaled to joint angles controlling the robot's legs | Walk with maximum forward velocity | sim |
| [42], [43] | 2020, 2019 | quadruped | RL network outputs parameters for a sinusoidal TG and adjustments which | Follow direction provided by higher level path planning RL network | sim |

| Reference | Year Published | Robot Type | Control Method and How the RL Agent is Implemented | Main Goal of Learning (from Reward Function) | Exp / Sim |
|---|---|---|---|---|---|
| | | | are added to the TG signals to control the robot joints | | |
| [44] | 2020 | hexapod | RL agent actions are desired joint angles which are tracked using a servo mimicking feedback loop | Follow a breadcrumb trajectory in a complex terrain environment | sim |
| [45] | 2019 | quadruped | RL agent actions are modification parameters of independent sinusoidal TGs for each leg | Walk forward following desired speed profile | exp |
| [46] | 2019 | quadruped | Actions from RL agent are swing angle and extension of legs which are mapped to joint angles using a PD controller | Most forward progress in an episode (maximum speed) | both |
| [47] | 2018 | quadruped | RL agent actions are parameters and correction factors for sinusoidal TGs which produce the robot joint angles | Track desired/commanded speed forward | both |

A few observations and major trends in the control of hexapod robots using reinforcement learning can be extracted from Table 1. The two options for controlling the actuators of a robot are through commanded positions or commanded torques. Actuating the robot using commanded joint angles is the more popular of the two options, with 15 of the surveyed works implementing position control and only 6 utilizing torque-controlled actuators. Position control will be used in the work of thesis, following the majority of current research. Of the 21 surveyed works, 9 of them utilize some form of central pattern generator (or variants with other names such as trajectory generator or oscillator). The other 12 directly control the robot joints with the reinforcement learning agent actions, although scaling and filtering on the agent outputs is commonly required. However, just three of the direct-control cases were implemented on a physical robot platform (25 %) while over half of the papers using a central pattern generator had a real-world implementation (5 of 9, 55.6 %). The objective of this work is to allow for potential deployment on a hexapod robot platform, so for this reason a central pattern generator is chosen to be implemented. In terms of the learning objectives in the papers found in Table 1, 76.2 % (16 of 21) of the papers have, as the main goal for the robot to walk straight forward at either a specified speed or the maximum possible speed. In training a multi-legged

robot for a forward walking task, small deviations and perturbations provide the variation required to achieve a robust final gait. The work in this thesis will have, during training, a goal to have the robot follow a specified straight line as quickly as possible while starting each episode with the hexapod position initialized at a random offset from the specified line. By training the robot to move towards and follow a straight line it is hypothesized that the robot will be able achieve more complex path-following behaviour as outlined in the thesis objectives.

Table 2 is a continuation of Table 1 and provides a visual representation of the types of observations used in the selected papers. The observations listed as columns in the table are, the joint angles, joint angular velocities, body angles, body angular velocities, joint torques, body displacements, body velocities, body height, and ground contact. The table indicates (with a dot and green fill) if the particular observation is used in the given paper. There are also two additional columns which list any other unique observations used, and if the work also provides the previous time step's actions back to the reinforcement learning agent as observations. The body height is the distance between the robot's body and the ground and, although technically this is one of the robot's three displacements, it is separated from the body displacements category which refers to if the robot is provided with knowledge about its location in the environment according to a reference point. The column for ground contact also provides additional notes as to whether the contact information is provided as a measurement of forces or as a binary contact signal (contact or no contact).

*Table 2: Survey of observations used for reinforcement learning in recent works with hexapod mobile robots (next page)*

| Reference | Joint Angles | Joint Angular Velocities | Body Angles | Body Angular Velocities | Joint Torques | Body Displacements | Body Velocities | Body Height | Ground Contact | Other | Previous Actions |
|---|---|---|---|---|---|---|---|---|---|---|---|
| [26] | ● | ● | | ● | | | ● | | | projected gravity vector | ● |
| [27] | ● | ● | ● | ● | | ● | ● | | | | |
| [28] | | | | ● | | | ● | ● | | target velocity | ● |
| [29] | | ● | ● | ● | ● | | ● | | binary contact | | |
| [30] | | | | ● | | | ● | ● | | target velocity, states of TGs | |
| [31] | ● | | ● | | | ● | ● | | contact force | | ● |
| [32] | ● | ● | ● | ● | | ● | ● | ● | | | |
| [33] | ● | ● | ● | | ● | | | | | | |
| [34] | ● | ● | ● | | ● | ● | ● | | | | ● |
| [35] | ● | ● | ● | ● | ● | | ● | | force and torque at toes | | ● |
| [36] | ● | ● | ● | | | | | | binary contact | | ● |
| [37] | ● | ● | ● | ● | | | | ● | | | |
| [38] | ● | ● | ● | ● | | | ● | | binary contact | relative angle between robot and goal | |
| [39] | ● | ● | ● | ● | | | ● | | | | |
| [40] | ● | ● | | | | ● | ● | | binary contact | one hot encoding of damage type | |
| [41] | ● | | ● | | | ● | | | binary contact | | ● |
| [42], [43] | | | ● | ● | | | | | | command vector from higher level network, TG state vector | |
| [44] | ● | ● | ● | | | | | | binary contact | terrain classification | |
| [45] | ● | | ● | | | | ● | | | phase of each TG, previous 4 time steps observations | |
| [46] | ● | | ● | ● | | | | | | previous 5 time steps observations | ● |
| [47] | | | ● | ● | | | | | | phase of TGs, desired velocity | |

Table 2 shows the various combinations of observations used in the literature. The papers are listed in order of date of publication with the most recent entries at the top of the table (the same order as Table 1). Table 2 clearly shows that, while some observations may be more common than others, there is no consensus within the literature on which set of observations to use for reinforcement learning locomotion of a hexapod or quadruped robot.

The most common proprioceptive observation within the surveyed papers is the measurement of body angles or tilt, as might be measured using an inertial measurement unit (IMU) onboard the robot body. An IMU is both inexpensive and relatively easy to include in the chassis of a robot build, making it a good candidate for widespread adoption. The body angles are also relatively easy to obtain in simulation during training. The inclusion of an IMU also allows for the measurement of the body velocities and angular velocities without the addition of any sensors – which perhaps explains their popularity as a choice of observation. The second most popular observation amongst the surveyed papers are the joint angles themselves. The state-of-the-art in the literature is for the robot joints to be controlled using some form of central pattern generator, so even if the robot is controlled via position control, the outputs of the reinforcement learning agent are typically not directly the joint positions. The joint positions are provided back to the reinforcement learning agent to provide context as to how the agent's outputted actions affect the robot's movement. Robotic joints are often actuated with servo motors, whose internal positions sensors allow the joint positions to be obtained without additional hardware.

Studying the data displayed in Table 2, there appears to be a trend in the literature of the use of an increasing number of observations, with more complex measurements such as the joint torques, body displacements, and body heights used more in recent research. The ground contact as an observation is used in 8 of the 22 surveyed works, and 6 of those 8 implemented the contact measurement as a binary signal (contact or no contact) as opposed to a measurement of force. Intuitively it seems that providing the reinforcement learning agent with information about the contact of each foot would improve performance; however, only about one third of the works listed utilized these signals. More investigation is needed to determine if ground contact observations are not used because they are not effective, or if they are not used due to the increased difficulty in obtaining a contact measurement from physical hardware on an actual

robot. The present work hopes to produce results that could help make objective data-driven decisions on the benefits of various observations including ground contact measurements.

While each of the reviewed papers presents a unique hexapod reinforcement learning scenario, none provide an in-depth explanation or reasoning as to why the observations used were selected for the particular case. The work in the present thesis explores the use of a designed experiment to provide an objective method for selecting the combination of observations that maximize the desired performance objective.

# Chapter 3: Simulator Development

The following chapter details the development of the hexapod robot simulation platform, and how both a central pattern generator and reinforcement learning were applied to the simulator to meet the first two thesis objectives.

## 3.1. Hexapod Learning Task

One of the goals of this work is to determine if a fractional factorial designed experiment can be used to aid in the selection of sensors/measurements to use when designing a hexapod robot for a particular reinforcement learning-based locomotion task. Therefore, this thesis will carry out a study to determine if a factorial designed experiment can select the optimal observations to use for a path following locomotion task.

The path following task is illustrated in Figure 1. The hexapod will start at a random offset perpendicular from a desired trajectory line as shown in the figure. The goal of the reinforcement learning is for the hexapod to correct its initial offset by tracking towards and then following along the goal trajectory line. Since the hexapod will be controlled using a central pattern generator, the robot will initially be able to make forward progress but must learn to adjust its gait to steer and follow the desired path as quickly as possible.

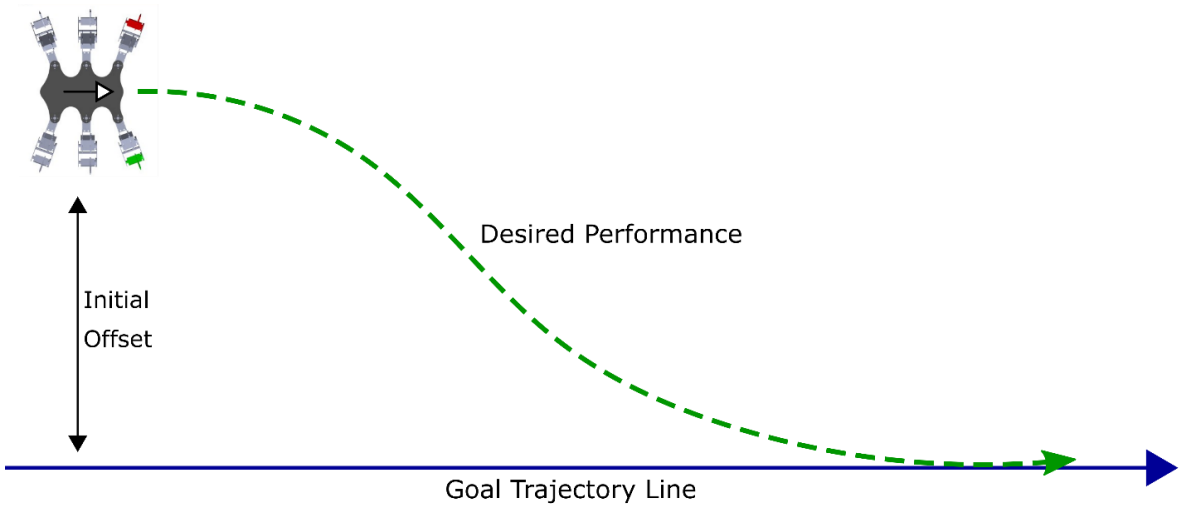



*Figure 1: Desired Hexapod Trajectory Tracking Behaviour*

## 3.2. Overall Simulator Approach

The work in this thesis is carried out with the application of an autonomous or remotely-controlled hexapod robot in mind. Figure 2 illustrates the scope of the autonomous hexapod control covered in this thesis. The control of a hexapod robot can be broken down into two levels of control: a high-level which includes some form of navigation or path planning, and a low-level which is reactive to the environment in order to follow the desired path. The higher-level control could consist of a neural network, a human operator using a remote control, or any other suitable methods. The high-level system determines the desired robot path through the environment, and also selects the base gait type that will be used in the locomotion (such as tripod or wave gaits). The lower-level control system consists of a central pattern generator and a reinforcement learning agent which follows the desired path using the provided base gait while adding the ability of the hexapod to react to disturbances and inputs from the environment. As shown in Figure 2, the higher-level control is not included in the scope of this work. It is assumed that both a path and base gait have been predetermined and are provided to the lower-level control system as inputs. The specific focus of this thesis is to study the effect of measurements/feedback provided to the reinforcement learning agent in a lower-level control system on the ability of the hexapod to follow the desired trajectory provided by a higher-level control system.

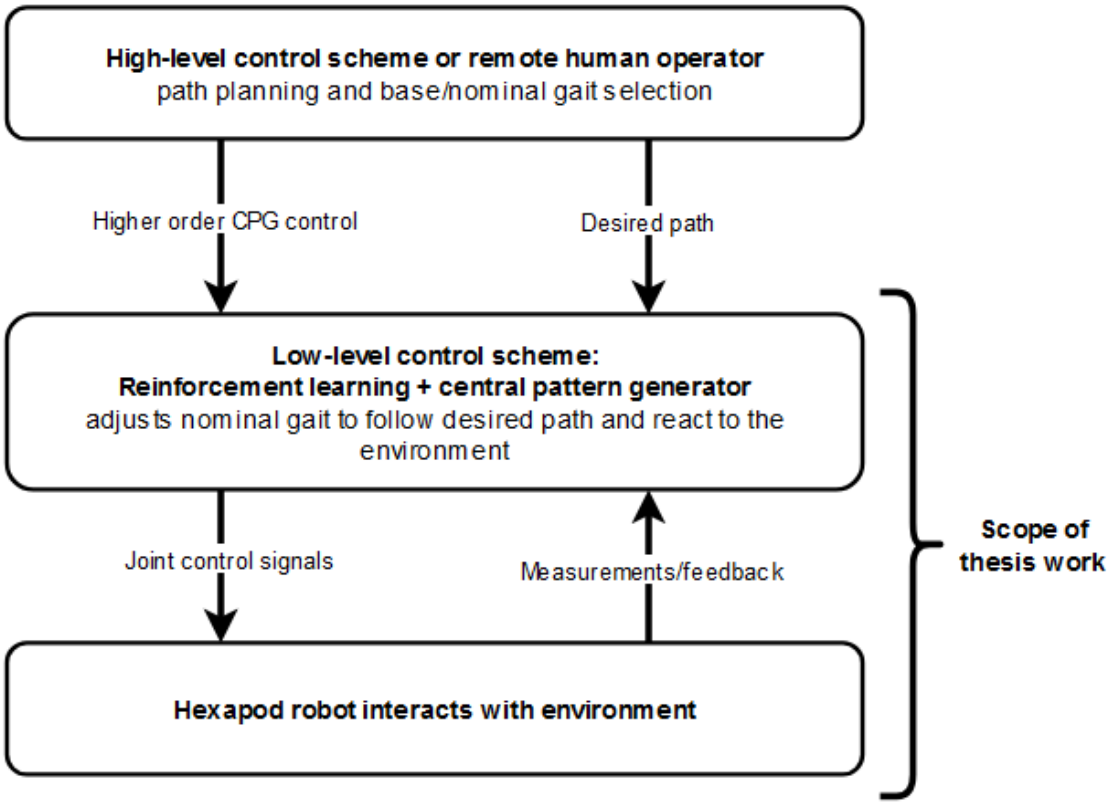



*Figure 2: High level hexapod control flowchart*

## 3.3. Structure of Simulator and Flow of Data

To meet the first thesis objective, a hexapod simulator was developed using MATLAB's Simulink environment using the Simscape Toolbox to simulate the rigid multibody dynamics and ground contact dynamics. The overall structure of the hexapod simulator developed in this thesis is illustrated in the flowchart shown in Figure 3, which follows a similar structure to [32] and [48]. The simulator can be broken down into three key sections that govern the various aspects of the hexapod control: the reinforcement learning section, the central pattern generator, and the hexapod simulation environment itself. The Simulink model is called by a driving routine which defines the parameters used in the simulator. Referring to Figure 3, the simulator operates in a loop with the flow of data as follows:

- The reinforcement learning (RL) section takes in measurements from the simulation environment and outputs the RL Agent's actions which will be used to control the hexapod.
- The RL Agent's actions are converted into joint signals by the central pattern generator and then sent to the hexapod simulation environment.
- The simulation environment applies these joint signals to the hexapod and records the measured response to be sent back to the reinforcement learning agent to start the cycle over again.

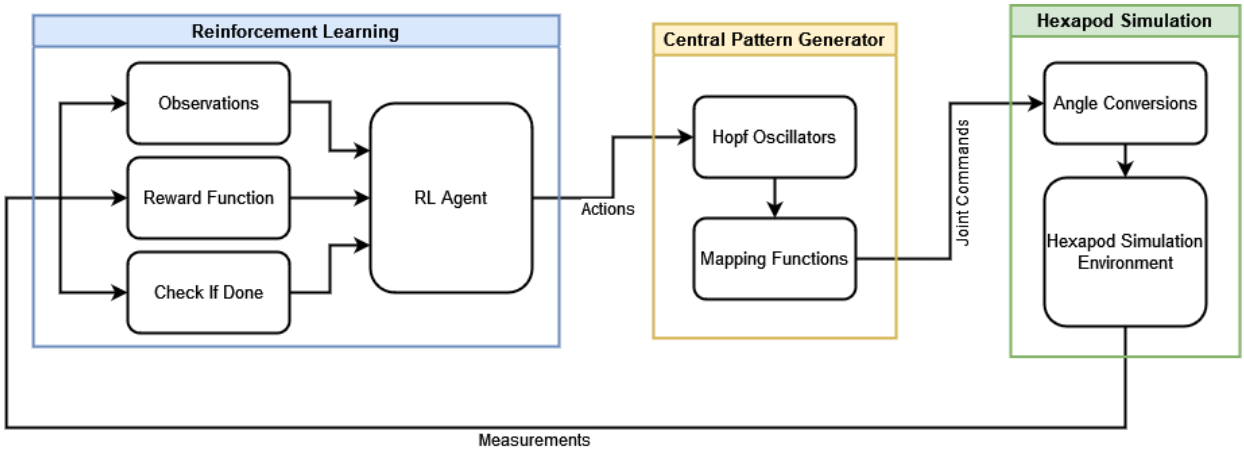



*Figure 3: Hexapod Reinforcement learning simulation block diagram*

## 3.4. The Hexapod Model and Simulation Environment

The hexapod robot is modelled after the Fire Ant robotics platform shown in Figure 4 sourced from Orion Robotics Inc. [49]. The Fire Ant is an Arduino-based platform which utilizes digital servo motors to control each of its joints. The hexapod has 18 degrees of freedom dedicated to walking. The robot has additional degrees of freedom in the form of a movable tail and head with independently-controllable pincer jaws; however, the hexapod robot to be studied in this work consists of the Fire Ant platform with the head and tail removed.

*Figure 4: Fire Ant Robot and Simulated Robot with head and tail removed*

The hexapod robot is modelled in SolidWorks by measuring the physical robot. The important aspects to include in the model are the key geometric features such as limb lengths and component masses. The SolidWorks model is then imported into SimScape Multibody using the SolidWorks to Simscape converter software. The modelling of the Fire Ant hexapod in SolidWorks was completed by Jeffrey Woodacre, a previous Dalhousie University graduate student [50]. Note that no electrical components other than the servo motors are modelled, as they do not contribute significantly to the total mass when compared to the servo motors and aluminum frame components.

To provide an idea of the overall scale of the Fire Ant hexapod, the following statistics are presented. The size of the hexapod body panels is 227 mm x 170 mm (length x width). The total length of each leg when fully straightened is 385 mm. The mass of the hexapod (with head and tail removed) is 1.49 kg.

The simulator is built using MATLAB R2021a and Simscape. This software was selected as the MATLAB package has the ability to handle both the hexapod simulation environment and it has built-in toolboxes to support DDPG reinforcement learning. This work could be accomplished using many different software packages, so the selection of MATLAB R2021a is not critical to the results.

The hexapod simulation environment consists of a flat ground plane with which the hexapod can interact. The hexapod can contact the ground plane through 6 points located at the tips of the legs. The dynamics of these interactions are modelled using the Spatial Contact Force

block in Simscape [51]. The hexapod's legs have rubber tips for improved grip on hard surfaces, so the parameters of the contact interaction are set to model this interaction as listed in Table 3. The coefficient of static friction was measured using a tilt test, while the coefficient of dynamic friction was estimated based on comparing the distance traveled per step in simulation to real-life testing using a robot. The remaining parameters are taken from [48] as it features a similar walking environment.

*Table 3: Spatial Contact Force Block Parameters*

| Spatial Contact Force Block Parameters for Hexapod Leg to Gound Plane Contact | |
|---|---|
| Coefficient of static friction | 0.5 |
| Coefficient of dynamic friction | 0.3 |
| Critical velocity (m/s) | 0.001 |
| Ground stiffness (N/m) | 1000 |
| Ground damping (N/(m/s)) | 100 |
| Ground transition width | 0.0001 |

Each leg of the hexapod has three degrees-of-freedom: one hip joint which allows the leg to swing laterally about the central body, and two joints (knee and ankle) in the plane of the leg for bending and extension. Figure 5 shows the range of motion of the hip angles for each of the six legs. Each hip angle is limited to 0.3 rad of motion which is the maximum allowable movement without any overlap of the ranges of motion from the other legs. If the range of motion of the legs were allowed to overlap, then there would be potential for self-collision – a complication that is not dealt with in this work.

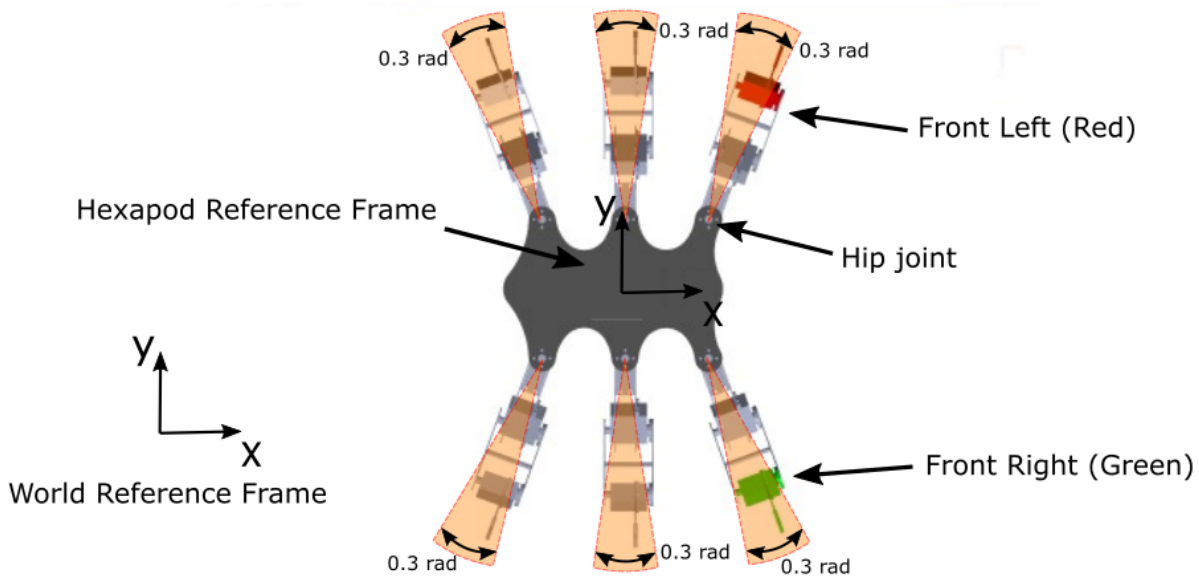



*Figure 5: Top view of hexapod showing range of motion of hip angles*

The two degrees-of-freedom in the plane of the leg are designated the knee and ankle joints and have ranges of motion of 0.5 rad and 0.275 rad, respectively. The full in-plane range of motion of a leg assembly is illustrated in Figure 6. This range of motion is sufficient for the task of walking/navigating on a flat terrain as is the case for the study conducted in this work. In the case of more extreme terrain that may include significant and abrupt changes in elevation, the range of motion of the legs could be increased to improve the hexapods performance. Limiting the range of motion of each leg simplifies the simulation by ensuring that only the tip of each leg can come into contact with the ground.

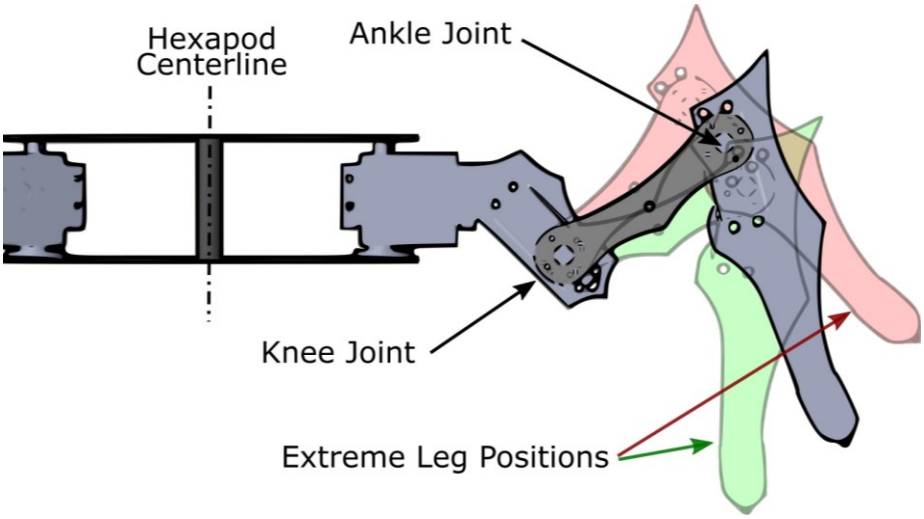



*Figure 6: Range of motion of hexapod leg*

## 3.5. Joint Position Command Filtering

The Fire Ant robot's 18 joints are actuated using position-based control with digital servo motors. The hexapod control system uses a central pattern generator which outputs relatively smooth joint angle signals; however, due to adjustments to these signals by the reinforcement learning agent operating in discrete time at a fixed sample rate, there will be small discontinuities in the resulting joint signals. These discontinuities are illustrated in Figure 7 which shows a sample joint angle signal taken from the first episode of a reinforcement learning training routine. In order to simulate a servo motor's response to the commanded joint angle signal, filtering is used to smooth out any small discontinuities. The corresponding filtered joint position signal is also shown in Figure 7.

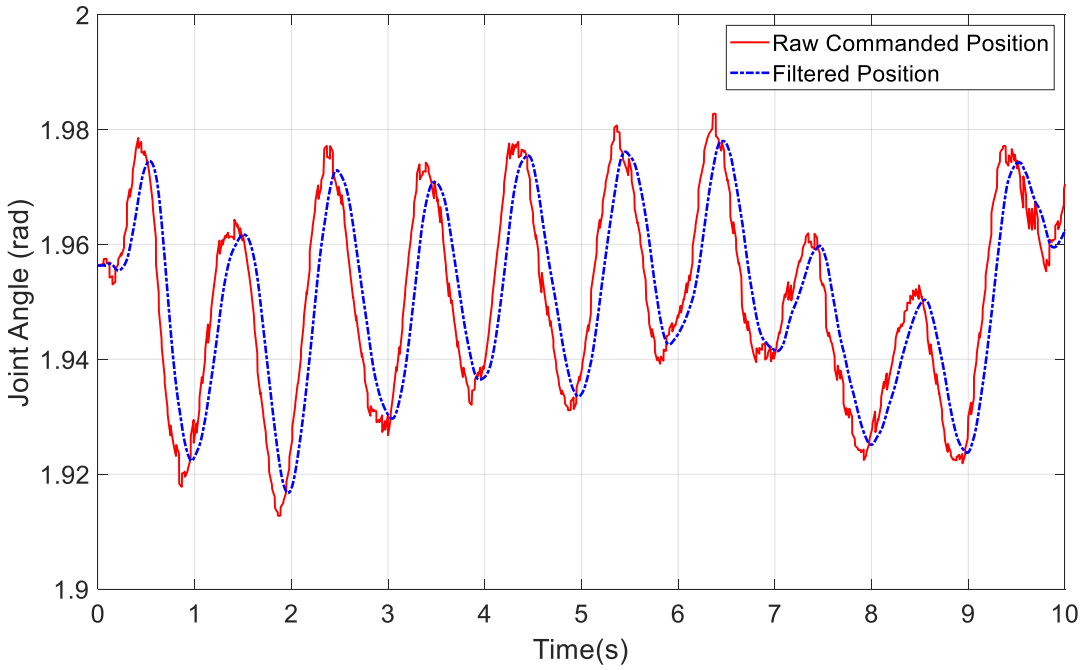



*Figure 7: Commanded and filtered joint angles*

The filtering is achieved within the Simulink to Physical Signal converter block in Simscape [52] before the signals are sent to the revolute joint blocks within the hexapod model. This block is necessary as a link between the reinforcement learning portion of the Simulink model and the Simscape hexapod and environment model. Second order filtering must be selected in order to provide the first- and second-order derivatives (velocity and acceleration) of the joint position signal to the revolute joint blocks in the hexapod model. Each time the converter block calculates a derivative it uses a first order low pass filter – the transfer of function of which is shown in the following equation:

$$H(s) = \frac{1}{\tau s + 1} \tag{1}$$

where the time constant τ is set to 0.05 seconds. This value was determined through hand tuning and is a balance between filtering capability and minimization of lag. The time constant should be large enough that the filtering removes any large jumps in position signal that the real servo motors would not be able to track due to speed limitations. It is also important to minimize the

lag induced by the filtering so that the reinforcement learning can better correlate its action commands with how they affect the hexapod's joint positions.

## 3.6. Central Pattern Generator

The hexapod control system uses a central pattern generator in combination with a set of mapping functions to produce smooth joint angle signals while offering precise control over the motion. The oscillators and mapping functions described by Wang *et al.* [23] for use with a genetic optimization algorithm are built upon and modified in this work to function with a reinforcement learning agent.

The central pattern generator consists of six coupled Hopf oscillators which, through the adjustment of various parameters, offers control over the amplitude, frequency, and phase angle between the hexapod's six legs. The Hopf oscillator is a proven basis for central pattern generation applied to hexapod robots [23], [29], [34], [53]. The single Hopf oscillator is described by the following mathematical model [23]:

$$\dot{x} = \alpha(\mu - x^2 - y^2)x - \omega y \tag{2}$$

$$\dot{y} = \beta(\mu - x^2 - y^2)y + \omega x \tag{3}$$

where the sinusoidal output of the Hopf oscillator state variables $x$ and $y$ can be controlled by a number of parameters, ω is the oscillator frequency, $\sqrt{\mu}$ is the amplitude, and α and β are the convergence velocities (where $\alpha > 0$ and $\beta > 0$). The Hopf oscillator allows for fine control of the behaviour of the state variable outputs which are then used to drive the rhythmic walking motion of the hexapod. The following figures are based on those from Wang *et al.* [23] and demonstrate the effects of the different parameters on the Hopf oscillator outputs.

Figures 8 and 9 illustrate how the frequency of the oscillator outputs (both x and y) can be controlled by changing just the oscillator frequency parameter ω. This parameter directly controls the frequency independently from the other key oscillator characteristics.

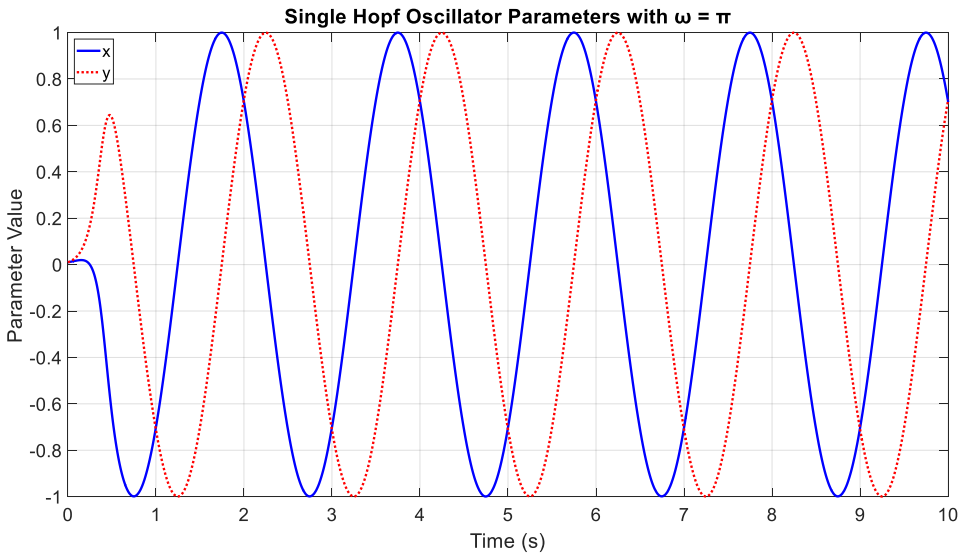



*Figure 8: Hopf oscillator state variables with ω=π*

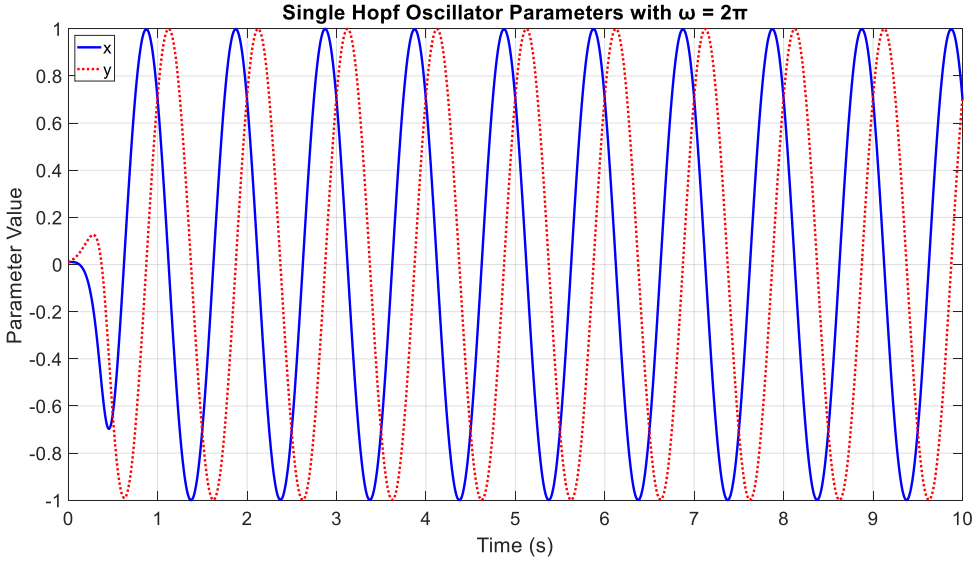



*Figure 9: Hopf oscillator state variables with ω=2π*

Changing the oscillator parameter μ directly controls the amplitude of the output signals *x* and *y* a shown in Figures 10 and 11.

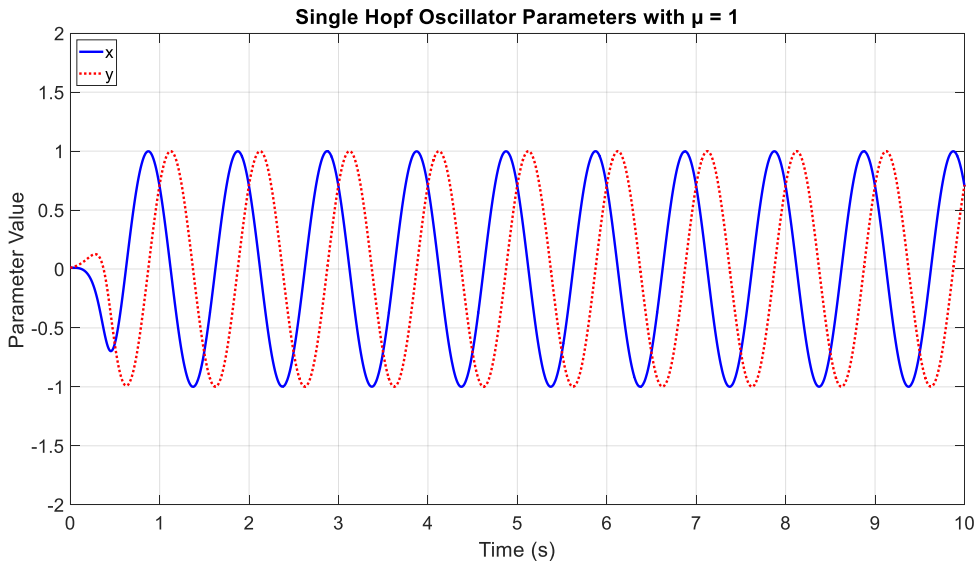



*Figure 10: Hopf oscillator state variables with μ=1*

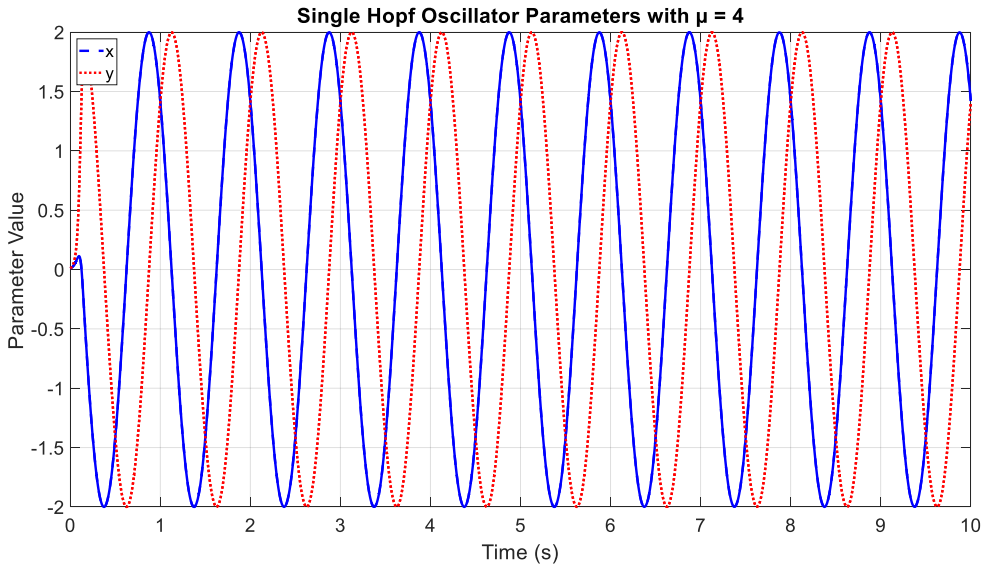



*Figure 11: Hopf oscillator state variables with μ=4*

Finally, Figures 12, 13 and 14 demonstrate how the convergence velocity β affects the behaviour of the Hopf oscillator. The higher the value of β, the faster the oscillator outputs converge to their final steady state sinusoidal oscillations.

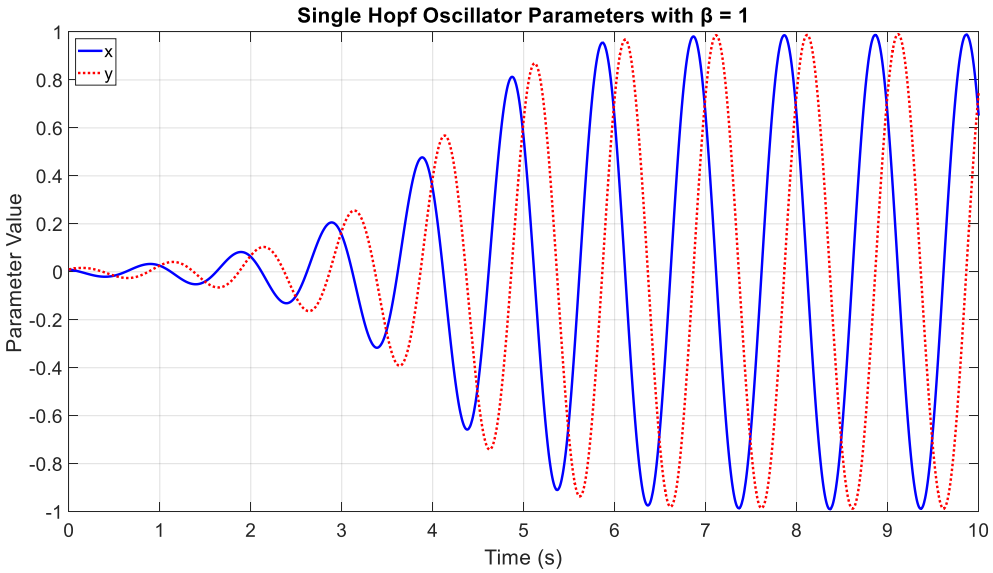



*Figure 12: Hopf oscillator state variables with β=1*

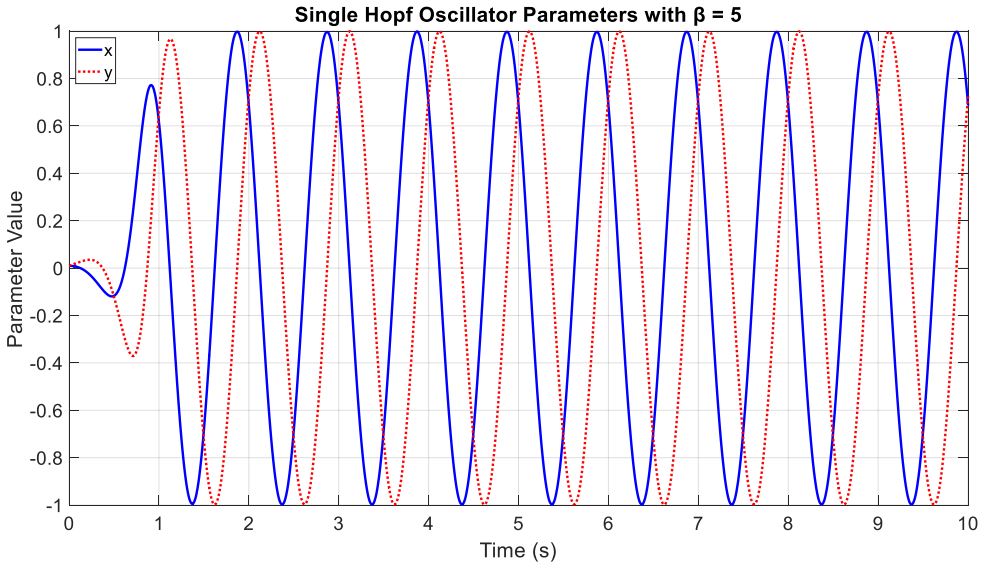



*Figure 13: Hopf oscillator state variables with β=5*

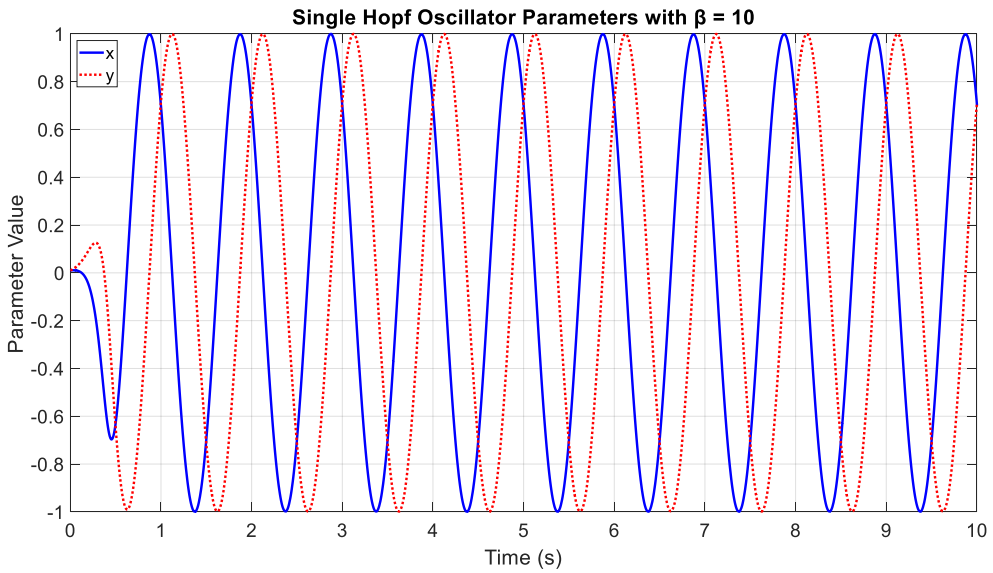



*Figure 14: Hopf oscillator state variables with β=10*

In order to produce the complete set of oscillatory signals that are used to drive the motion of the hexapod, the central pattern generator uses six coupled Hopf oscillators. Each leg of the robot is controlled by an individual oscillator and the inter-oscillator coupling determines the phase differences between the hexapod's legs. The previously-described Hopf oscillator is modified to include an additional coupling term to govern the interactions between the six legs of the hexapod robot. Each of the coupled Hopf oscillators are described by the following equations [23], [29].

$$\dot{x}_i = \alpha(\mu - {r_i}^2)x_i - \omega_i y_i \tag{4}$$

$$\dot{y}_i = \beta(\mu - {r_i}^2)y_i + \omega_i x_i + \delta \sum\nolimits_j \Delta_{ji} \tag{5}$$

$$\Delta_{ji} = y_j \cos\left(\theta_j^i\right) - x_j \sin\left(\theta_j^i\right) \tag{6}$$

where once again, $x$ and $y$ are the oscillator state variables, ω is the frequency, $\sqrt{\mu}$ is the amplitude, and β is the convergence velocity. The subscripts $i$ and $j$ indicate the oscillator/leg numbers (between 1 and 6). The new parameters dealing with the oscillator coupling are the

coupling strength between oscillators δ, the coupling value $\Delta_{ji}$, and the phase angle $\theta_j^i$ between oscillator $i$ and oscillator $j$.

The phase angles between the oscillators $\theta$ is a matrix representing the coupling connections between oscillators and, therefore, hexapod legs. Wang *et al.* [23] implements fully-symmetric bidirectional coupling between the six oscillators; however, it was found that using a simpler method of leader oscillators and followers also produces the desired result. In this work one of the oscillators is selected as the leader, and the phase angles for the remaining five oscillators are all based on the phase angle of the leader. Using this leader-follower method simplifies the phase difference matrix $\theta$ without compromising the ability of the hexapod to achieve various base gaits. Two of the most common hexapod gaits, derived from the study of insects in nature, are the tripod and wave gaits. The tripod gait is the fastest hexapod gait which can be achieved while maintaining dynamic stability through three points of contact with the ground at all times. On the other hand, the wave gait is a slow and stable gait where each leg moves independently – all equally spaced with a phase difference of $\pi/3$. The phase difference matrix used to obtain the tripod and wave gaits are as follows:

$$\theta_{tripod} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -\pi/2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -\pi/2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -\pi/2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$\theta_{wave} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -\pi/3 & 0 & 0 & 0 & 0 & 0 \\ -2\pi/3 & 0 & 0 & 0 & 0 & 0 \\ -\pi & 0 & 0 & 0 & 0 & 0 \\ -4\pi/3 & 0 & 0 & 0 & 0 & 0 \\ -5\pi/3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

The coupled Hopf oscillators generate a continuous output irrespective of the initial conditions which means that the central pattern generator is able to switch between hexapod base gaits while still providing a continuous output. Figure 15 shows the central pattern generator

switching from a tripod gait to a wave gait and back to tripod gait. The figure shows the first state variable, $x$, for each of the six oscillators and demonstrates the phase difference between oscillators (and therefore legs). The oscillators signals are initially separated in two groups of three with a phase difference of $\pi/2$ for the tripod gait, then at 5 seconds transition to each have their own unique phase angle in a wave gait arrangement before returning to the tripod gait at 10 seconds.

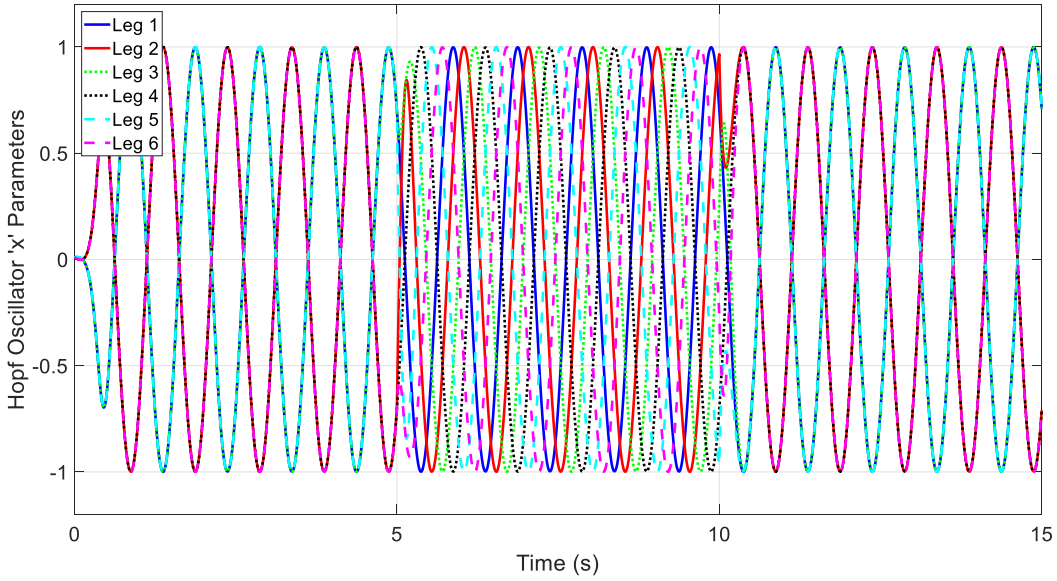



*Figure 15: Transition from tripod gait to wave gait and back to tripod gait*

During a gait transition the leader leg (leg 1 in this case) keeps the same sinusoidal motion while the remaining legs adjust their phases accordingly. In the case of bidirectional coupling between all six oscillators, all legs would adjust their phases in a distributed manner to achieve the new gait. There are potential advantages to each gait; however, for the purposes of this work the tripod gait is used exclusively so the leader-followers method is sufficient.

The coupling strength δ can be tuned to control how quickly the oscillator outputs adjust to a change in the phase angles in a similar fashion to how the convergence velocity affects each individual oscillator. Figure 16 shows a switch from wave gait to tripod gait with a coupling strength of 10 leading to a rapid and more abrupt change in base gait.

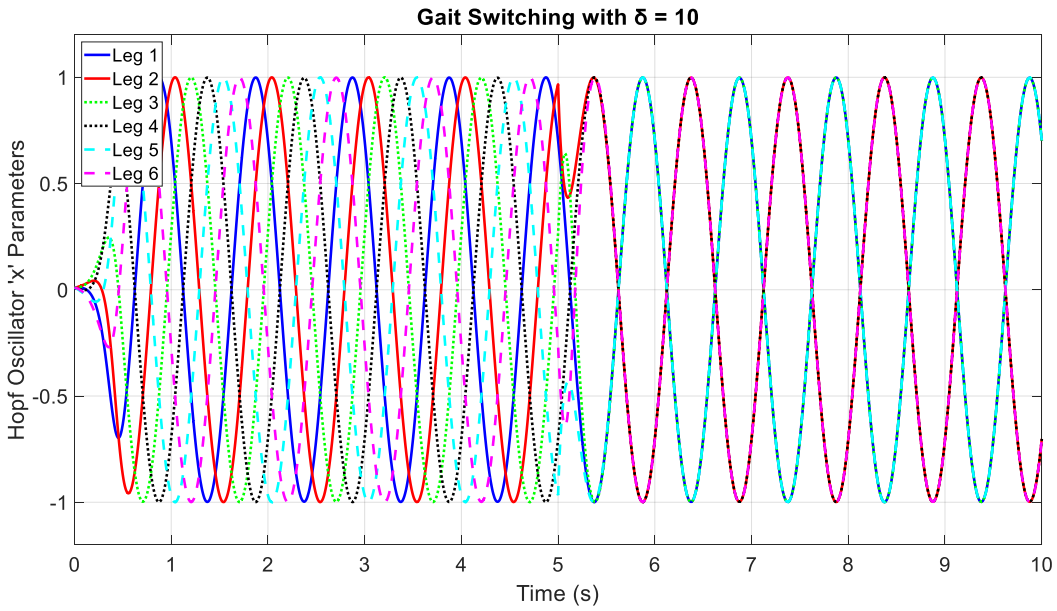



*Figure 16: Gait transition with δ=10*

If the coupling strength is reduced the change in oscillator output due to the abrupt change in phase angles is a more smooth and more blended (albeit slower) transition. This output is shown in Figure 17.

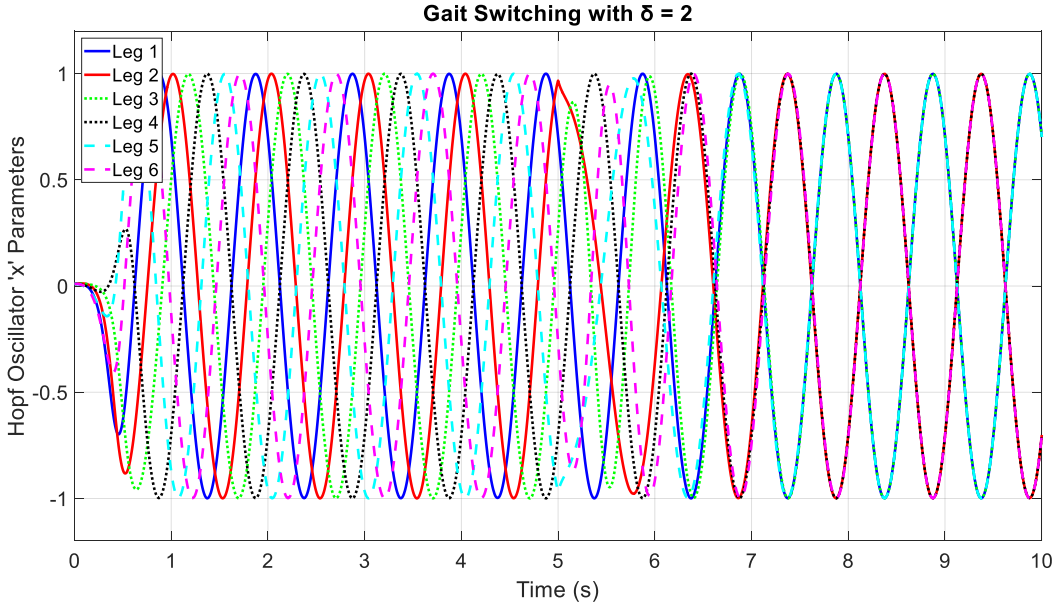



*Figure 17: Gait transition with δ=2*

Mapping functions are used to transform the Hopf oscillator *x* and *y* signals into joint angle signals for each of the hexapod's 18 degrees of freedom. There are six coupled oscillators, each corresponding to a leg on the hexapod, so there are also six sets of mapping functions (one for each individual leg). The mapping functions take the state variables *x* and *y* from the Hopf oscillator and convert them into three angle signals for the hip, knee and ankle joints of the corresponding leg. The mapping functions used in this work are based on those presented by Wang *et al.* [23] but are modified to better suit reinforcement learning (as opposed to the genetic optimization algorithm for which they were originally proposed). The mapping functions presented by [23] utilise piecewise functions for both the knee and ankle angles to differentiate between the swing and stance phases of the leg motion. In the present work, these piecewise functions are replaced with a single function for all conditions as the reinforcement learning agent is able to infer when each leg is in a swing or stance phase from the provided observations, and the agent can also adjust the mapping function parameters in reaction to external stimuli. The mapping functions corresponding to each oscillator are as follows (where the subscript *i* indicates the oscillator/leg number):

$$\theta_{1,i} = k_{0,i} y_i \tag{9}$$

$$\theta_{2,i} = k_{1,i} x_i + b_{1,i} \tag{10}$$

$$\theta_{3,i} = k_{2,i} \theta_{2,i} + b_{2,i} \tag{11}$$

where $\theta_{1,i}$ indicates the hip angle for leg $i$, $\theta_{2,i}$ is the knee angle for leg $i$ and $\theta_{3,i}$ is the ankle angle also for leg $i$. $k_0$, $k_1$ and $k_2$ are proportionality factors, and $b_1$ and $b_2$ are bias values in the mapping functions which can be tuned to modify the base gait. The proportionality factors can therefore be expressed as a matrix of size 6×3, with three parameters for each of the six legs. The biases can be similarly written as a single matrix of size 6×2. These parameters are the means through which the reinforcement learning agent controls the motion of the hexapod. The agent is able to adjust all of the proportionality factors and biases in real time in order to modify the underlying base gait set by the coupled Hopf oscillators. The total number of adjustable parameters is 30 corresponding to three proportionality factors and two biases for each of the hexapod's six legs.

The ranges for which the mapping function parameters can be adjusted by the reinforcement learning agent are shown in Table 4. These values were determined through manual tuning in order to provide a balance between exploration capabilities of the potential action space and learning speed.

*Table 4: Mapping function parameter ranges*

| Parameter | Minimum Value | Maximum Value |
|---|---|---|
| $k_0$ | 0 | 0.3 |
| $k_1$ | 0 | 0.3 |
| $k_2$ | 0 | 0.3 |
| $b_1$ | -0.1 | 0.1 |
| $b_2$ | -0.1 | 0.1 |

## 3.7. The Reinforcement Learning

To meet the second thesis objective, DDPG reinforcement learning for the hexapod robot is implemented in the hexapod robot simulator through the use of the Simscape Reinforcement Learning Agent block [54]. This agent block requires three inputs: the set of observations provided to the agent, the calculated reward signal, and a "check if done" signal to indicate the end of each episode. The reinforcement learning parameters are provided to the Agent Block in the simulation driving routine. The following sections discuss each aspect of the reinforcement learning and how they are related to the hexapod simulation environment.

### 3.7.1. The “Check If Done” Function

The “check if done” function receives as input the measured body tilt angles from the simulation environment and produces a binary signal based on the orientation of the hexapod. This signal indicates when to abandon a training episode early due to the robot becoming inverted. If any of the three body angles exceed the limits shown in Table 5, then the training episode will be immediately terminated.

*Table 5: Maximum body angles for termination conditions*

| | |
|---|---|
| Max Pitch | ± 90° |
| Max Roll | ± 90° |
| Max Yaw | ± 360° |

The pitch and roll are limited to 90° to terminate the episode if the hexapod becomes inverted. Since the hexapod is not permitted to experience being inverted, the result is that the robot can never learn to invert itself and will never be able to reorient itself if it becomes inverted during deployment. This work focusses on training the hexapod to steer and the issue of dealing with inversion is out of the scope of this thesis. Smaller roll and pitch limits were initially used in an effort to limit the amount of body tilt present in the final gait; however, this resulted in the hexapod learning to flip itself over right at the start of each episode to quickly terminate the episode, rather than incur negative reward penalties throughout the course of the entire 15 second episode. The large roll and pitch limits are required to allow the hexapod to explore its entire range of motions without triggering a premature episode termination.

The maximum yaw angle is set to be 360° to allow for the hexapod to steer and follow the goal path, but also stops the training if the hexapod ends up in a situation where it has entered a repeating pattern such as walking in a circular path or spinning on the spot; however, this problem was not encountered during the training regime used in this work.

### 3.7.2. The Reward Function

The reward function is used to guide the learning of the RL agent towards the desired behaviour. In this case, the reward function is designed to train the hexapod to follow the desired trajectory line both rapidly and smoothly. The reward function contains both positive and negative reward terms to encourage positive actions and discourage undesired actions, respectively. The reward terms are calculated using measurements from the simulation environment. The total reward is calculated as shown in Equation 12 with the individual reward terms detailed in Table 6.

$$R_{total} = R_{velocity} - R_{tilt} - R_{offset} - R_{bodyheight} + R_{constant} \quad (12)$$

*Table 6: Detailed breakdown of reward function terms*

| **Reward Term** | **Reward Type** | **Equation** |
|---|---|---|
| Forward velocity | Positive reward | $R_{velocity} = C_1 v_x$<br>where:<br>- $v_x$ is the component of the velocity of the hexapod (measured at the body center) along the stationary global frame's +x axis (which also aligns with the desired straight-line trajectory)<br>- $C_1$ is the reward term scale factor |
| Body tilt | Negative penalty | $R_{tilt} = C_2\left({\theta_{pitch}}^2 + {\theta_{roll}}^2 + {\theta_{yaw}}^2\right)$<br>where:<br>- $\theta_{pitch}$ is the pitch angle of the hexapod body in radians<br>- $\theta_{roll}$ is the roll angle of the hexapod body in radians |

| **Reward Term** | **Reward Type** | **Equation** |
|---|---|---|
| | | - $\theta_{yaw}$ is the yaw angle of the hexapod body in radians<br>- $C_2$ is the reward term scale factor |
| Offset from goal trajectory | Negative penalty | $$R_{offset} = C_3\lvert\Delta z\rvert$$ where:<br>- $\Delta z$ is the offset of the hexapod body center from the desired trajectory line taking the shortest distance perpendicular to the desired trajectory line<br>- $C_3$ is the reward term scale factor |
| Body height | Negative penalty | $$R_{bodyheight} = C_4(h - h_{desired})^2$$ where:<br>- $h$ is the height of the hexapod body from the ground plane, measured from the bottom of the hexapod<br>- $h_{desired}$ is the desired body height set for the hexapod<br>- $C_4$ is the reward term scale factor |
| Constant term | Positive reward | $$R_{constant} = C_5\frac{t_s}{t_f}$$ where:<br>- $t_s$ is the sample time of the reinforcement learning<br>- $t_f$ is the training episode length (final time)<br>- $C_5$ is the reward term scale factor |

The most important reward terms are the forward velocity term and the offset term as these provide the driving force for the RL agent to learn the desired path-following behaviour. The forward velocity term is a positive reward given to encourage forward movement of the hexapod in a direction parallel to the goal path. This reward term uses the component of the hexapod's absolute velocity in the global x direction (parallel to the goal trajectory). This term becomes negative if the hexapod moves in the opposite direction. There is no goal velocity or speed of movement. The forward velocity reward simply encourages the hexapod to move as quickly as possible. The top speed of the hexapod is limited by the frequency of the base gait built into the central pattern generator and the range of motion of the legs, so rewarding the maximum possible speed will theoretically lead to a tripod gait optimized for speed.

The offset reward term is a penalty term calculated by taking the absolute value of the perpendicular distance from the hexapods body center to the goal trajectory line. The offset reward term produces an increasing negative reward the further the hexapod is from the goal line on either side. The offset reward is the driving force behind the reinforcement learning agent being able to steer the hexapod.

There are two additional penalty terms in the reward function which, while not crucial to the overall goal of path following, are used to provide further control over the characteristics of the learned gait. The body tilt reward term penalizes rotation of the hexapod body in any of the three axes (pitch, roll and yaw) with the goal of limiting oscillation of the hexapod body and increasing the smoothness of the learned gait. This reward term could be especially important in a case where the hexapod is transporting a delicate payload or a payload which must be kept level during motion. To make the smoothness of the gait a greater priority of the learning, the reward term factor for the tilt penalty could be increased. The tilt penalty does include the yaw angle which might seem counterproductive when the hexapod is required to turn to follow the goal trajectory; however, including the yaw angle ensures that the hexapod will walk straight along the trajectory line in the desired orientation without crab walking, and the benefit gained by turning towards the goal trajectory line to reduce the offset penalty outweighs the additional yaw tilt penalty.

The body height reward term is used to set the desired height of the hexapod's body for the learned gait, ranging from a low crouch to a tall stance with legs fully extended. This reward

term would become more important if the hexapod was trained on uneven terrain where ground clearance might become an issue.

The final term of the reward function is a small constant reward given each time step to encourage the hexapod to make it to the end of each training episode without triggering the early termination conditions. During initial testing, one peculiar case emerged where the behaviour learned by the hexapod was to immediately flip itself over at the beginning of each episode to trigger the termination condition. As mentioned previously, the RL agent learned that, by terminating each episode as quickly as possible, it could avoid collecting negative reward penalties that it would incur during exploration of further behaviour. The reinforcement learning agent had become caught in a local maximum. The addition of the constant reward term helps to remedy this problem by giving the RL agent some positive reward just for completing each training episode in full, even if it is not yet achieving the desired behaviour. Increasing the tilt tolerance for the termination condition was also used to increase the hexapod's exploration ability and allow it to escape local maxima during training.

Each term of the reward function has a corresponding scale factor which is used to tune the importance/weight of the individual reward terms. These scale factors allow the user to decide which of the reward terms to prioritize during training of the hexapod gait. In this case, priority was given to the forward reward term and the offset penalty which have the largest impact on the ability of the RL agent to achieve the goal of learning to follow a desired trajectory. The final scale factors used in this thesis, which were obtained through a hand-tuning process, are shown in Table 7. Note that the relative values of the scale factors do not directly correspond to the relative weight/importance given to each reward term as the reward terms are not normalized and their scales differ.

*Table 7: Reward function scale factor values*

| Reward Term Scale Factor | Value |
|---|---|
| $C_1$ | 50 |
| $C_2$ | 5 |
| $C_3$ | 50 |
| $C_4$ | 50 |
| $C_5$ | 1000 |

### 3.7.3. Observations

The following sections outline the set of observations available to the reinforcement learning agent. All selected observations utilize sensors internal to the hexapod robot, meaning that the hexapod could be operated in an unstructured environment. These observations do not consist of an exhaustive list of all potential sensors/observations for a hexapod but are the chosen set which could be used on a hardware platform such as the Orion Robotics Fire Ant. The selected observations are prevalent observations used in the literature for hexapod and quadruped robots as identified in Table 2 of Section 2.4. There are some observations which are always used and are, therefore, not included as factors in the designed experiment. These observations are either deemed critical for the desired performance of the hexapod and/or are readily available without the use of additional sensors. All observations are normalized to between −1 and +1 before being passed to the neural network.

### 3.7.4. Observations Included in All Designed Experiments

The following observations were included in all the designed experiment runs:

### *Joint Angles*

In this work the joint angles are considered a critical observation for the success of the learning and are always provided to the reinforcement learning agent. Preliminary tests showed that the learning ability is significantly negatively impacted if the joint angles are not included in the set of observations. The hexapod is controlled by providing joint angle commands to the 18 servo motors, which track the commanded angles using built-in rotary potentiometers. No additional sensors are required to monitor the current joint angles. For these reasons the 18 joint angles are always provided as observations and do not change throughout the designed experiment. The joint angles obtained directly from the hexapod simulation environment are first converted back into the nominal angle space in which the center of travel of each joint is designated as the origin. They are then normalized by dividing the nominal measured angle by the maximum possible angle for the given joint. The maximum possible (nominal) joint angles are listed in Table 8.

*Table 8: Maximum joint angles*

| Angles (6 each) | Maximum Range of Nominal Joint Angle (±) |
|---|---|
| Hip $\theta_1$ | 0.15 rad |
| Knee $\theta_2$ | 0.25 rad |
| Ankle $\theta_3$ | 0.1375 rad |

### *Joint Angular Velocities*

The joint angular velocities are also always included in the set of observations. The joint angles are already being measured; therefore, it takes no additional sensors and minimal effort to also include the angular velocities. In order to determine the maximum possible value that could be expected for normalization of the measured angular velocities, three cases were considered with the aim of encompassing the full range of potential angular velocities. The maximum angular velocity was recorded during the first episode of training, after the agent has been fully

trained for 1000 episodes, and for the nominal gait case with the reinforcement learning removed from the loop. The maximum recorded angular velocities for each case were 1.72 rad/s, 3.14 rad/s and 0.86 rad/s, respectively. The largest value (from the fully trained case) was taken and an additional safety factor of 1.2 was added to obtain the angular velocity normalization factor of 3.77 rad/s – the value by which each measured angular velocity is divided before being passed to the reinforcement learning network.

### *Hopf Oscillator Parameters*

The Hopf oscillator parameters are the internally-calculated $x$ and $y$ values that are produced by the Hopf oscillator central pattern generator before they are transformed into joint angle signals via the mapping functions (refer to Section 3.6). The six $x$ parameters of the Hopf oscillators are provided as observations to the neural network to provide context into the current phase of the oscillators. These observations provide information to the reinforcement learning agent about the phase and frequency as well as the type of base gait (e.g. tripod or wave gaits) for the hexapod motion. There is no normalization necessary as the Hopf oscillator $x$ parameters are, by nature, confined to be between −1 and +1.

### *Previous Time Step Actions*

The previous time step's actions are fed back to the neural network as observations. The outputted actions are already between −1 and +1 so no normalization is required.

### *Offset Observation*

The final unchanging observation that was included in all designed experiments is the offset from the goal trajectory line. This observation is not measured using sensors but provided to the neural network from a higher-level control system for path planning. This offset observation is the novel method to enable the hexapod to know its position relative to a desired path. The hypothesis is that, by training using a straight goal trajectory in combination with the offset observation, the hexapod will develop a robust behaviour able to follow more complex

paths. When normalizing the offset to between −1 and +1 it is desired to maintain a continuous function. The normalization should also prioritize the region closest to the goal trajectory (where offset $\Delta z = 0$) as this region is where the hexapod is most likely to be operating. In this work it is highly unlikely that the hexapod will be more than a few meters from the goal trajectory, but the normalization should be able to deal with large deviations without producing any results that could lead to errors when fed to the neural network. The hyperbolic tan function was selected as the normalization method to comply with these requirements. The measured offset (in meters) is divided by two before the hyperbolic tan is taken to produce a function which is approximately linear between −2 and +2 (shown in Figure 18 – which is the expected working region of the hexapod for this research. Unlike simply saturating the offset value when the hexapod is far away from the goal trajectory, the hyperbolic function will still produce a different normalized value for each unique position, while remaining bounded between −1 and +1.

$$\text{offset observation} = \tanh\left(\frac{\text{offset } \Delta z \text{ in meters}}{2}\right) \tag{13}$$

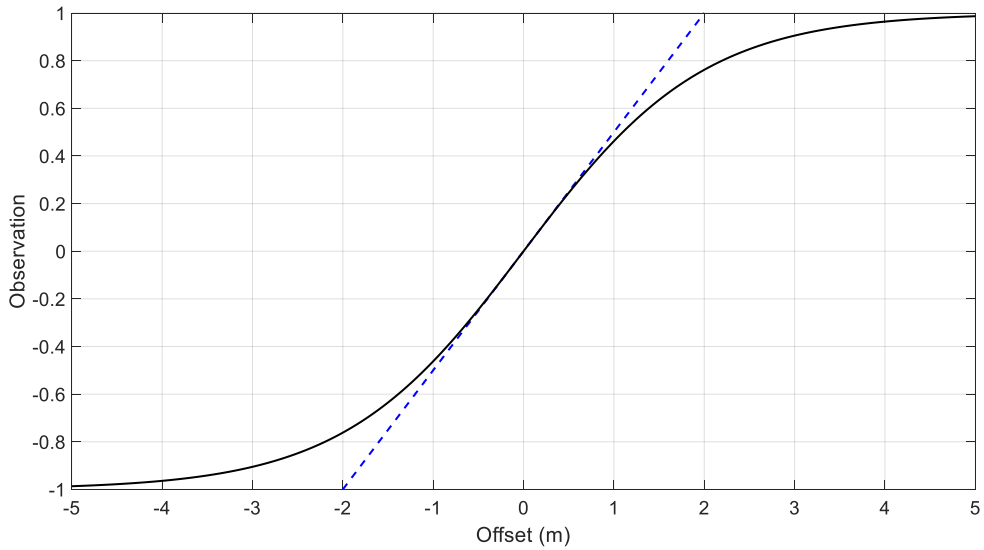



*Figure 18: Offset observation value with respect to the perpendicular offset from goal trajectory line*

Having defined the critical observations which were used in all of the designed experiment runs, the following section details the set of observations used as factors in the

designed experiment. The following observations are turned on and off according to the designed experiment run plan to produce unique combinations of observations.

### 3.7.5. Observations to Vary for the Designed Experiments

The following observations were studied as factors in the designed experiment, where they were systematically turned on or off (present or not present) according to the experimental run plan described in Chapter 4:

#### *Joint Torques*

The joint torques can be measured in the simulation directly from the joint blocks, but need to be normalized according to a maximum expected value. This maximum torque value is calculated for a worst-case scenario where the entire mass of the robot is being lifted by a single ankle joint with the leg at full extension to maximize the moment arm. This scenario could occur if one of the hexapod's legs becomes stuck/fixed in the ground terrain (although no such terrain is used in this study the normalization method is still useful). The full leg extension which provides the worst-case moment arm for the mass of the robot is illustrated in Figure 19. This distance of 254.5 mm is used to calculate the maximum expected torque.

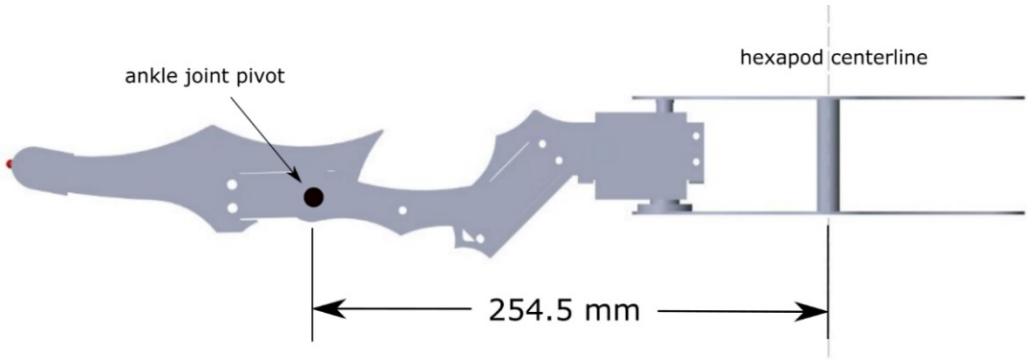



*Figure 19: Maximum hexapod leg moment arm*

The torque normalization value is calculated by assuming the entire mass of the hexapod is placed at the distance shown above from the pivot. The motor torque required to counteract this moment is determined to be 3.71 N·m or 37.85 kg·cm and is calculated as follows.

$$T_{max} = m_{total} g L_{extend} \tag{14}$$

$$T_{max} = 1.486\ kg \times 9.81\ m/s^2 \times 0.2545\ m$$

$$T_{max} = 3.71\ N \cdot m = 37.85\ kg \cdot cm$$

The 18 joint torques are normalized by dividing by the maximum expected torque. The torque signals are saturated to eliminate any large spikes. Spikes can occur due to the nature of the simulation leg tip to ground contacts, and always occur at the start of the simulation when the hexapod first descends onto the ground plane.

### *Body Velocities*

The three hexapod body velocities are measured at the center of the hexapod body as would be done using some form of inertial measurement unit (IMU). To determine the value to be used to normalize the body velocities, three test cases were used: the hexapod behaviour after training for one episode, after training for the full 1000 episodes, and the nominal gait case with no reinforcement learning. The maximum recorded velocity (ignoring large initial spikes) from these three cases was 0.64 m/s in the *x* direction from the fully trained hexapod. An additional safety factor of 20 % was added to obtain the velocity normalization value of 0.77 m/s. As with the joint torques, the normalized velocities are saturated between −1 and +1 to eliminate large spikes that can occur at the start of a training episode.

### *Body Angles/Tilt*

The body angles, orientation, or tilt of the hexapod are the roll, pitch, and yaw angles of the hexapod's body relative to the world reference frame as would be measured with an

accelerometer or IMU. These observations are provided to the reinforcement learning agent in the form of a quaternion; therefore, no normalization is required.

### *Body Angular Velocities*

The body angular velocities about the three axes of roll, pitch, and yaw can also be provided as observations. As with the body tilt, these observations are based on the center of the hexapod body relative to the world reference frame, as would be measured by an accelerometer or IMU. To normalize the angular velocities, the same method used for the body velocities is employed to determine the maximum expected value. The maximum recorded angular velocity during testing was 3.14 rad/s for the case of a fully trained agent controlling the hexapod. Multiplying by a safety factor of 1.2 results in the normalization value of 3.77 rad/s.

### *Body Height*

The body height observation measures the distance between the underside of the hexapod body and the ground plane. The point of measurement on the hexapod body is the center of the body underside, so the body height measurement is minimally affected by the body tilt of the hexapod. The height is measured by taking the shortest distance between the center of the hexapod underside and the ground plane, so when the body is tilted the measurement line will no longer be perpendicular to the robot body. Measurement of the body height could be obtained using a downward facing infrared sensor, or even could be estimated based on knowledge of the leg's positions and ground contacts. The use of this observation becomes more difficult when the terrain is not a flat plane, but it would still be possible to obtain a useful measurement. Even a series of distance measurements could be obtained to provide some information on the terrain profile to the reinforcement learning agent. The body height observation is normalized by dividing the measured body height by the maximum obtainable body height on flat ground with the legs fully extended as shown in Figure 20. Full leg extension produces a maximum body height of 209.7 mm, and the observation is saturated at this value to ignore the initial body height when the hexapod drops onto the ground when each episode is initialized.

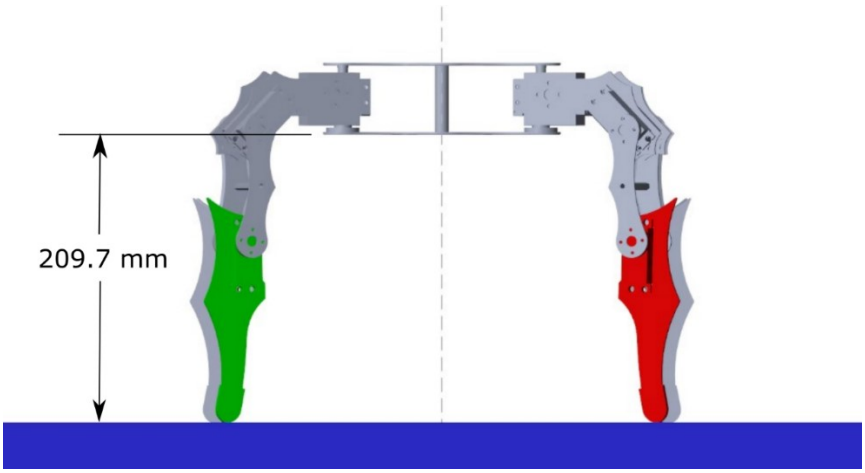



*Figure 20: Maximum body height under full leg extension*

### *Leg Tip Ground Contacts*

The final observation that was varied for the designed experiments are the ground contacts between the hexapod leg tips and the ground plane. The contacts are modelled in the simulation as point contacts located at the very tip of the hexapod's legs. The Spatial Contact Force block used in the simulation provides the magnitude of both the normal and frictional forces at the point of contact. The maximum possible force is taken as the full weight of the hexapod supported by a single leg. The contact observations are normalized by dividing the measured force by 14.58 N, which corresponds to the total weight of the hexapod. This normalization uses only the static weight of the hexapod and, therefore, does not account for an increase in contact force magnitudes due to the dynamics of locomotion. It is therefore possible for the contact force at a given leg tip to exceed the static weight of the hexapod, but it is extremely rare for the hexapod to place all of its weight on a single leg as a minimum of three legs must be in contact with the ground for the robot to be statically stable. The contact observations are saturated at the total weight of the hexapod to eliminate any unwanted spikes in the signals.

Another simplified representation of the contact forces seen in the literature can be used as observations by converting the contact normal forces into binary signals: +1 if contact between leg tip and ground is detected, and 0 otherwise. The measured forces and binary

representations lead to 3 sets of 6 possible observations: the binary contacts, the contact normal forces, and the contact frictional force magnitudes.

### 3.7.6. Additional Notes Regarding the Observations

There was an extensive testing and exploration period that focussed on the various measurements available as potential observations before the completion of the final study. This section consists of a few notes and lessons learned that were observed for the simulation conditions used in this research which may be useful for future research.

1. The use of the body accelerations as observations had a negative impact on performance. This result seems counterintuitive since one would think that giving the reinforcement learning agent more information would be beneficial. Through analysis of the observation signals one can see why this might not always be the case. Figures 21a-21c are generated by extracting the position, velocity, and acceleration signals of the hexapod body center-point in the vertical direction from a randomly-selected reinforcement learning training episode. The measurement signals are then converted into observations by normalizing them so that each signal has a range of 0 to +1. At a glance, one can see that the acceleration observation differs from its position and velocity counterparts even though they are all describing the same cyclic walking movement. Figure 21a shows how the vertical position of the hexapod body follows cycling motion resembling a sinusoid during walking. The velocity profile also has a similar sinusoidal appearance as shown in Figure 21b. The lower troughs of the velocity profile are more pointed as the velocity changes abruptly when the hexapod steps switching the legs that are in contact with the ground. The effect of the stepping is magnified in the acceleration signal which shows large spikes when legs of the hexapod contact the ground plane during walking. The acceleration spikes are so severe that they dominate the normalized signal, rendering the observation less useful to a reinforcement learning agent. This issue could potentially be avoided with the use of a filter or saturation on the acceleration signal, but for the scope of this research it was not considered as the position and velocity signals had already

shown to provide sufficient information to obtain the desired performance. It might be interesting to study the acceleration spikes on a variety of terrain, as a soft sand surface for example should result in lower spikes than the hard ground considered in this work.

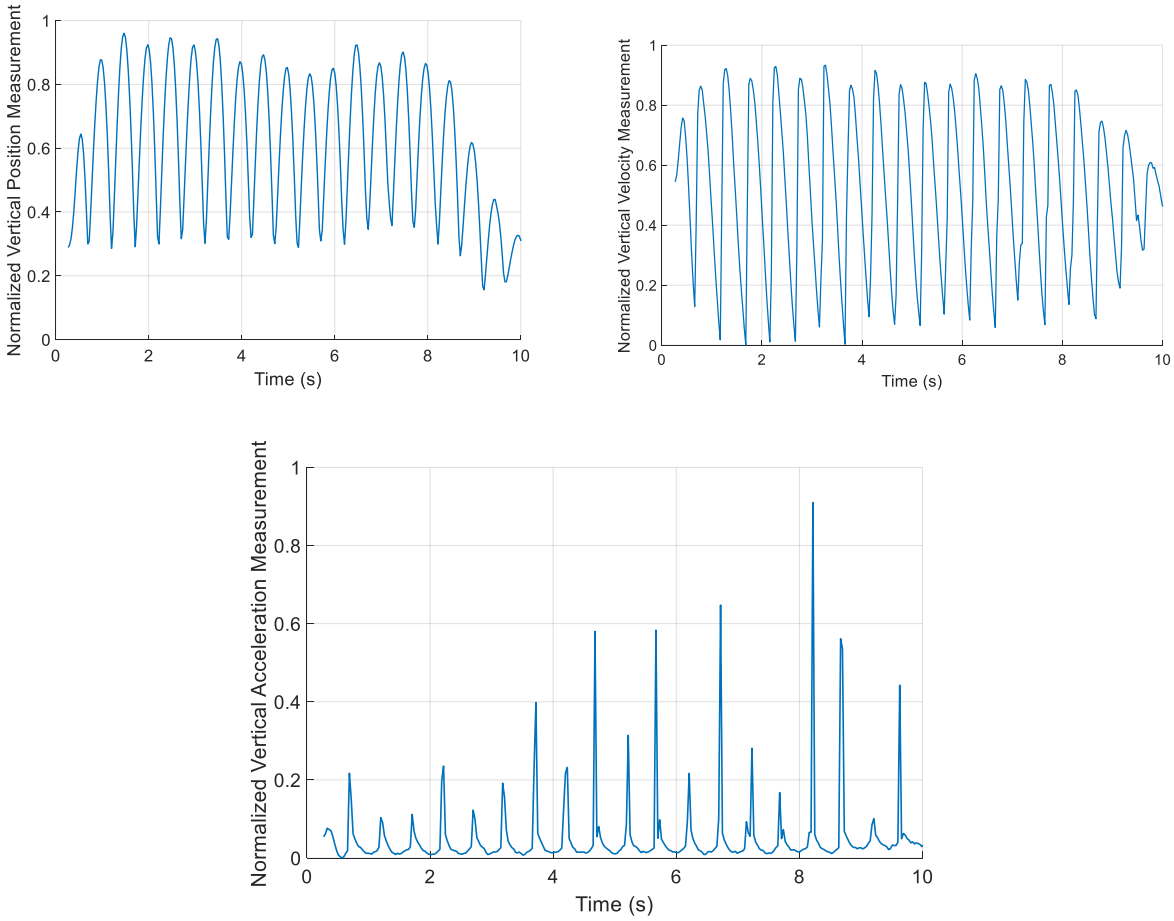



*Figure 21: Position (a), velocity (b) and acceleration (c) of hexapod body in the vertical direction*

2. Any observation that cannot experience the full range of its expected values over the course of a training episode was found to be detrimental to the performance of the reinforcement learning. The negative impact of such observations was observed during testing with the inclusion of the hexapod body displacement relative to the initial position as a potential observation. During training, the episode length is fixed so the hexapod is only able to travel a certain distance before its position is reset – with the maximum travel over a fixed time limited by the frequency selected in the central pattern generator. Using the displacement from initial position as observation, the hexapod appeared to

learn a satisfactory gait during the training episodes, but the learned behaviour was not suitable for deployment. If the learned agent was deployed and the simulation was left to run for a long period of time, the learned gait completely broke down when the hexapod surpassed the maximum displacement from the training episodes. It should be noted that the joint angles are also a position-based observation, but since they are of a cyclic nature, the reinforcement learning agent is able to experience the entire range of possible joint position values during the short training episodes.

3. Measuring the actual joint angles using a sensor may not be necessary, although including joint angles as an observation is crucial. The joint positions are an important observation for the reinforcement learning agent to infer how the outputted actions affect the hexapod through the central pattern generator. However, it is not actually necessary to use a sensor to measure the joint angles as the position command signals produced by the central pattern generator were found to be sufficient. As discussed in Section 3.5, these signals differ only by the filtering applied to simulate a servo motor response time. The actual servo position does lag behind the commanded position but, for the simulation conditions used in this research, it did not appear to affect the learning performance. The use of commanded positions as measured joint positions is well suited to the use of a central pattern generator where the maximum commanded motor speed can be limited so the servo can closely track the commanded position. Note, however, that this approach fails when the commanded motor speed far exceeds the servo motor's maximum speed or if the servo reaches stall torque and is unable to track the commanded position.

## 3.8. The Reinforcement Learning Agent

The hexapod is controlled by a reinforcement learning agent trained using the Deep Deterministic Policy Gradient (DDPG) [55], which has been proven effective for legged robotics locomotion applications [28], [32], [34], [35]. In this work, DDPG reinforcement learning is applied to the central pattern generator-based control presented by Wang *et al.* [23], which had previously been combined with a genetic algorithm for optimization.

DDPG is an actor-critic reinforcement learning method, meaning that two neural networks are required. The actor network contains the learned policy that will ultimately control the hexapod when deployed after training, while the critic network is utilized solely during the training phase. The actor and critic networks applied in this work utilize the same structure and size as those found in [48], which applies reinforcement learning to a quadruped robot with the same magnitude of observations and actions.

The actor network takes as input the observations from the hexapod simulation and outputs the next time step actions with the goal of maximizing hexapod performance. The actor network consists of an input layer for the observations, two fully-connected hidden layers, and a fully-connected output layer as shown in Figure 22. The input layer size changes based on the number of observations used in the trial, but the remaining layers maintain a fixed number of nodes. The two hidden layers and output layer have 400, 300 and 30 nodes, respectively. The hidden layers utilize a ReLU (rectified linear unit) activation function while the output layer uses a hyperbolic tangent activation function to produce the 30 actions with values constrained between −1 and +1.

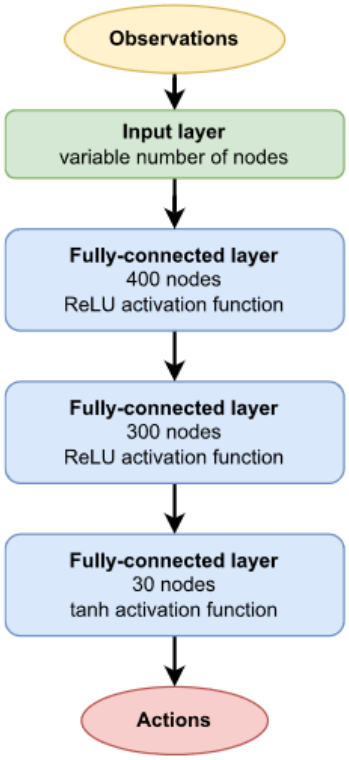



*Figure 22: Actor network flowchart representation*

The optimizer parameters set in the MATLAB simulation driving script for the actor network are listed in Table 9. It was found that the parameters used for the quadruped in [48] provided satisfactory results without needing additional time spent tuning these parameters. The actor network uses the Adam (adaptive movement estimation) gradient descent method [56] with a learning rate of 1e−3. The gradient threshold is set to 1 and the L2 regularization factor is set to 2e−4.

*Table 9: Actor network optimizer parameters*

| Actor Network Optimizer Parameters | |
|---|---|
| Optimizer | Adam |
| Learning Rate | 1e−3 |
| Gradient Threshold | 1 |
| L2 Regularization Factor | 2e−4 |

The critic network takes as input the observations from the hexapod simulation and the action signals produced by the actor network, and outputs the expected reward that will be obtained using the given actions. As shown in Figure 23, the critic network consists of two initially-separate branches that merge to produce a single output. The first branch has an input layer of the observations (changing layer size) followed by two fully-connected hidden layers of 400 and 300 nodes. The second branch takes as input layer the 30 actions from the actor network and then has a fully-connected hidden layer of 300 nodes. Both branches of the network are connected with an addition layer that appends the two branches into a single layer of 600 nodes. Then follows a final fully-connected layer of a single node which produces the critic output. All hidden layers use ReLU activation functions while the final output layer does not have any activation function to generate the critic output.

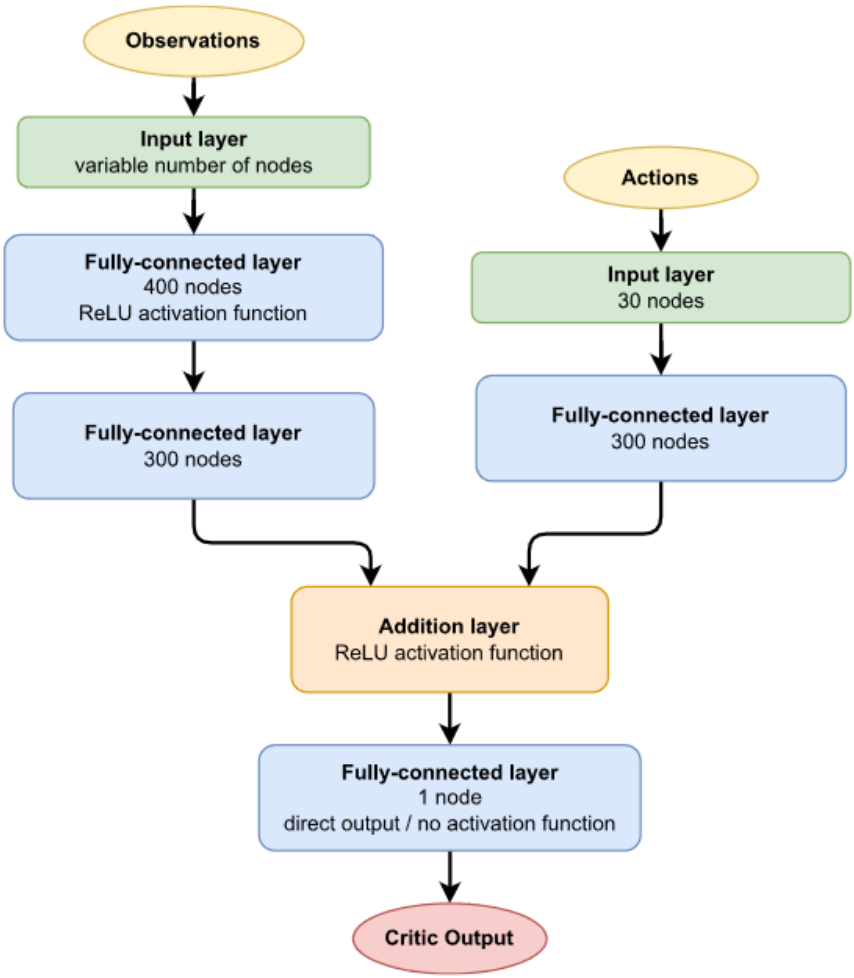



*Figure 23: Critic network flowchart representation*

As with the actor network, the optimizer parameters set in the MATLAB simulation driving script for the critic network are listed in Table 10. The critic network also uses the Adam (adaptive movement estimation) gradient descent method but with a learning rate of 1e−4. In the case of the critic network gradient threshold is set to 1 and the L2 regularization factor is set to 1e−5.

*Table 10: Critic network optimizer parameters*

| Critic Network Optimizer Parameters | |
|---|---|
| Optimizer | Adam |
| Learning Rate | 1e−4 |
| Gradient Threshold | 1 |
| L2 Regularization Factor | 1e−5 |

## 3.9. Reinforcement Learning Training Routine

The following section outlines the reinforcement learning training procedure that is used to train the DDPG agent for a given set of parameters in the designed experiment.

Each training routine consists of 1000 individual episodes lasting 15 seconds each. The training hyperparameters set for the reinforcement learning using the available MATLAB training options are shown in Table 11. These values were determined by starting with values used in [48], and then fine tuning them by hand before completing the designed experiment study. The parameters in Table 11 remain fixed throughout the designed experiment.

*Table 11: DDPG learning hyperparameters set in driving routine*

| Reinforcement Learning Training Hyperparameters | |
|---|---|
| Number of Episodes | 1000 |
| Sample Time (s) | 0.03 |
| Discount Factor | 0.99 |
| Mini Batch Size | 250 |
| Experience Buffer Length | 1e6 |
| Target Smoothing Factor | 1e−3 |
| Noise Mean Attraction Constant | 0.15 |
| Noise Standard Deviation | 0.1 |

Before each training episode a reset function is executed which is used to improve the robustness of the learning policy by randomizing the hexapod starting position for each training episode and, therefore, increasing the variety of the hexapod's experiences. In this research, the starting position of the hexapod is randomized according to a random uniform distribution between an offset of −1 to +1 meters from the goal trajectory line (in a direction perpendicular to the goal trajectory line). An example distribution of the starting points for 1000 trials is shown graphically in Figure 24.

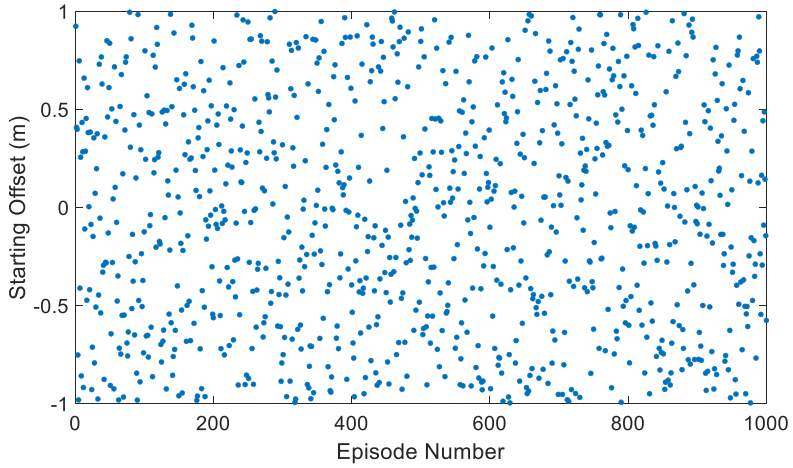



*Figure 24: Visual representation of the random starting offset during training*

Each episode during training runs for the full 15 seconds unless the hexapod becomes inverted, in which case the angle of tilt exceeds the stopping condition and the episode is terminated early. It is important to note that the simulation does not run at real time when training. A single training routine of 1000 episodes takes roughly 8-10 hours to simulate, demonstrating the importance of minimizing the number of trials used to select observations via a designed experiment approach. The simulations are carried out on a PC with an Intel Core i7-10700K processor, Nvidia GeForce RTX 3070 GPU, and 16 GB of ram. Although up to four separate instances can be run at once, it is still a considerable time commitment just for the training of all trials.

# Chapter 4: The Designed Experiment

To meet the third and fourth thesis objectives, this section explores the use a fractional factorial designed experiment to gain insight into the relative importance of a set of possible observations for the reinforcement learning agent. A factorial experiment refers to the systematic study of different factors affecting a given response variable. The goal of this study is to determine whether a factorial designed experiment could be used as a tool to select the most important observations from a set of potential measurements for a specific hexapod trajectory-following case study.

## 4.1. Design of Experiments Background Information

Originally developed for use in the fields of agriculture and industrial manufacturing, design of experiments is a statistical methodology used to plan experiments that can provide insight into the effects of various factors or parameters on a process result in the most efficient way possible. The results of a designed experiment can be used to build a mathematical model of the process which can then be used to make informed decisions about the process and its factors.

Take, for example, a case where a designed experiment is used to study the effect of three factors ($X_1$, $X_2$ and $X_3$) on a given process result $Y$. The process is assumed to be approximated by a linear regression model as shown in Equation 15 [57].

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_{12} X_1 X_2 + \beta_{13} X_1 X_3 + \beta_{23} X_2 X_3 + \beta_{123} X_1 X_2 X_3 + \epsilon \quad (15)$$

where $Y$ is the process result, $X_1$, $X_2$, and $X_3$ are the three main factors to be studied, the first-order interaction terms are $X_1X_2$, $X_1X_3$, and $X_2X_3$, and finally $X_1X_2X_3$ is the single second-order interaction term present in this model. Interactions occur when the effect of one factor on the response is dependent on the level of another factor(s) [57]. The constant term and coefficients

$\beta_0$-$\beta_{123}$ can be estimated from the results of a designed experiment. Finally, $\epsilon$ represents the experimental error.

The coefficients for each term in the model can be estimated using a 2-level full-factorial designed experiment. The two levels of each factor could be high/low settings for a continuous factor, or on/off for a categorical or binary factor. To test all possible combinations of factors, a full factorial design can be used which, when there is 3 factors, has 8 runs as shown in Table 12, with a −1 indicating the factor is set to its low setting and +1 indicating the high setting [5].

*Table 12: Example three factor designed experiment run plan*

| Run | Factor Level | | |
|---|---|---|---|
| | $X_1$ | $X_2$ | $X_3$ |
| 1 | −1 | −1 | −1 |
| 2 | +1 | −1 | −1 |
| 3 | −1 | +1 | −1 |
| 4 | +1 | +1 | −1 |
| 5 | −1 | −1 | +1 |
| 6 | +1 | −1 | +1 |
| 7 | −1 | +1 | +1 |
| 8 | +1 | +1 | +1 |

The model is fit to the experimental data by calculating the coefficients for each of the model terms. Each coefficient ($\beta$) in the model can be estimated by calculating the effect of the model term on the response variable Y. Equation 16 illustrates how the coefficient for each factor is calculated using the average responses from runs that have the given factor at its high and low levels.

$$\beta_i = \frac{\text{Factor Effect}}{2} = \sum_{i=1}^{n} \frac{Y_i^1}{n} - \sum_{i=1}^{n} \frac{Y_i^0}{n} \tag{16}$$

where $Y_i^1$ represents the experimental response for a run which includes the factor in question at its high level, and $Y_i^0$ represents a run with the factor set to the low level. The effect of each factor is the difference in response between the high and low factor settings averaged over the $n$ runs for each setting. The calculated factor effects are then converted into the model coefficients by dividing by two as the model is centered around the overall measured average final reward which becomes the constant term in the model.

The accuracy of the estimations can be improved by performing multiple trials, or replicants, of each unique experiment run. The method of least squares regression is used to estimate the effect of factors based on all data gathered during the designed experiment [5], [57]. Analysis of variance in the experimental results can be completed to obtain confidence intervals for the effect of each factor as detailed in [5], [57], [58]. The analysis of variance can also be used to eliminate insignificant terms from the chosen mathematical model.

The example given for 3 factors has 8 possible unique combinations that can be tested; however, the number of runs needed to test all possible combinations increases exponentially with increasing number of factors. The present thesis explores 7 observations as factors which results in 128 unique possible combinations. To minimize the number of runs required, a fractional factorial experiment can be used. A fractional factorial designed experiment reduces the number of runs required by factors of 2, resulting in experiments with, for example, ½ or ¼ the number of runs of the full factorial experiment. The number of runs is reduced in such a way that orthogonality is maintained in the experiment. Orthogonality is important so that each factor has the same number of runs at its high and low levels to ensure no unnecessary bias is introduced to the results. Table 13 shows the required runs for a half fraction factorial experiment for the previous three factor example.

*Table 13: Example run plan for half fraction factorial designed experiment with three factors*

| Run | Factor Level | | |
|---|---|---|---|
| | $X_1$ | $X_2$ | $X_3$ |
| 1 | −1 | −1 | +1 |
| 2 | +1 | −1 | −1 |
| 3 | −1 | +1 | −1 |
| 4 | +1 | +1 | +1 |

The model coefficients can be calculated in the same way as for a full factorial experiment; however, the trade-off made with the fractional factorial designed experiment is that, with a reduced number of unique runs, one can no longer estimate all terms of the complete model independently. A half fraction factorial experiment can estimate half of the complete model terms from Equation 15. The model terms (main effects and interactions) remaining become aliased in pairs with those which have been removed. When two factors or interactions are aliased, this condition means one cannot distinguish between the effect of the two factors, and they are said to be confounded. The aliased (or confounded) pairs of factors for the half fraction factorial experiment detailed in Table 13 are listed in Table 14 (the constant term is represented by $I$) [57]. In this particular example, the $X_1$ factor is aliased with the $X_2X_3$ interaction term, so the estimation for the effect of $X_1$ that we can calculate from this experiment also includes the effect of $X_2X_3$. Typically, a fractional factorial designed experiment is setup so that the main effects and key interactions are aliased with higher-order interactions, as higher-order interactions are often assumed to be less significant.

*Table 14: Aliased or confounded pairs for the half fraction factorial designed experiment in Table 13*

| Aliased Pairs for Half Fraction Experiment with 3 Factors |
|---|
| $I \leftrightarrow X_1X_2X_3$ |
| $X_1 \leftrightarrow X_2X_3$ |
| $X_2 \leftrightarrow X_1X_3$ |
| $X_3 \leftrightarrow X_1X_2$ |

The assumptions which are made in order to proceed with the fractional factorial designed experiment are given in the *Engineering Statistics Handbook* [57]:

1. The measurement system used to record data is assumed to have sufficient accuracy and sensitivity to measure changes in the process response caused by changing the factors. Since the study will be carried out in a simulation environment, this assumption is valid.
2. The process is assumed to be stable and not to drift with time over the course of the study. Again, due to the controlled nature of the simulations in this research, this assumption is valid. The most significant random aspect of the simulations used in this work is the random initialization of the neural network weights and, to minimize the effect of this necessary randomness, each simulation is averaged over 10 independent runs.
3. An assumption must be made that the process is likely to be approximated well by the selected model. In this case a linear regression model is a good choice as each factor (observation) has only two discrete levels (on or off). There is no need to model any curvature between the two factor levels, as might be the case for continuous factors or those with more than two levels.
4. The classic assumptions for regression analysis are made in that residuals between the observed and predicted responses should be normally distributed and the magnitude of these errors should be independent of the time of measurement, the magnitude of the predicted response or the factors settings used to obtain the measurement. These

assumptions will be verified during the analysis of the designed experiment results in Chapter 5.

All work for this thesis related to the design of experiments was completed using the Minitab® Statistical Software package [59].

## 4.2. The Designed Experiment Setup

The set of potential observations listed in Table 15 below are as discussed in Chapter 3. This table also lists the number of individual signals that are associated with each observation. For example, if the joint torques are used as observations, then the torque is measured at all 18 joints. Each of the seven possible observations used in the designed experiment is assigned an indicator letter for convenience as shown in Table 15.

*Table 15: Summary of the designed experiment factors*

| Indicator Letter | Observation | Number of Measurement Signals Associated with the Observation |
|---|---|---|
| A | Joint torques | 18 |
| B | Body velocities | 3 |
| C | Body tilt | 4 |
| D | Body angular velocities | 3 |
| E | Body height | 1 |
| F | Ground contacts binary | 6 |
| G | Ground contact forces | 12 |

The seven possible observations each have the binary options of being on or off (used for learning or not); therefore, they are two-level factors. This factorial design leads to 128 possible

unique combinations from which an optimal set of observations for the given locomotion task could be selected. In addition, due to the inherently-random nature of reinforcement learning, multiple repetitions of each trial are needed to confidently distinguish between the performance of different combinations of observations. For this work 10 replicants were selected to be confident that there is a statistical significance when comparing learning results. This test plan leads to running 1,280 separate reinforcement learning routines to generate the data required to compare all possible combinations of the potential observations. Given that a single training routine takes roughly 10 hours, there is a need for a method which can provide information about the usefulness of the observation that uses less trials.

A fractional factorial designed experiment was, therefore, applied to explore its potential use as a guiding tool in the selection of observations/measurements for a hexapod locomotion problem. A quarter fraction factorial designed experiment was selected which consists of 32 separate cases to be run – reduced significantly from the 128 required for the full factorial designed experiment. With 32 cases and 10 replicants per case, a total of 320 separate reinforcement learning routines were needed. The quarter fraction factorial design was set up using the Minitab statistical software [59]. Table 16 shows the design table for the experiments generated using Minitab, where each case/run will be repeated for 10 replicants in order to account for the random nature of reinforcement learning. A plus sign indicates that the observation will be used for the specific run, while a negative sign shows when the observation is turned off (the observations are labelled using the indicating letters from Table 15).

*Table 16: Quarter fraction factorial designed experiment run test plan (x10 replicants)*

| Run | Observations | | | | | | |
|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G |
| 1 | - | - | - | - | - | + | + |
| 2 | + | - | - | - | - | - | - |
| 3 | - | + | - | - | - | - | - |
| 4 | + | + | - | - | - | + | + |
| 5 | - | - | + | - | - | - | + |
| 6 | + | - | + | - | - | + | - |
| 7 | - | + | + | - | - | + | - |
| 8 | + | + | + | - | - | - | + |
| 9 | - | - | - | + | - | - | - |
| 10 | + | - | - | + | - | + | + |
| 11 | - | + | - | + | - | + | + |
| 12 | + | + | - | + | - | - | - |
| 13 | - | - | + | + | - | + | - |
| 14 | + | - | + | + | - | - | + |
| 15 | - | + | + | + | - | - | + |
| 16 | + | + | + | + | - | + | - |
| 17 | - | - | - | - | + | + | - |
| 18 | + | - | - | - | + | - | + |
| 19 | - | + | - | - | + | - | + |
| 20 | + | + | - | - | + | + | - |
| 21 | - | - | + | - | + | - | - |
| 22 | + | - | + | - | + | + | + |
| 23 | - | + | + | - | + | + | + |
| 24 | + | + | + | - | + | - | - |
| 25 | - | - | - | + | + | - | + |
| 26 | + | - | - | + | + | + | - |
| 27 | - | + | - | + | + | + | - |
| 28 | + | + | - | + | + | - | + |
| 29 | - | - | + | + | + | + | + |
| 30 | + | - | + | + | + | - | - |
| 31 | - | + | + | + | + | - | - |
| 32 | + | + | + | + | + | + | + |

The quarter fraction factorial design is of a resolution that allows for the main effects of each factor (observation) to be distinguish from one another and from any 2-way interactions. A sufficient resolution level is important as the goal of the designed experiment is to gain insight into the effect of the different observations (factors) and be able to objectively compare their

relative effects on performance. The alias structure for the quarter fraction factorial design can be found in Appendix A. The main effects are confounded with 3-way interactions and 2-way interactions could be confounded with one another; however, it is expected that these interactions will be minimal. If there is a significant interaction, additional runs could be added to further explore these interactions.

The metric used to evaluate the performance of the reinforcement learning agent using different observations is the final average reward after training for 1000 episodes. The desired performance of the hexapod is accurate and efficient tracking of the goal trajectory line and, since the reward function was tuned to encourage this behaviour, it should be a good metric of success. The only changes to the hexapod simulation throughout this study is the set of observations used (according to Table 16) which, in turn, affects the number of nodes in the input layer of both the actor and critic networks. All other aspects of the simulation remain fixed.

The quarter fraction designed experiment was carried out over a span of 6 weeks with each run simulated 10 times for a total of 320 complete training routines. The final average reward was recorded for each trial along with other potentially-useful data, and the results of the study are analyzed in the subsequent chapter.

# Chapter 5: Results and Analysis

This chapter presents and analyzes the results of the designed experiment. Using the gathered data, a regression model is formulated, then used to predict which combination of observations yields optimal reward during training. The model predictions and corresponding trained agent are then validated within the developed hexapod simulator.

## 5.1. Sample Learning Results

First consider the case of a single training routine and its resulting saved reinforcement learning agent. Figure 25 shows the resulting learning curve over 1000 episodes for a sample successful training routine taken from the designed experiment (run 5 replicant number 3). The blue curve is the final reward value for each episode. The black curve shows the running average final reward over the last 250 episodes as in [48]. Due to the random initialization in starting position of the robot for each episode, it can be seen in the figure that the final reward per episode varies significantly from the general learning trend. By taking the running average over a larger number of episodes, the goal is to produce a smooth curve and reduce the effect of the stochastic nature of the training routine. The final average reward is the value used to compare the performance of different trials.

As can be seen in Figure 25, the learning curve shows an initial dip in performance, before it reaches a "point of discovery" and begins to steadily climb at around 100 episodes. At the beginning of each training routine, the reinforcement learning agent has randomly-initialized weights in its neural (actor) network, so the resulting random actions tend to offset one another and result in the hexapod closely following the underlying central pattern generator base gait. As the agent begins to explore new actions the hexapod's performance decreases, until the agent discovers more rewarding actions at which point the hexapod's episode-by-episode performance increases significantly and the average reward climbs steadily. The average reward levels off at around 800 episodes when the reinforcement learning agent has extracted the maximum performance from the given action space and learning conditions. Note that the individual

episode rewards are largely negative because there are more negative penalty terms than positive rewards in the reward function. Since the reward function is fixed throughout the designed experiment, the resulting learning curves throughout the designed experiment can be directly compared.

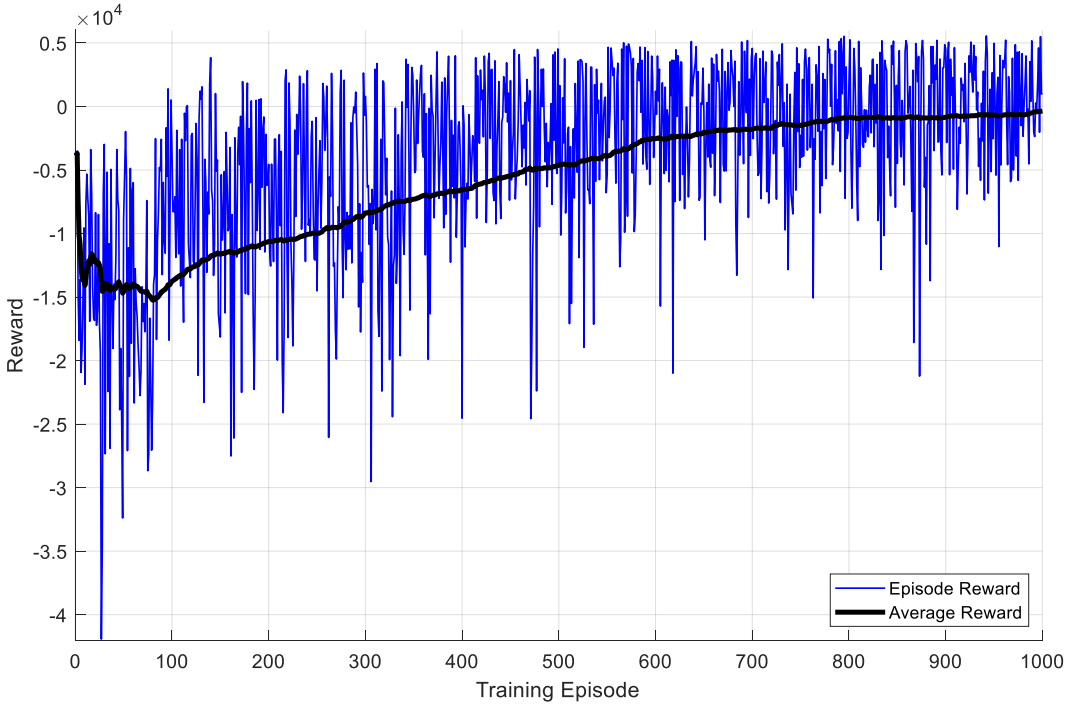



*Figure 25: Sample learning curve for successful training episode*

Once learning is complete, the final agent saved after the 1000$^{th}$ episode can be tested to help understand how the final average reward correlates to the actual walking performance of the hexapod. The final agent is tested by starting the hexapod at 7 different offsets from the goal trajectory line and allowing the simulation to run for 30 seconds resulting in hexapod paths as shown in Figure 26. Note that the two outermost positions at ± 1.5 m are outside of the (± 1 m) range of starting offsets used during training to test the robustness of the learned behaviour and the ability of the RL agent to adapt to previously-unseen scenarios. The path of the center of the hexapod body is illustrated in the figure and shows how the agent has learned to direct the hexapod toward and then follow along the goal trajectory line. In this particular case the hexapod is able to converge to the goal trajectory line from all starting positions after roughly 1.5 m of

forward travel. Figure 26 also shows that there is a small offset remaining for two of the seven test cases once they have reached a steady state where the robot travels parallel to the goal trajectory line.

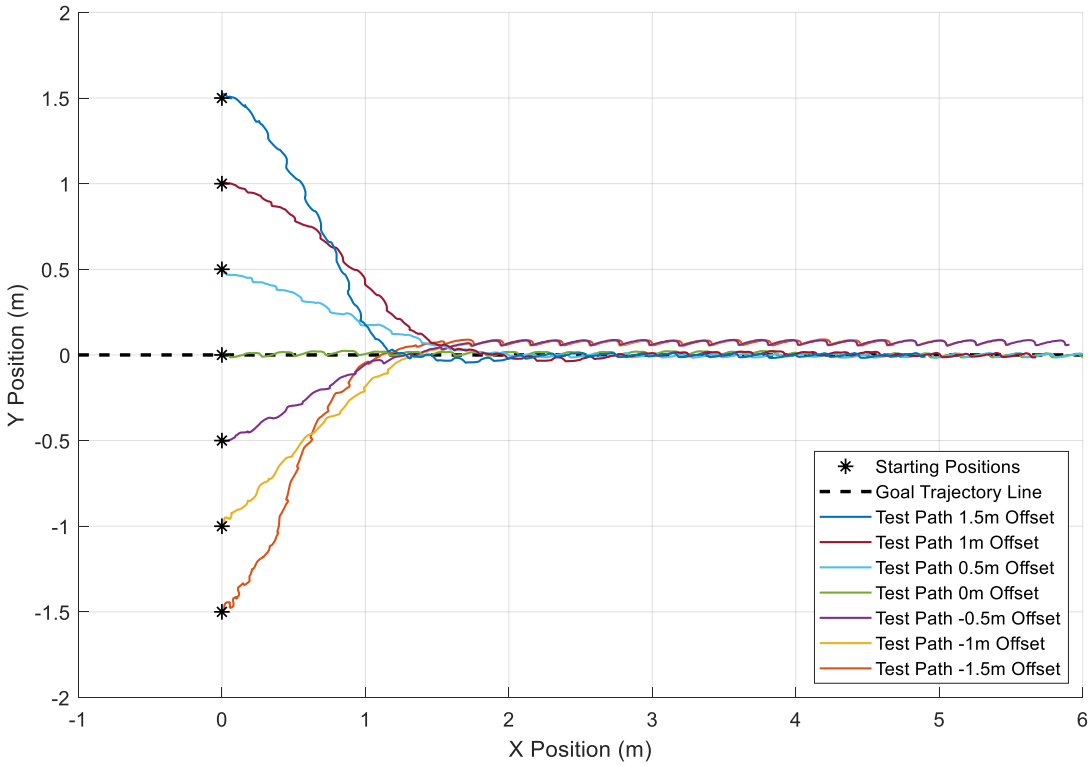



*Figure 26: Example test paths for successfully trained agent*

Figure 27 shows another sample result taken from run 4 replicant 1 for comparison purposes. This figure shows a comparatively less successful learning routine where the final reward reached is lower, and thus the learned behaviour is less desirable. The slope of the average reward curve is shallower and has no obvious "point of discovery" (where the reward function begins to increase more significantly) as in the previous case. This particular case also illustrates the exploration of different behaviours during training between episodes 600 to 700 where the episode reward curve takes a significant dip before recovering and rising further. This location indicates a situation where the hexapod attempted a new behaviour (initiated by the random exploration factor) which resulted in a decrease in reward so the new unsuccessful

behaviour was eventually discarded. Overall it is clear that, for the set of observations used in this run, the agent has more difficulty learning an optimized behaviour.

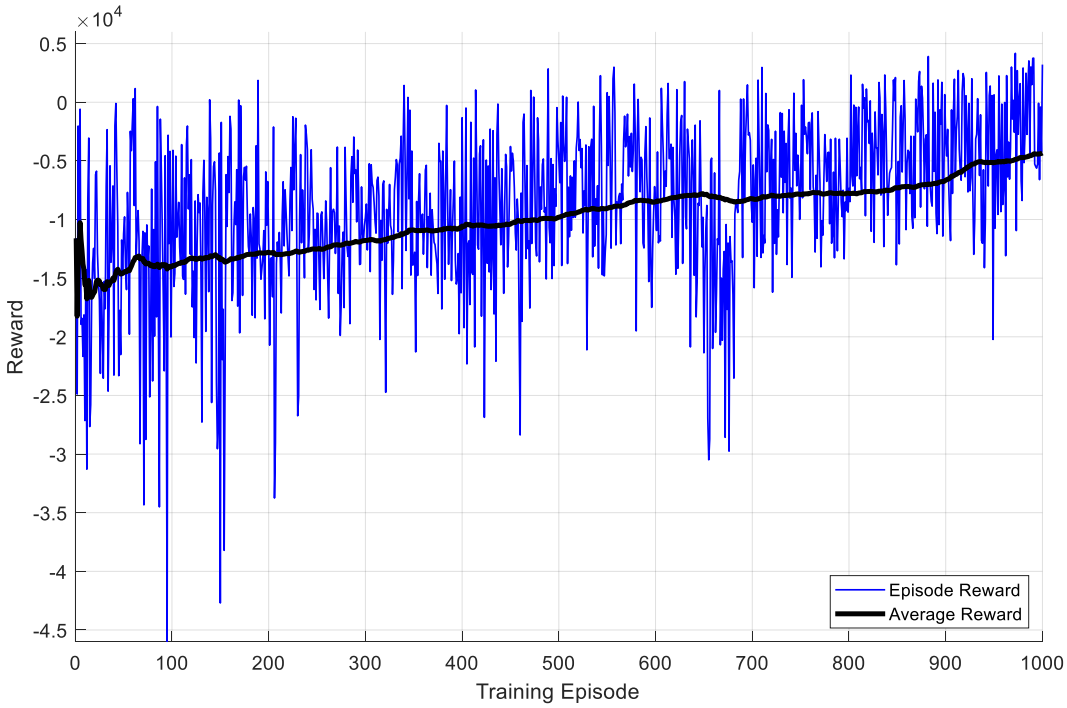



*Figure 27: Learning curve for a typical less successful case*

The learned behaviour for this less-successful case can again be illustrated by plotting a series of test paths, to show what this lesser final reward value corresponds to in terms of hexapod performance. Figure 28 shows the resulting hexapod paths when tested at the seven different starting offsets. The paths represent 30 seconds of movement of the hexapod body center. It is clear to see why this particular agent does not achieve the same final reward as the previous example. When the hexapod is started with zero offset it is able to follow along the goal trajectory line; however, for all other starting positions the resulting path demonstrates an oscillatory behaviour mimicking an underdamped system response. The agent has learned to direct the hexapod toward the goal trajectory but not to straighten out its path in time to prevent overshooting multiple times. By looking at the final positions of the hexapod after 30 seconds have elapsed (the end of the path lines), one can see that this particular learned gait is also slower than the previous example as it is not able to travel as far in the positive direction (to the right in

the figure). The reduced walking speed and inability to stay on the goal trajectory line account for the lesser final reward value achieved by this trial run.

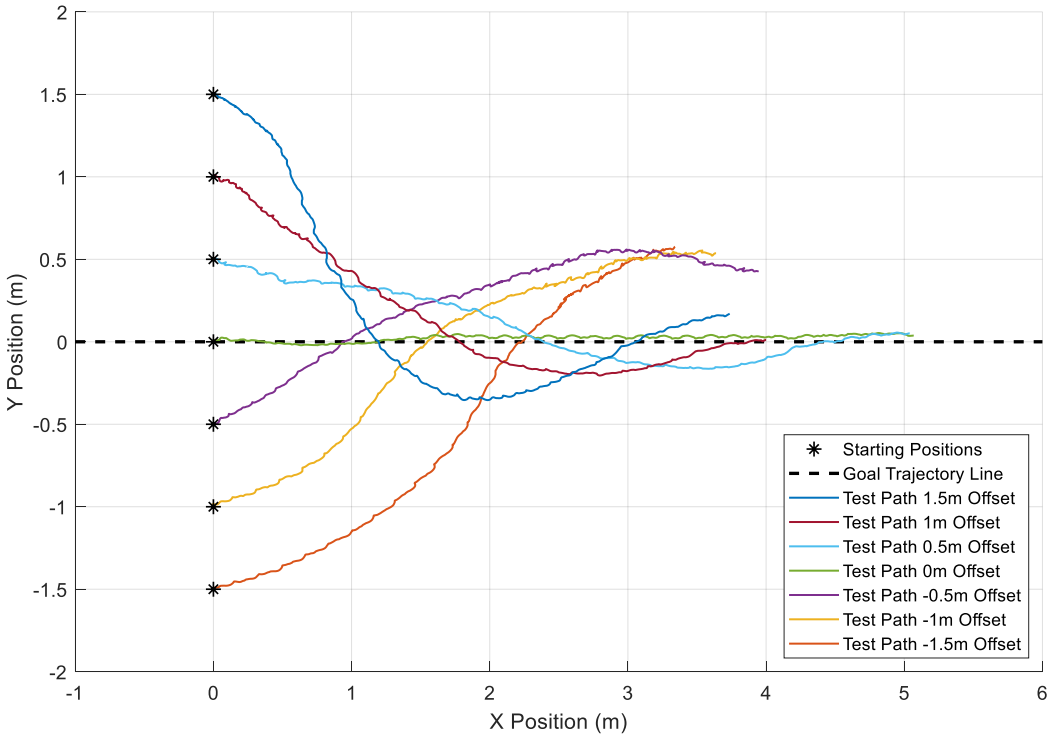



*Figure 28: Test paths for the less successful case*

There are also cases where the combination of observations used in the run does not allow the reinforcement learning agent to gain sufficient information about the hexapod control system and simulation environment. For these cases, the hexapod will typically either walk in a random direction or become trapped in a circular path, with no success at forward progress along the goal trajectory.

Each of the ten replicants for a given run (corresponding to a particular combination of observations) can be plotted together on a single graph to visualize the repeatability of the learning process. Figure 29 shows the ten replicants for run 5 as an example. The moving average reward curves are plotted in black and an overall average reward curve is calculated and shown in red. Figure 29 shows that, for run 5, eight out of the ten trials (replicants) follow a very

similar learning curve, while two of the replicants failed to learn. The learning routine includes several randomized processes such as the initialization of neural network weights and biases, as well as the noise artificially introduced for exploration of the possible action space. The stochastic nature of reinforcement learning means that the results of a single trial cannot be fully repeatable.

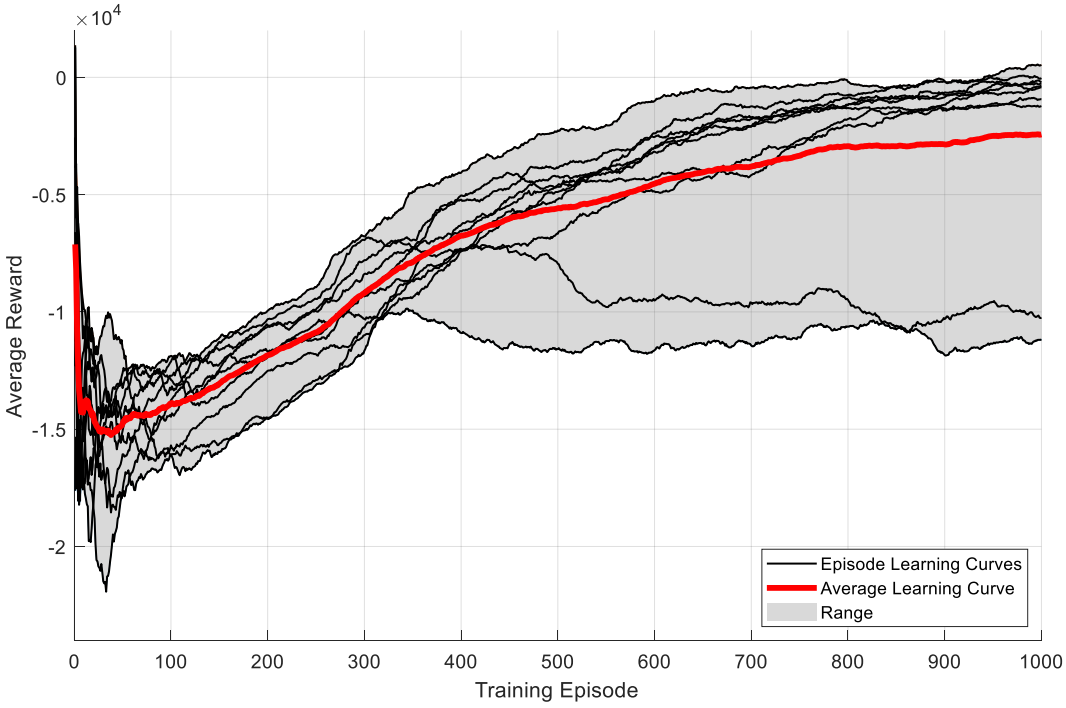



*Figure 29: Learning curves for all 10 replicants of run 5*

Again, one can examine the hexapod paths for seven different starting locations to show the correlation between final reward and actual behaviour. Figure 30 shows all paths taken by the hexapod in 30 seconds when each of the ten replicants for run 5 are tested at the seven staring locations. This plot clearly shows that the two outlier runs with significantly lower final rewards have not learned the desired behaviour. The remaining eight replicants all show successful learning and manage to track towards and follows the desired trajectory.

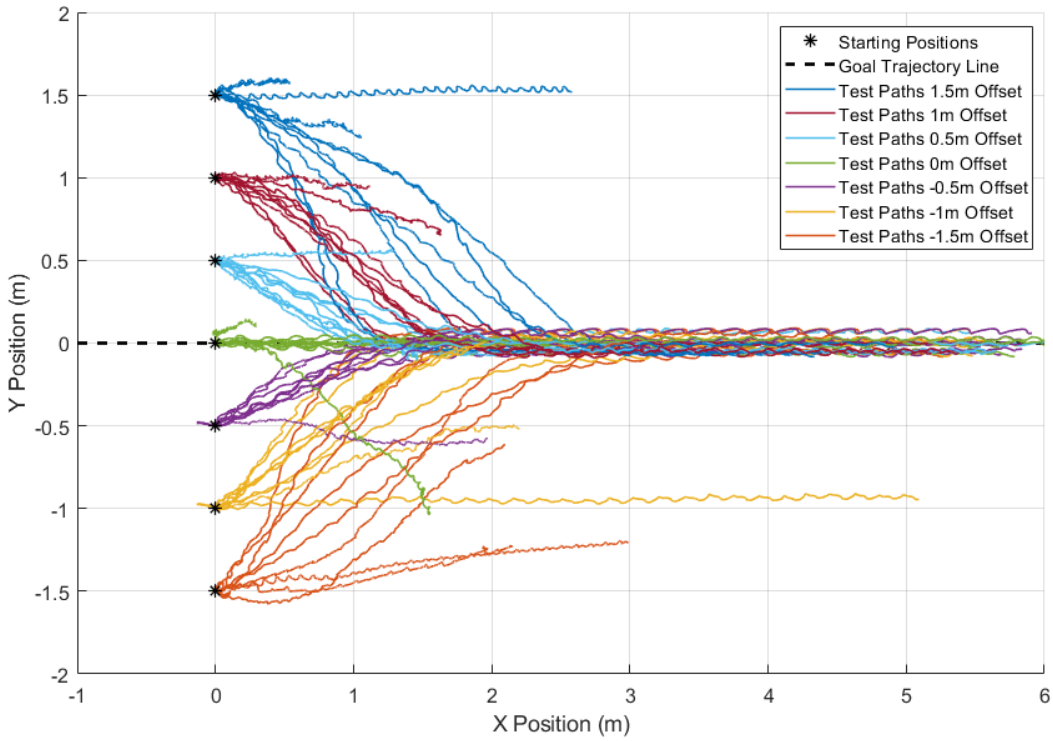



*Figure 30: Test paths for all 10 replicants of run 5*

The complete set of results including plots for all 32 runs of the designed experiment are found in Appendix B. The set of observations used clearly influences not only the final overall average reward reached, but also the level of repeatability of the learning process – both of which are important for deploying reinforcement learning on an actual hexapod robot. There are many other factors which also affect the repeatability and performance of the learning routine including, but not limited to, the terms in the reward function, the neural network sizes, the learning routine hyperparameters such as the noise, and many more. It would be a significant undertaking to explore all combinations of every parameter in an attempt to optimize the reinforcement learning process. For the purposes of this work the main focus for optimization is the observations, while all other parameters are tuned by hand to achieve a satisfactory level of performance and held constant throughout the designed experiment. Future research could examine how optimization of the other parameters influences the results obtained in the designed experiment.

## 5.2. Designed Experiment Results

The results of all 320 trials (10 replicants of 32 different runs) are compiled in Table 17 which lists the mean average final reward obtained across all 10 replicants of each run. The time spent running all trials for the quarter factorial designed experiment on a single PC was about 6 weeks – a substantial amount of time which would have been 4 times longer if a full factorial experiment was run. Comparing the different rows of Table 17 to the last row, it can be seen that using the maximum number of observations does not achieve the highest learning performance. Run 32 (which contains all seven observations) yielded an average final reward over 10 replicants of −1776.19 which is lower than Runs 6, 8, 14, 16, 21, 22, 28, 30, and 31 (which contain different combinations of fewer than seven observations).

*Table 17: Designed experiment summary of results, runs with rewards in bold font have average final rewards that are higher than Run 32 (which contains all seven observations)*

| Run | Observations | | | | | | | Average Final Reward over 10 Replicants |
|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | |
| 1 | - | - | - | - | - | + | + | -8048 |
| 2 | + | - | - | - | - | - | - | -7384 |
| 3 | - | + | - | - | - | - | - | -3500 |
| 4 | + | + | - | - | - | + | + | -6764 |
| 5 | - | - | + | - | - | - | + | -2433 |
| 6 | + | - | + | - | - | + | - | **-1752** |
| 7 | - | + | + | - | - | + | - | -2489 |
| 8 | + | + | + | - | - | - | + | **-1392** |
| 9 | - | - | - | + | - | - | - | -6198 |
| 10 | + | - | - | + | - | + | + | -10328 |
| 11 | - | + | - | + | - | + | + | -4704 |
| 12 | + | + | - | + | - | - | - | -5369 |
| 13 | - | - | + | + | - | + | - | -2226 |
| 14 | + | - | + | + | - | - | + | **-1265** |
| 15 | - | + | + | + | - | - | + | -2002 |
| 16 | + | + | + | + | - | + | - | **-1692** |
| 17 | - | - | - | - | + | + | - | -7664 |
| 18 | + | - | - | - | + | - | + | -5956 |
| 19 | - | + | - | - | + | - | + | -6855 |
| 20 | + | + | - | - | + | + | - | -1957 |

| Run | Observations | | | | | | | Average Final Reward over 10 Replicants |
|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | |
| 21 | - | - | + | - | + | - | - | **-630** |
| 22 | + | - | + | - | + | + | + | **-773** |
| 23 | - | + | + | - | + | + | + | -2364 |
| 24 | + | + | + | - | + | - | - | -2014 |
| 25 | - | - | - | + | + | - | + | -7748 |
| 26 | + | - | - | + | + | + | - | -4675 |
| 27 | - | + | - | + | + | + | - | -4982 |
| 28 | + | + | - | + | + | - | + | **-1224** |
| 29 | - | - | + | + | + | + | + | -2658 |
| 30 | + | - | + | + | + | - | - | **-1414** |
| 31 | - | + | + | + | + | - | - | **-579** |
| 32 | + | + | + | + | + | + | + | -1776 |

Figure 31 illustrates a histogram of the final average reward for all 320 learning routines. This histogram provides some insight into the challenges faced by the DDPG learning agent to learn the actions required to control the hexapod so that it follows the desired trajectory. The histogram illustrates the spread of final rewards achieved over the course of the designed experiment, and the extent of outliers compared to the most commonly-obtained results. As the final average reward is a direct measure of hexapod performance, this histogram demonstrates the rate of success of learning across all tested combinations of observations. Figure 31 shows that a large portion of the trials resulted in a final reward between −1000 and 0. There are many outliers on the lower end of the spectrum, resulting in a negative skew in the distribution of the results – as can be seen by the long tail on the left side of the distribution. These are trials where the hexapod has not been able to learn to walk towards the goal trajectory line, or perhaps not even been able to learn to walk at all. Above a final average reward of approximately −4000, the hexapod gait starts to become much more successful – a subjective transition between failing to learn and approaching the desired behaviour. The shape of the histogram plot seems to indicate that there is a critical factor (number or potential inclusion of a certain combination) of observations for which the performance of the hexapod reaches a satisfactory level, as can be seen by the large peak centered around the –500 to 0 bin. Figure 31 demonstrates that the combination of using a central pattern generator in tandem with a DDPG agent is successful over a wide range of the different combinations of observations used in the trials. There appears to be

limited potential performance gains above this peak (around zero final reward) with the maximum reward reached throughout all trials of the designed experiment not exceeding 2000. The upper limit to the final average reward is limited by the frequency and maximum amplitudes of the central pattern generator which, in turn, are determined by the maximum possible speed of the hexapod servo motors. Optimization of the hexapod gait using reinforcement learning should allow the hexapod to approach its maximum potential speed (and, therefore, reward) as set by the terms of the Hopf oscillator.

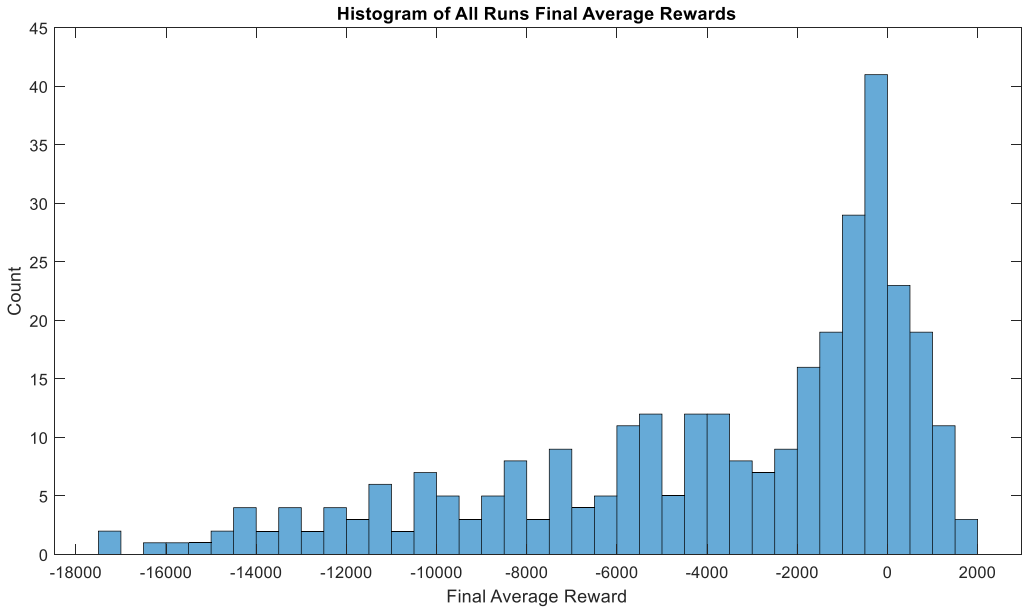



*Figure 31: Histogram of all final average rewards obtained during the designed experiment*

To begin the analysis of the results of the designed experiment, one can examine the overall average effect of each factor of the designed experiment (observation) on the final average reward across all trials. Figure 32 illustrates the mean effect of each observation in a side-by-side comparison. The overall mean effect of each observation is determined by taking the mean final average reward of all trials where the observation was excluded, and directly comparing this reward to the mean of final average rewards for all trials for which the observation was present. The error bars indicate the standard error of the mean final reward estimates based on the 10 replicant samples. The horizontal x-axis in Figure 32 indicates the absence or presence of each observation as indicated with a 0 or 1, respectively, and the vertical

y-axis shows the corresponding mean values of the final average reward. The observations are listed using the indicator letters introduced in Table 15.

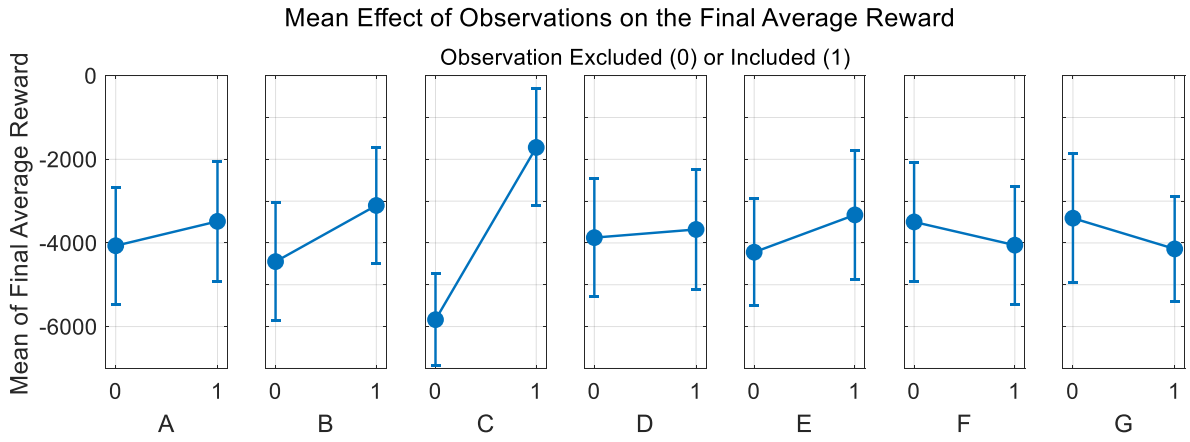



*Figure 32: Mean effect of the seven main factors (observations) on the final average reward*

From Figure 32 one can see that observation C, the body tilt, has the greatest positive effect on the final average reward – significantly more than the other six observations. The body tilt is the only observation whose error bars do not overlap, so we can be certain that there is statistical significance in its effect. The remaining observations appear to have varying effects on the mean final average reward, but further analysis is needed to investigate their significance. The inclusion of observations A (joint torques), B (body velocities) and E (body height) have a mean positive impact on the hexapod performance. The body angular velocities (observation D) do not seem to be as critical to the final reward obtained since the mean values do not differ significantly when these measurements are included or excluded. Interestingly, both of the ground contact observations F (binary contact) and G (contact forces) appear to have an overall negative effect on the mean final average reward. While Figure 32 provides a good indication of which observations to choose to maximize performance, it does not take into account any interactions between the factors. Interactions may occur because many of the observations provide information to the reinforcement learning agent or can be related through the kinematics of the hexapod motion. For example, although Figure 32 would indicate that the inclusion of the ground contact F has a negative effect on reward, an interaction may be present that warrants its inclusion. For example, if the body tilt C has an interaction with the ground contact F, then the

inclusion of ground contact could increase the reward benefit obtained over just including just the body tilt. Such an interaction could be explained physically due to the connected nature of the hexapod ground contact states and body tilt during the walking gait; therefore, Figure 32 cannot directly be used to determine the best possible combination of observations since it does not consider any interactions.

To fully explore the relationship between different observations and the final average reward, a regression model is fit to the experimental data. The model is evaluated according to the following R-squared statistics.

### 5.2.1. R-Squared Statistics Used to Evaluate Model Fit

The analysis of the designed experiment was carried out using the Minitab statistical software package [59]. The following equations detail the various R-squared statistics that are used to quantify and improve the fit of the model to the data from the designed experiment analysis.

The R-squared value quantifies the fit of the determined model and produces a value between 0 and 1 which indicates the percentage of variability in the data which is accounted for by the model. An R-squared value of 100 % indicates that the model accounts for and explains 100 % of the variability in the dataset. The R-squared statistic is calculated as follows [60]:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{17}$$

where $y_i$ represents the final average reward obtained for the $i^{\text{th}}$ test run during the designed experiment, while $\hat{y}_i$ is the final average reward predicted by the model for the given trial $i$. Finally, $\bar{y}$ is the mean of all final average rewards in the designed experiment dataset. The numerator in Equation 17 is the sum of the squared errors (or residuals) between the actual final average rewards and those predicted by the model for the given combination of observations. The sum of squared errors is a measure of the variation in the data which is not explained by the model. The denominator is a measure of the total variation in the final average rewards obtained in the designed experiment which is used to normalize the results.

The R-squared statistic does not account for model size/complexity in quantifying the performance of the model [61]. The R-squared value will continue to increase with the addition of terms to the model even if they provide relatively little improvement to the model fit. An improvement to the R-squared statistic is the R-squared adjusted value [62] which considers the number of variables used in the model. The adjusted R-squared statistic encourages a simple model, and its value is calculated according to the following formula:

$$R_{adj}^2 = 1 - \frac{(1 - R^2)(n - 1)}{(n - k - 1)} \tag{18}$$

where $R^2$ is the R-squared statistic as calculated in Equation 17, $n$ is the number of datapoints in the experiment dataset (in this specific case 320 total trials), and $k$ represents the number of independent variables (coefficients) in the regression model.

The final statistical value that will be used to evaluate model's fit is the predicted R-squared value $R_{pred}^2$ as shown in Equation 19 [62]. The predicted R-squared statistic is calculated by systematically removing a single value from the experimental dataset, fitting the desired model to the remaining data, and then determining how well the model predicts the missing datapoint. This procedure is repeated for all values of the dataset and then the predicted R-squared is calculated similarly to the standard R-squared value. The only difference between Equations 17 and 19 is that, for the predicted R-squared value, the residuals in the sum of squared errors (numerator) use a different model for each residual as opposed to a single model for the entire dataset.

$$R_{pred}^2 = 1 - \frac{\sum(y_i - \hat{y}_i^*)^2}{\sum(y_i - \bar{y})^2} \tag{19}$$

In Equation 19, as in the case of the standard R-squared value, $y_i$ is the final average reward from the $i^{\text{th}}$ experiment and $\bar{y}$ is the mean of all final average rewards obtained during the designed experiment. $\hat{y}_i^*$ represent the predicted final average reward for the $i^{\text{th}}$ experiment, obtained using a model fit to the dataset with the result from the experiment run to be predicted removed. The predicted R-squared value is an important statistic to compare the fit of different models as it helps to detect overfitting of the model [62]. The predicted R-squared value will

always be less than the standard R-squared because of the removal of each data point during calculation, and the random noise in each experimental data point is by definition impossible to predict. In contrast the standard R-squared value has all of the data available to fit at once − there are no unknowns leading to a higher R-squared value. A large difference between the predicted R-squared value the standard R-squared indicates that the model is overfitting the experimental dataset.

## 5.3. Designed Experiment Analysis

The following section details the step-by-step procedure that was taken (using the Minitab statistical software) to produce and refine the final model fit to the data generated using the design experiment. The model is then used to make a prediction about the optimal combination of observations required to maximize the final average reward.

Equation 20 shows the complete linear regression model which was fitted to the experimental data.

$$Y_{pred} = \beta_0 X_0 + \beta_1 X_1 + \beta_3 X_3 + \cdots + \beta_{31} X_{31} \tag{20}$$

where $Y_{pred}$ is the final average reward predicted by the model, $X_i$ and $\beta_i$ are the model terms and associated coefficients, respectively, as listed in Table 18. The model contains all terms which may be separately calculated from the designed experiment. The designed experiment is a quarter fraction factorial experiment so not all possible combinations of observations have been run, which means that not all higher-order interactions can be estimated from the data. The higher-order interactions that are confounded with the model terms are listed in Appendix A.

*Table 18: List of all terms in complete linear regression model from the designed experiment*

| Label | Term | Term Type | Coefficient |
|---|---|---|---|
| $X_0$ | 1 | constant | $\beta_0$ |
| $X_1$ | A | Main effect | $\beta_1$ |
| $X_2$ | B | Main effect | $\beta_2$ |
| $X_3$ | C | Main effect | $\beta_3$ |
| $X_4$ | D | Main effect | $\beta_4$ |
| $X_5$ | E | Main effect | $\beta_5$ |
| $X_6$ | F | Main effect | $\beta_6$ |
| $X_7$ | G | Main effect | $\beta_7$ |
| $X_8$ | AB | First-order interaction | $\beta_8$ |
| $X_9$ | AC | First-order interaction | $\beta_9$ |
| $X_{10}$ | AD | First-order interaction | $\beta_{10}$ |
| $X_{11}$ | AE | First-order interaction | $\beta_{11}$ |
| $X_{12}$ | AF | First-order interaction | $\beta_{12}$ |
| $X_{13}$ | AG | First-order interaction | $\beta_{13}$ |
| $X_{14}$ | BC | First-order interaction | $\beta_{14}$ |
| $X_{15}$ | BD | First-order interaction | $\beta_{15}$ |
| $X_{16}$ | BE | First-order interaction | $\beta_{16}$ |
| $X_{17}$ | BF | First-order interaction | $\beta_{17}$ |
| $X_{18}$ | BG | First-order interaction | $\beta_{18}$ |
| $X_{19}$ | CD | First-order interaction | $\beta_{19}$ |
| $X_{20}$ | CE | First-order interaction | $\beta_{20}$ |
| $X_{21}$ | CF | First-order interaction | $\beta_{21}$ |
| $X_{22}$ | CG | First-order interaction | $\beta_{22}$ |
| $X_{23}$ | DE | First-order interaction | $\beta_{23}$ |
| $X_{24}$ | DF | First-order interaction | $\beta_{24}$ |
| $X_{25}$ | DG | First-order interaction | $\beta_{25}$ |
| $X_{26}$ | ACE | Second-order interaction | $\beta_{26}$ |
| $X_{27}$ | ACG | Second-order interaction | $\beta_{27}$ |
| $X_{28}$ | BCE | Second-order interaction | $\beta_{28}$ |
| $X_{29}$ | BCG | Second-order interaction | $\beta_{29}$ |
| $X_{30}$ | CDE | Second-order interaction | $\beta_{30}$ |
| $X_{31}$ | CDG | Second-order interaction | $\beta_{31}$ |

The model is fit to the experimental data by calculating the coefficients for each of the model terms. Coefficients are determined by calculating the effect of the model term on the response variable (final average reward). Equation 21 illustrates how the effect of each term

(Term Effect) is calculated using the average responses from runs that included and excluded the observations described in the model term [57].

$$\text{Term Effect} = \sum_{i=1}^{n} \frac{Y_i^1}{n} - \sum_{i=1}^{n} \frac{Y_i^0}{n} \tag{21}$$

In this equation, $Y_i^1$ represents the experimental response for a run which includes the model term in question, and $Y_i^0$ represents a run without the model term. The fractional factorial used in this study is orthogonal, meaning it is designed in such a way that there are the same number of runs with each model term included and excluded; hence, for this study $n = 16$. The calculated Term Effects are then converted into the model coefficients by dividing by two as the model is centered around the overall measured average final reward which becomes the constant term in the model.

The analysis was carried out using Minitab to produce the following regression equation model. Model reduction was not completed at this stage so this equation contains the maximum 32 terms which could be estimated based on the 32 runs of the designed experiment.

$$\begin{aligned} Y_{pred} = -3775 &+ 292\,A + 671\,B + 2059\,C + 98\,D + 446\,E - 278\,F - 368\,G + 38\,AB - 86\,AC \\ &- 82\,AD + 564\,AE + 47\,AF + 166\,AG - 744\,BC + 215\,BD - 61\,BE + 41\,BF \\ &+ 86\,BG - 83\,CD - 256\,CE + 28\,CF + 251\,CG + 99\,DE - 175\,DF + 82\,DG \\ &- 738\,ACE + 159\,ACG - 24\,BCE - 65\,BCG - 195\,CDE - 189\,CDG \end{aligned} \tag{22}$$

In this equation, letters A through G are the observation indicating letters, and correspond to the presence (+1) or absence (−1) of the corresponding observation.

The fit of this initial model is described by the three different R-squared values described previously. The R-squared value is only 35 %, the adjusted R-squared is 28 %, and the predicted R-squared is even lower at 20 %. Based on these metrics, the initial model is not a good fit to the designed experiment results.

To investigate why the initial model is a poor fit, the residuals can be plotted. The residuals are determined by taking the difference between the final average reward predicted by

the model with each of the sets of observations tested during the designed experiment, and the actual values measured during the experiment. A well-fitting model that is able to describe all trends of the experimental data should result in residuals that consist only of the experimental error and noise of the system [60]. The residuals should display the characteristics of a random normal distribution and there should be no discernible trends in the spread of residuals [57].

Figure 33 shows the residuals for all 320 trials plotted as a function of the run number. The residuals are unitless and relate to the value of the final average reward. The residuals should be completely independent of the run number, as is confirmed visually by Figure 33. The desired distribution of the residuals in the vertical y-direction is for them to be symmetrical about the horizontal x-axis to create a normal distribution centered on a residual of zero. In Figure 33, it can be seen that the distribution of the residuals is skewed in the negative direction by a significant number of large negative outliers. These outliers are the most likely cause for the poor model fit.

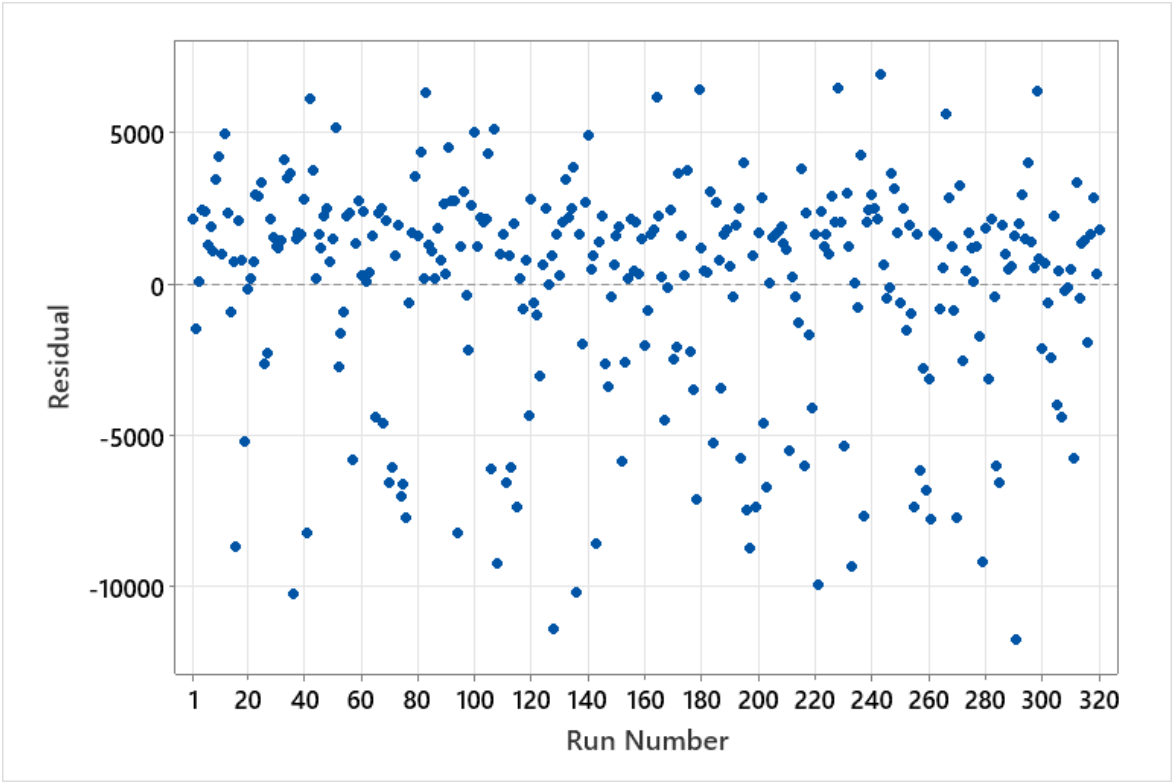



*Figure 33: Residuals plotted against run number for the initial model fit*

The model residuals can also be illustrated using a histogram plot as shown in Figure 34. The residuals are grouped into 20 bins of equal width over the total range, with the frequency in each bin shown on the vertical y-axis. Figure 34 clearly shows the long left-sided tail of the residual distribution and, by consequence, how the peak of the distribution is shifted to the right of zero. The outliers need to be eliminated in order to obtain the desired normal distribution of residuals.

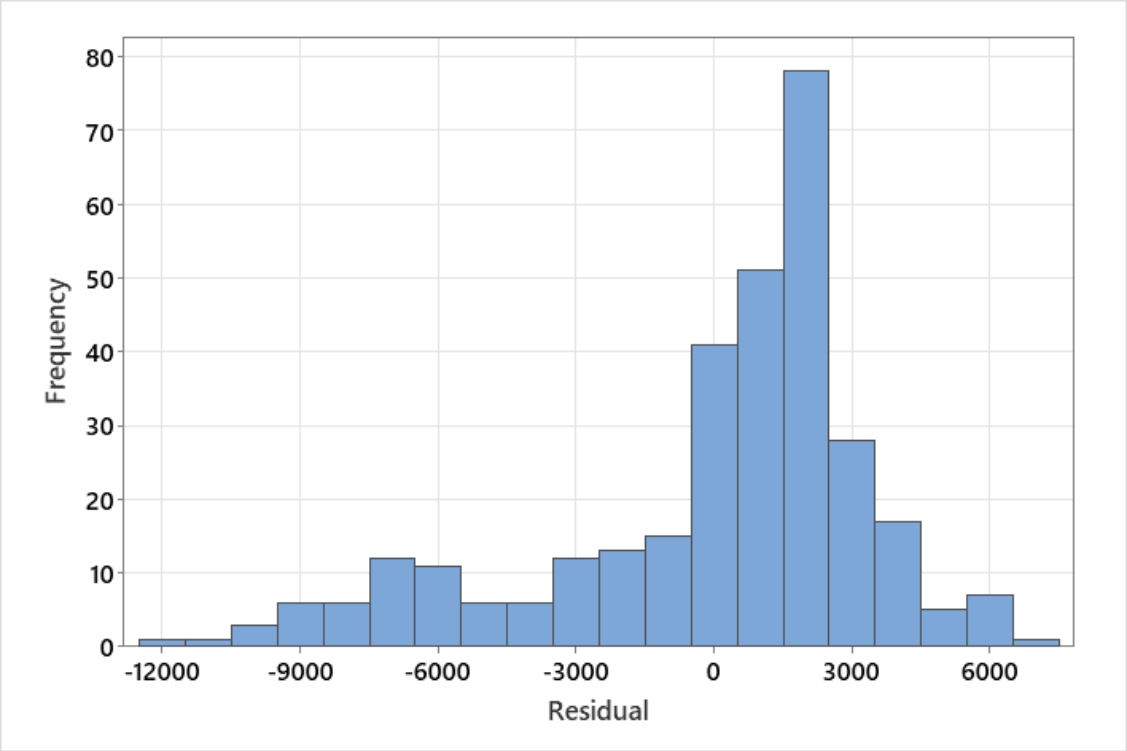



*Figure 34: Histogram of residuals for the initial model fit*

Closer examination of the compiled learning curves (see Appendix B) from the replicants of each of the 32 different runs reveals the source of the negative skew in the residual distribution. There appears to be a general trend that the runs lack repeatability within the ten replicates. Take run 23 as an example. Figure 35 shows the learning curves for each of the replicates for run 23 (black lines) as well as the average learning curve across all ten replicates (red line). The x-axis is the training episode and the y-axis shows the current average reward at the given episode. Run 23 illustrates a generally successful combination of observations as the hexapod is able to learn the desired trajectory-following behaviour with some level of

repeatability. The majority of the learning curves in Figure 35 end in a tight grouping around zero final average reward; however, there are three replicates which are outliers that resulted in little learning success and bring the average learning curve down significantly. Similar outliers can be seen throughout all runs of the designed experiment, regardless of how successful the majority of replicates are for the given run.

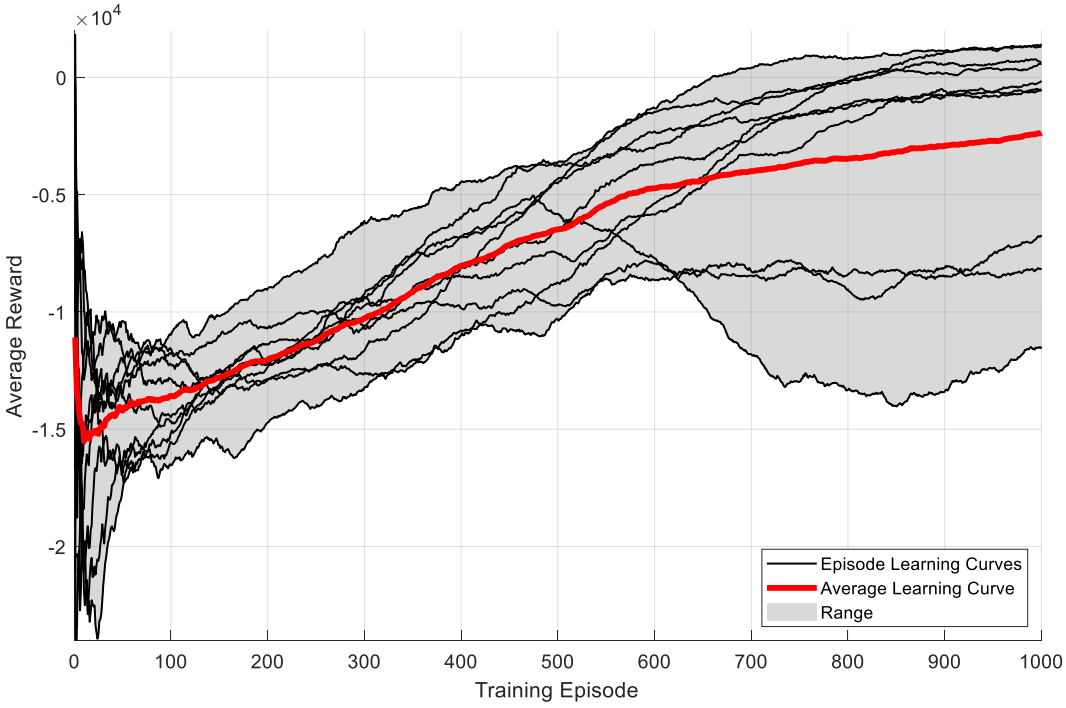



*Figure 35: Learning curves for run 23*

A similar spread of learning curves as shown in Figure 35, as well as the few lower-case outliers obtained for multiple repetitions of the same DDPG learning process, were observed in the literature for hexapod applications in the works of Schilling *et al.* [41] and Naya *et al.* [35]. The main two potential reasons for this spread of results are as follows: the stochastic nature of the reinforcement learning process, and the selection of RL learning parameters used. The reinforcement learning process, by definition, introduces a level of randomness to the results. This randomness is required in order for the reinforcement learning agent to explore the action space and escape local maximums in the reward surface. By its nature, the reinforcement learning process is not repeatable; however, tuning various parameters in both the reinforcement

learning agent and the learning routine can improve the repeatability of the final average reward for a given run. For the scope of this research, the final learning parameters used in the designed experiment were manually tuned and held constant throughout all of the designed experiments. Future work could carry out a more exhaustive parameters optimization method such as a grid search to try to further improve the repeatability of the learning results and reduce the number of outliers.

Since the spread of the final average rewards for a given run cannot be completely eliminated, and it is inevitable that there will be some outliers in the data, the three worst outlier replicants from each of the 32 runs were removed from the data. Given that the goal of this research is to explore the potential for the application of a designed experiment in the selection of observations for deployment on a physical robot hexapod, it is justifiable to remove these the outlying replicants because, in a deployment scenario, only the best reinforcement learning agents would be considered for the hexapod from multiple learning routines.

The same analysis as before is carried out using Minitab on the seven best replicants from each run of 10 to obtain a new model. With the outliers removed, the model now achieves a much better fit – with the R-squared, adjusted R-squared, and predicted R-squared values more than doubled. The model determined from the best seven analysis has an R-squared value of 73 %, an adjusted R-squared of 68 % and a predicted R-squared of 63 %. The improvements in R-squared values obtained by removing the three worst replicants are summarized in Table 19.

*Table 19: Improvement in R-squared statistics by dropping three worst replicants*

| Model Description | R-Squared | Adjusted R-Squared | Predicted R-Squared |
|---|---|---|---|
| Initial Model | 35 % | 28 % | 20 % |
| Best Seven Replicants Model | 73 % | 68 % | 63 % |
| Improvement in R-Squared | 38 % | 40 % | 43 % |

The residuals can again be plotted as a function of the run number to ensure that eliminating the worst three replicants has removed the outliers and improved the distribution of residuals. Figure 36 shows the plot of residuals versus run number. Comparing Figure 36 to

Figure 33, it can be observed that the residuals are still independent of run number but now appear more normally distributed about zero along the vertical y-axis as desired.

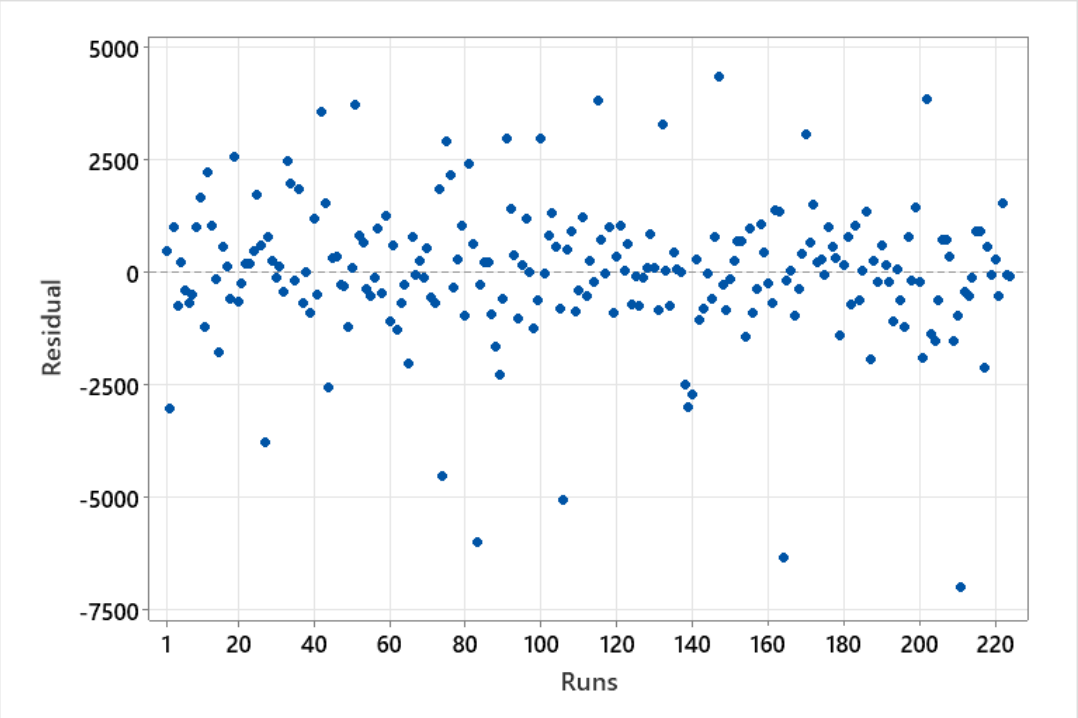



*Figure 36: Residuals plotted against run number for the best seven model*

Next a histogram is plotted in Figure 37 to check if the residuals follow the desired normal distribution. Comparing Figure 37 to Figure 34, it can be seen that the residuals now appear to be more normally distributed, as the distribution center is closer to zero and more symmetrical.

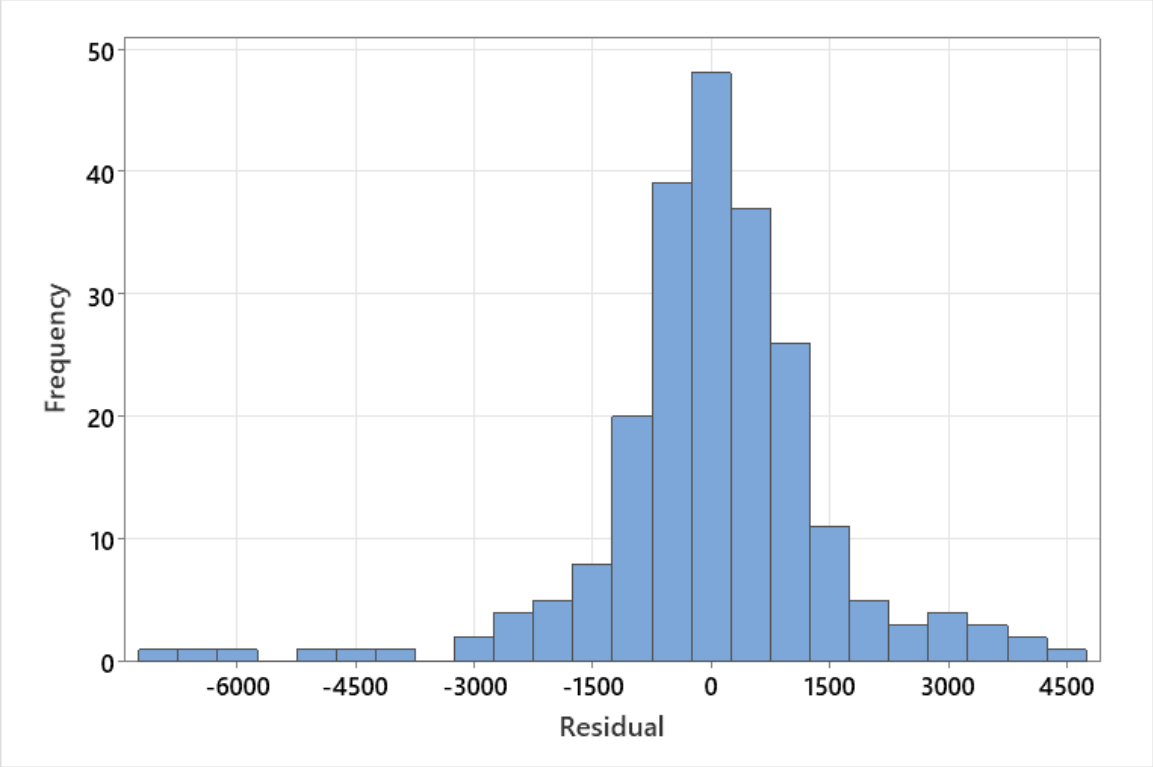



*Figure 37: Histogram of residuals for the best seven model*

The degree to which the residuals are normally distributed can be further confirmed using a normal probability plot. The normal probability plot shows the experimental data plotted against the theoretical values for the normal distribution fit to the data. Figure 38 shows such a plot for the residuals of the best seven models. The x-axis shows the values of the residuals, which are displayed in the plot in ascending order. The y-axis shows the corresponding theoretical probability that a random value taken from a normal distribution with the same mean and variance as the experimental data falls below the given experimental value. The probability plot shows how the experimental data differs from the normal distribution by illustrating deviations from the diagonal red line (where the red line represents a true random normal distribution). The blue experimental data exhibits an S-shaped curve, which indicates skewness in the distribution of the residuals.

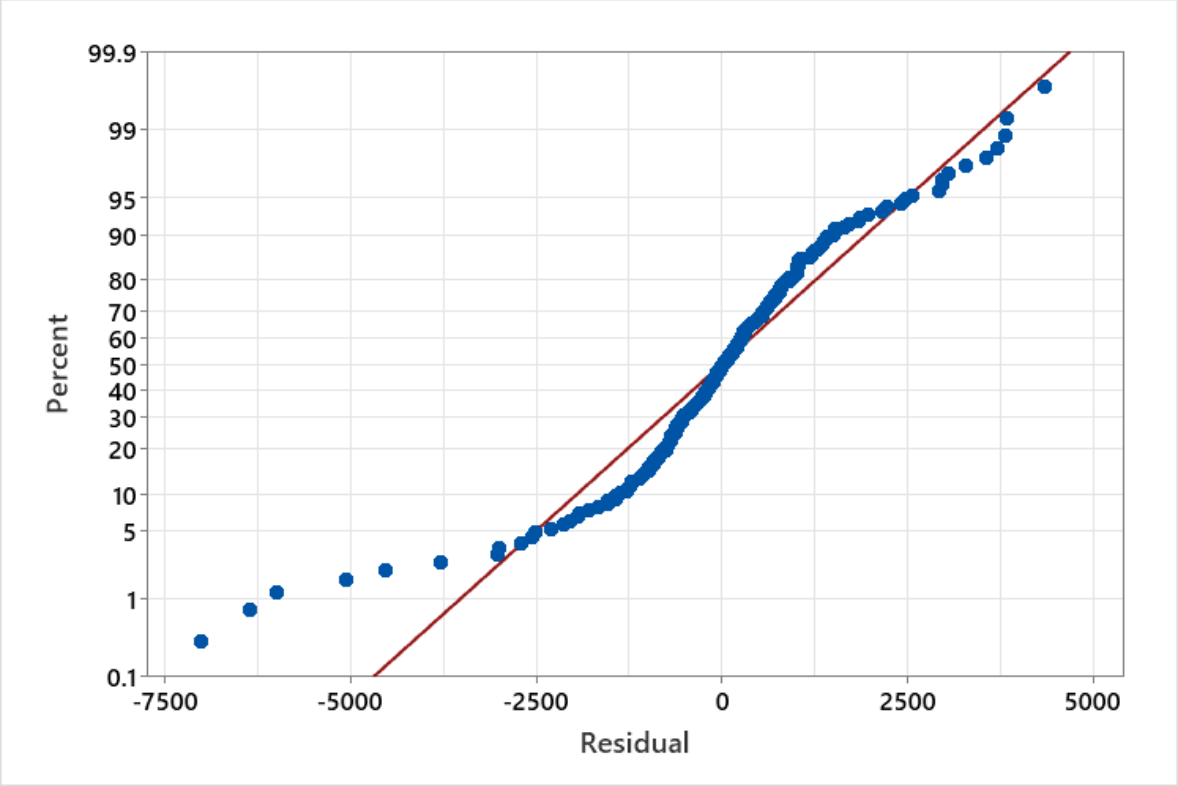



*Figure 38: Normal probability plot for the best seven model*

The skewness indicated by the normal probability plot was not prominent in the previously-used residual plots, so a new scatter plot is needed. Figure 39 is obtained by plotting the residuals against the corresponding final average reward predicted by the best seven model from which the residual originates. Since there are seven experimental data points but only one model prediction for each run, the residuals are displayed in vertical bands of seven points. The distribution shown in Figure 39 demonstrates a horn shape or skewness with respect to the final average reward as illustrated by the dashed magenta lines. From Figure 39, it appears that the model is better at predicting higher values of final average reward, as the residuals are tightly grouped on the right-hand side of the graph. Ideally this plot would not show such a trend as the model should be equally proficient at predicting values anywhere in the experimental range.

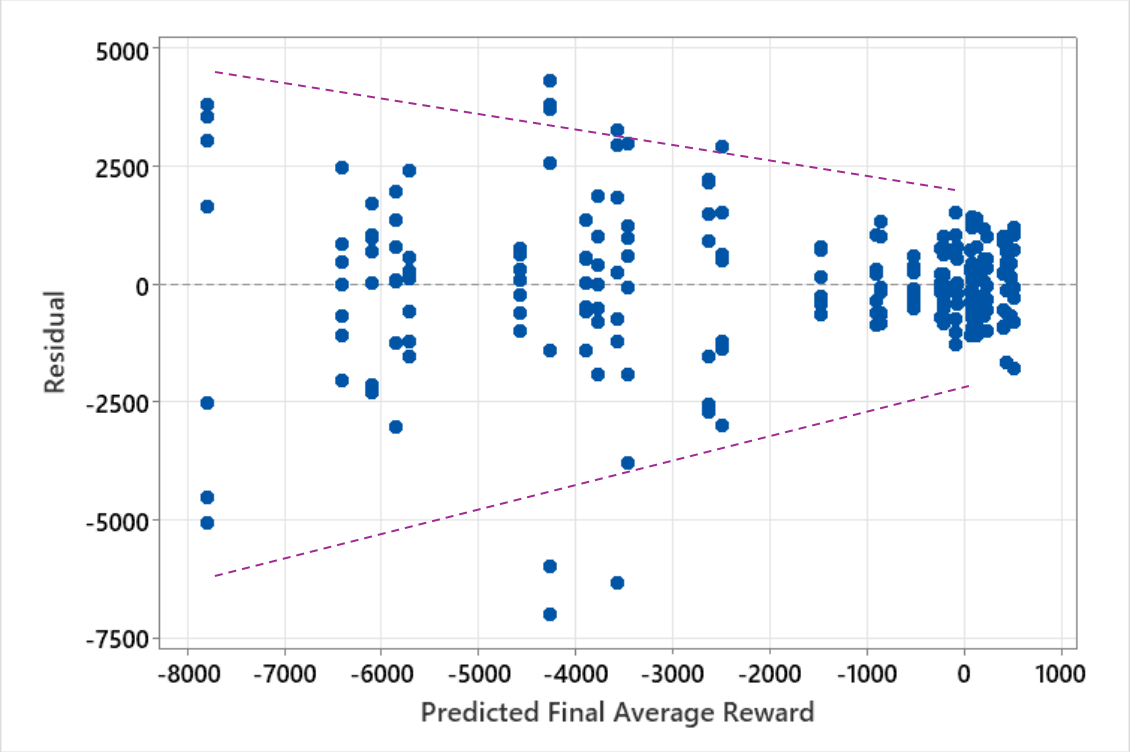



*Figure 39: Plot of residuals vs predicted final average reward for the best seven model showing a skewed or horn shaped distribution*

There is a potential argument for leaving the model with this bias towards better predictions at higher final average rewards, as this higher average reward region is of most interest in an optimization problem. However, in the interest of final reward optimization, an extrapolation may be needed to predict a final average reward that is higher than all those obtained during the designed experiment, which may fall outside of the accurate range of the skewed model. A more generalized model would be better suited to ensure the model has not been fit to only a portion of the data. In addition, a generalized model could be used to make predictions or help answer other questions over the entire range of possible final average rewards.

A transformation is, therefore, applied to the final average reward (designed experiment response) in order to correct the skewness of the residual distribution [5], [57], [60]. A new model is then fit to the transformed response data. To correct the S shape of the normal probability plot an exponential transformation is applied. The designed experiment response

(final average reward) is first scaled to reduce the effect of the transformation before applying the exponential. The resulting transformation function is shown in Equation 23:

$$Y' = e^{\frac{Y}{7500}} \tag{23}$$

where *Y* is the final average reward (response variable), and *Y'* is the resulting transformed response. The scaling factor was determined empirically using an iterative approach to maximize the R-squared value of the model fit to the transformed data, similar to the approach used by Pardoe *et al.* [60].

Once the transformation has been applied, Minitab is then used as before to fit a linear model to the now transformed response data. Without changing any parameters in the Minitab analysis process the model fit to the transformed data now achieves a fit with R-squared value of 77 %, adjusted R-squared value of 74 %, and predicted R-squared value of 69 %. Transforming the data results in an improved fit of about 5 % to 7 % as shown in Table 20.

*Table 20: Improvement in R-squared by applying transformation to response data*

| Model Description | R-Squared | Adjusted R-Squared | Predicted R-Squared |
|---|---|---|---|
| Best Seven Replicants Model | 73 % | 68 % | 63 % |
| Model using Transformed Data | 77 % | 74 % | 69 % |
| Improvement in R-Squared | 4 % | 6 % | 6 % |

A new normal probability plot is generated to ensure that transforming the response data has further increased the normality of the model residuals. Figure 40 shows this plot where, once again, the residuals are plotted in ascending order on the x-axis with the corresponding percent of random normal values that would be expected under their value on the y-axis. The normality of the residuals is much improved from the non-transformed result shown previously in Figure 38 since the residuals fall closer to the true normal distribution indicated by the red line. Note that

the values along the x-axis have changed due to the transformation; therefore, direct numerical comparisons between Figures 38 and 40 cannot be made.

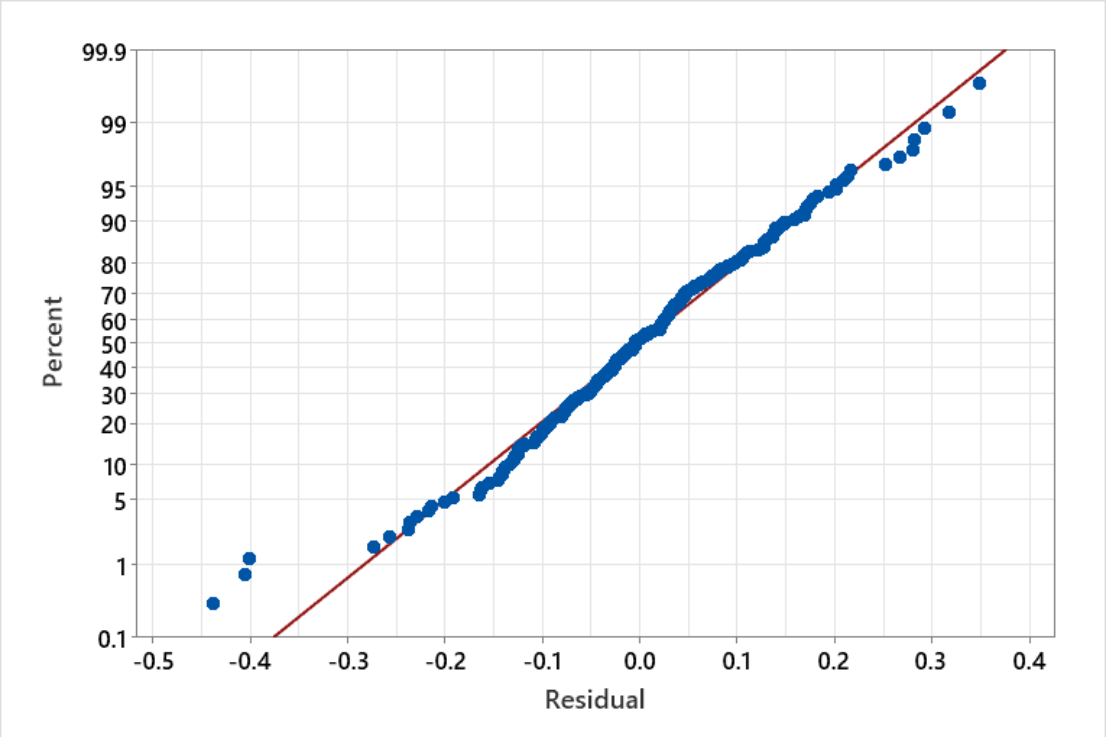



*Figure 40: Normal probability plot for model fit to transformed response data*

To view the effect of the transformation on the individual residuals with respect to the response value, Figure 41 is generated. The x-axis shows the response (transformed final average reward) predicted by the model and the y-axis shows the seven corresponding experimental residuals. When compared to Figure 39, the transformed response model produces a residual distribution which is more uniform (once again the units of the x-axis have changed due to the transformation). The tendency for the model to result in lower residuals only at higher response values has been removed, as the distribution has similar spread in the y-direction on both ends of the response range. A jump in the residuals near the center of the chart at roughly 0.65 transformed reward remains. This outlier was not removed by the transformation applied to the response data. One potential theory for this outlier is that this level of reward exists at the crossover point where the hexapod begins to have highly-successful learning routines, but the hexapod still experiences learning routines that result in a complete failure to learn the desired

behaviour. The learning in the middle of the range of rewards would be, therefore, inconsistent leading to larger residuals in this area.

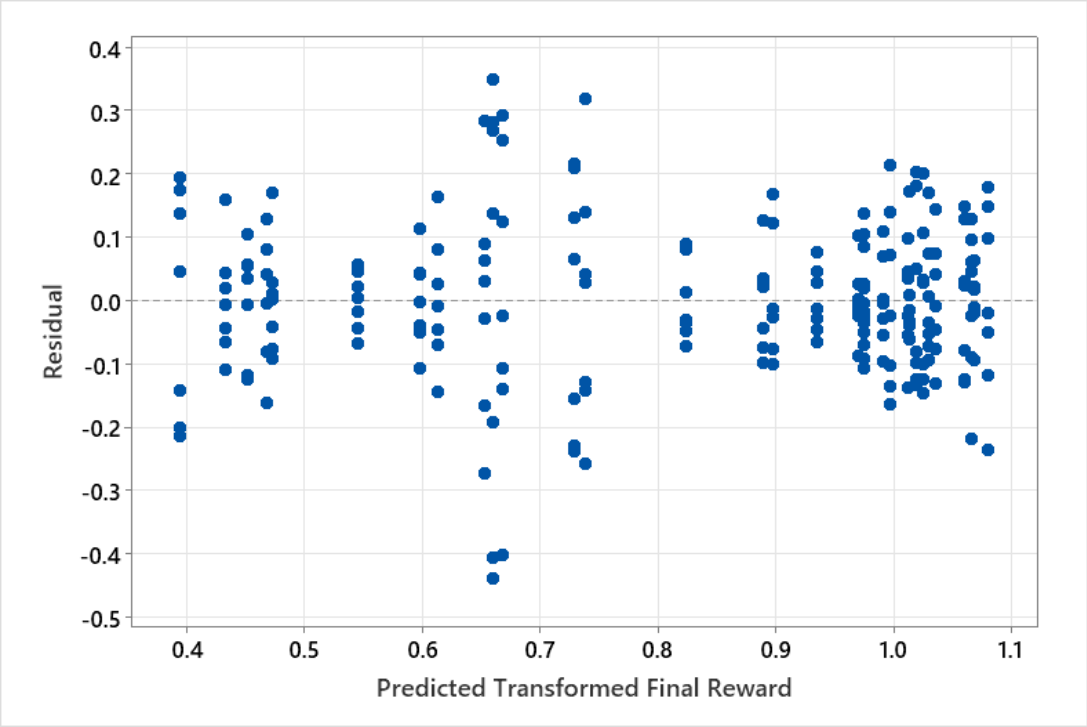



*Figure 41: Plot of residuals vs predicted response for the transformed response model*

The linear regression model produced using Minitab for the transformed response data is shown in Equation 24. The equation is in uncoded units and produces an estimate for the transformed final average reward Y'. All five decimal places shown by Minitab are retained to ensure the maximum possible accuracy is maintained, as a small change in the transformed response can lead to a much larger change in final average reward when the data transformation is reversed.

$$
\begin{aligned}
Y' = 0.82132 \; &+ \; 0.01430\,A \; + \; 0.08893\,B \; + \; 0.18580\,C \; + \; 0.00557\,D + \; 0.01102\,E \; - \; 0.02781\,F \\
&- \; 0.00955\,G \; - \; 0.00088\,AB \; - \; 0.00642\,AC \; + \; 0.00926\,A* \; + 0.03372\,AE \\
&+ \; 0.00449\,AF + \; 0.00658\,AG \; - \; 0.04969\,BC \; + \; 0.00185\,BD \; - \; 0.00020\,BE \\
&- \; 0.00629\,BF + \; 0.00857\,BG \; - \; 0.00903\,CD \; - \; 0.00715\,CE \; + \; 0.01014\,CF \\
&+ \; 0.01170\,CG + \; 0.00465\,DE \; - \; 0.01724\,DF \; + \; 0.01377\,DG \; - \; 0.03979\,ACE \\
&- \; 0.01615\,ACG + \; 0.00175\,BCE \; - \; 0.01124\,BCG \; - \; 0.00889\,CDE \\
&- \; 0.00954\,CDG
\end{aligned}
\tag{24}
$$

In Equation 25, the letters A through G are the observation-indicating letters and correspond to the presence (+1) or absence (−1) of the corresponding observation. The model contains all possible interaction terms that could be estimated using the 32 runs from the designed experiment. The quarter fraction designed experiment with 32 unique runs is able to estimate 32 parameters for the regression equation, including the constant term, the coefficients for all seven main factors, 18 of the 21 possible first order interactions, and 6 second order interaction terms.

As can be seen in Equation 24, the coefficients range from 0.18580 for the main effect C to three orders of magnitude smaller at 0.00020 for the B*E interaction term. It is clear that not all terms are significant and model reduction should therefore be applied to arrive at the final model. As well as simplifying the model/equation, eliminating the insignificant terms will reduce overfitting to the experimental data. Overfitting occurs when a model contains too many parameters for the complexity of the data, and results in the model being fit to noise within the dataset and not just to the underlying trends, leading to a reduction in model accuracy for new data and predictions. Model reduction will help generalize the model in order to make better predictions about untested combinations of observations.

A P-value criteria is used to determine which terms should remain in the model [57], [60]. The P-values are calculated by Minitab based on the distribution of the replicants' results for different factor levels and are related to a null hypothesis about the effect of the model terms. The null hypothesis taken in this case is that the effect of each term on the response is zero, meaning the responses with a given factor turned on or off are the same and any difference is indistinguishable from random noise. The null hypothesis needs to be rejected for each factor to confirm their significance. The P-value represents the probability that the measured change in

response (effect) could be explained by random noise and not the change in factor. The smaller a P-value, the more likely it is that the model term has a significant effect on the response. Table 21 shows the 32 model terms along with their corresponding effects on the model prediction, coefficient, and associated P-value. The indicating letter representing each observation is either set to 1 or −1 in the model if the given observation is turned on or off, respectively. Switching an observation from off to on results in a change in the associated model term by a factor of two, as the indicating letters switches from a value of –1 to +1, hence the associated coefficient must be half of the term's effect.

*Table 21: Effect, coefficient value and P-value for each term of the complete model*

| Term | Effect | Coefficient | P-Value |
|---|---|---|---|
| **Constant** | | 0.82132 | **0.000** |
| **A** | 0.02861 | 0.01430 | 0.105 |
| **B** | 0.17786 | 0.08893 | **0.000** |
| **C** | 0.37161 | 0.18580 | **0.000** |
| **D** | 0.01115 | 0.00557 | 0.526 |
| **E** | 0.02205 | 0.01102 | 0.211 |
| **F** | -0.05562 | -0.02781 | **0.002** |
| **G** | -0.01911 | -0.00955 | 0.278 |
| A*B | -0.00176 | -0.00088 | 0.920 |
| **A*C** | -0.01285 | -0.00642 | 0.465 |
| A*D | 0.01851 | 0.00926 | 0.293 |
| **A*E** | 0.06745 | 0.03372 | **0.000** |
| A*F | 0.00897 | 0.00449 | 0.610 |
| A*G | 0.01315 | 0.00658 | 0.455 |
| **B*C** | -0.09938 | -0.04969 | **0.000** |
| B*D | 0.00369 | 0.00185 | 0.834 |
| B*E | -0.00040 | -0.00020 | 0.982 |
| B*F | -0.01257 | -0.00629 | 0.475 |
| B*G | 0.01714 | 0.00857 | 0.330 |
| C*D | -0.01805 | -0.00903 | 0.305 |
| **C*E** | -0.01430 | -0.00715 | 0.416 |
| C*F | 0.02028 | 0.01014 | 0.249 |
| C*G | 0.02339 | 0.01170 | 0.184 |
| D*E | 0.00929 | 0.00465 | 0.597 |
| D*F | -0.03449 | -0.01724 | 0.051 |
| D*G | 0.02754 | 0.01377 | 0.118 |
| **A*C*E** | -0.07958 | -0.03979 | **0.000** |
| A*C*G | -0.03231 | -0.01615 | 0.067 |

| Term | Effect | Coefficient | P-Value |
|---|---|---|---|
| B*C*E | 0.00350 | 0.00175 | 0.842 |
| B*C*G | -0.02249 | -0.01124 | 0.202 |
| C*D*E | -0.01778 | -0.00889 | 0.312 |
| C*D*G | -0.01908 | -0.00954 | 0.278 |

A statistical significance level is selected to filter those model terms which will be dropped. A significance level of 0.05 is used as it is the most common threshold for level of significance [58].

To remove insignificant terms and reduce the model to a simpler form, a procedural backward elimination is used in Minitab. In each iteration, the model term with the highest P-value is removed and the model is refit to the experimental data using the remaining terms. The process is repeated until only terms with a P-value less than or equal to the chosen statistical significance of 0.05 remain. There are two exceptions. First, all first-order factors (the seven observations) are kept in the model regardless of significance in order to be able to predict the best possible combination of observations. Second, since one of the second-order interactions was found to be significant, the first order interactions which make up the second order term were retained to maintain model hierarchy. The principle of effect heredity dictates that an interaction term is much more likely to be significant if one or more of the terms which make up the interaction are significant themselves [63]. For example, the significance of the A*C*E second-order interaction term indicates that one or more of the factors or interactions that make up the second order term may be significant. While further testing targeted to the A*C*E interaction term could provide more insight, for the purposes of this work model hierarchy is maintained by retaining the constituent terms of the second-order interaction: the A*C, A*E, and A*E terms regardless of their individual significance.

Using a P-value significance level of 0.05, only the bold terms in Table 21 were kept to generate the final reduced model which is shown in Equation 25. The final model contains a constant term, the seven main effects, four first order interactions, and a single second order interaction.

$$Y' = 0.82132 + 0.01430\,A + 0.08893\,B + 0.18580\,C + 0.00557\,D + 0.01102\,E - 0.02781\,F - 0.00955\,G - 0.00642\,A*C + 0.03372\,A*E - 0.04969\,B*C - 0.00715\,C*E - 0.03979\,A*C*E \quad (25)$$

The final model achieves a fit quantified by an R-squared value of 75 %, an adjusted R-squared value of 73 %, and a predicted R-squared value of 72 %. Table 22 summarizes the changes in R-squared values due to model reduction. Reducing the model has brought the predicted R-squared value to within just over three percent of the R-squared value, indicating that the model is not overfitting to the designed experiment data. Both the R-squared value and the adjusted R-squared value have decreased slightly from the model reduction while the predicted R-squared has been increased as shown in Table 22. The R-squared and adjusted R-squared values have decreased because the complexity of the model has been reduced limiting its ability to fit the noise in the dataset. However, the predicted R-squared value has increased indicating that the reduced model is better at capturing the underlying trends in the data while ignoring the residual noise; therefore, overfitting to the data as been reduced.

*Table 22: Change in R-Squared values from model reduction*

| Model Description | R-Squared | Adjusted R-Squared | Predicted R-Squared |
|---|---|---|---|
| Model using Transformed Data | 77 % | 74 % | 69 % |
| Final Reduced Model | 75 % | 73 % | 72 % |
| Change in R-Squared | –2 % | –1 % | +3 % |

The residuals of the final model are plotted with respect to the run order in Figure 42 to confirm that they appear to be independent and centered around zero.

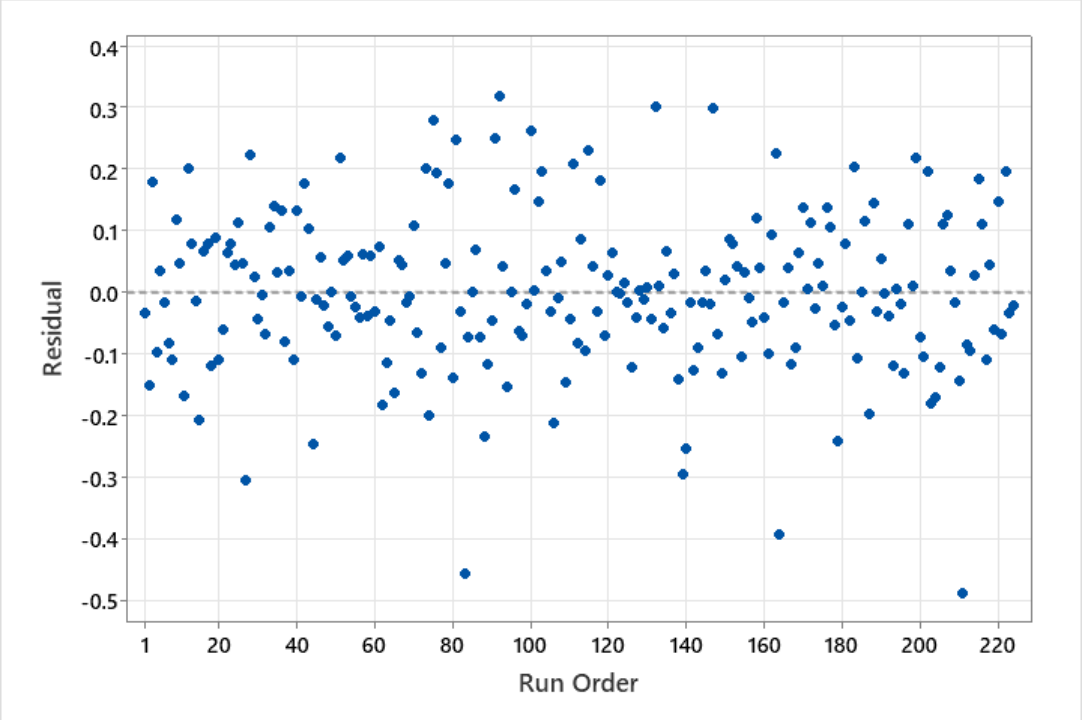



*Figure 42: Plot of residuals against run order for the final model*

The histogram of residuals shown in Figure 43 displays a wide peak centered around zero which shows that the model is highly accurate and well fit to many of the experimental data points. There are three residuals in a very small tail on the left-hand side of the distribution. These are remnants of the eliminated outliers and are to be expected from the runs with combinations of observations that result in inconsistent and poor learning. These three points can also be seen in the lower portion of the previous Figure 42.

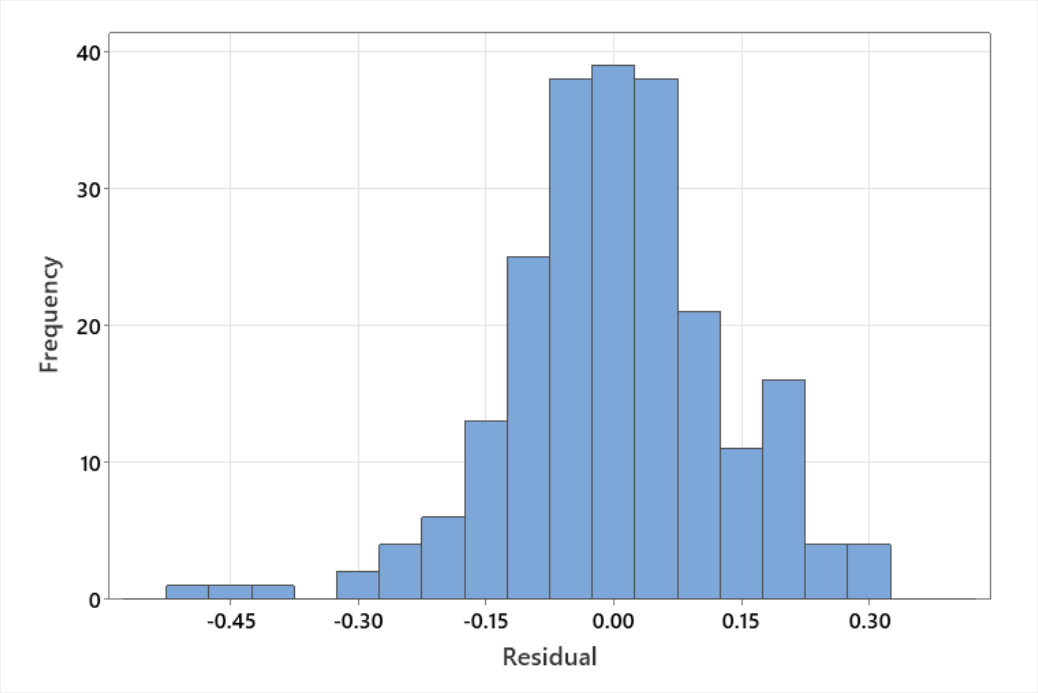



*Figure 43: Histogram of residuals for the final model*

The final residuals are also plotted on a normal probability plot shown in Figure 44. It can be seen in this figure that the residuals closely follow the red line for a true normal distribution, again confirming the successful fit of the final model.

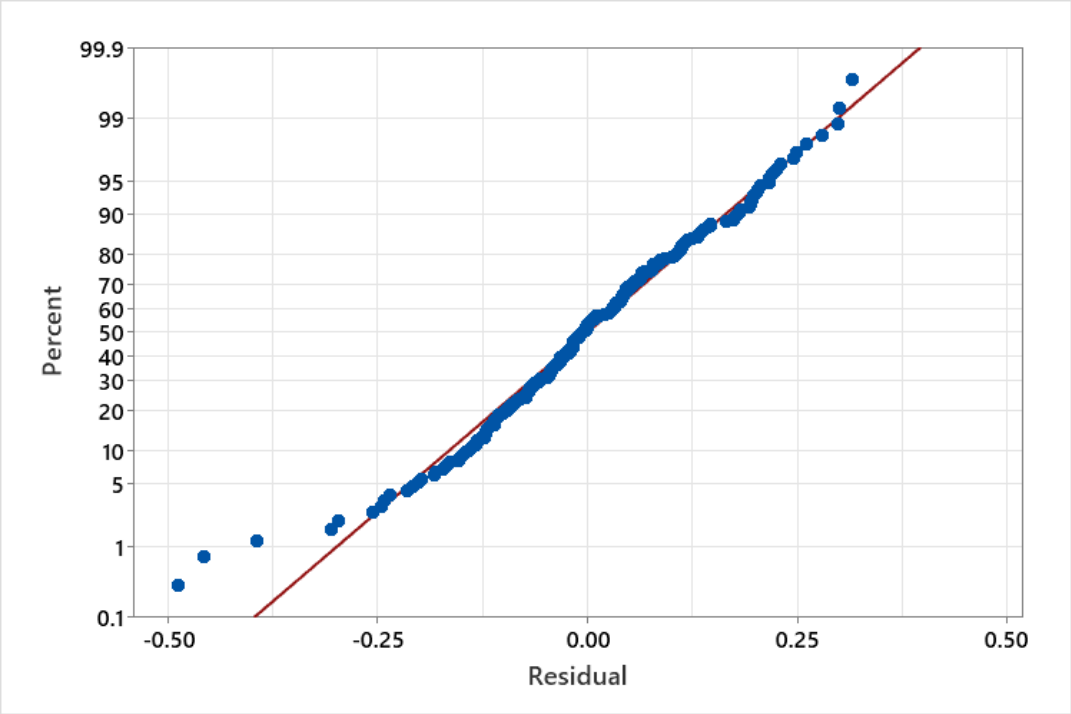



*Figure 44: Normal probability plot for the final model*

Figure 45 plots the residuals with respect to the predicted response (transformed final average reward). The range of the residuals is fairly uniform across the entire range of predicted responses, with only a few instances falling outside of ± 0.3.

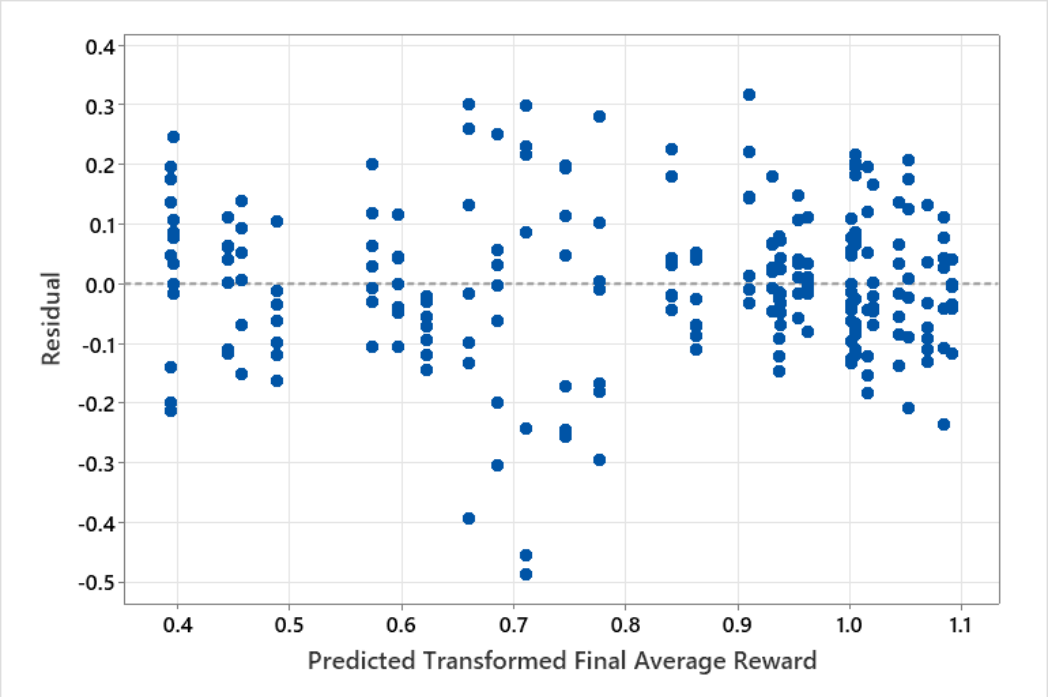



*Figure 45: Plot of the residuals vs predicted response for the final model*

Table 23 summarizes how the R-squared, adjusted R-squared and predicted R-squared values changed with each step in the model refinement. It can be seen from this table that the most significant improvement in model accuracy was obtained by eliminating the three worst replicants from each run as the R-squared values more than doubled. The original dataset would have been unusable without this important truncation. Applying the exponential transformation to the final average reward data further increased all three R-squared values by 5-7 %. Finally, reducing the model helped to reduce overfitting, as illustrated by the reduction in R-squared and the adjusted R-squared, while showing an increase in predicted R-squared. The reduced model is better generalized to make more accurate predictions of unseen circumstances as demonstrated by the increase in predicted R-squared. Achieving higher R-squared values would likely be difficult for this particular application because of the stochastic nature of the reinforcement learning process since the random noise introduced during the learning process is difficult to model.

*Table 23: Improvement in model fit with each key step in the analysis*

| Model Description | R-Squared | Adjusted R-Squared | Predicted R-Squared |
|---|---|---|---|
| Initial Model | 35 % | 28 % | 20 % |
| Best Seven Replicants Model | 73 % | 68 % | 63 % |
| Model using Transformed Data | 77 % | 74 % | 69 % |
| Final Reduced Model | 75 % | 73 % | 72 % |

The final model contains four first-order interaction terms and a single second-order interaction. An interaction plot can be created to study these effects as shown in Figure 46. This figure shows all possible first-order interactions between the seven main effects studied in the designed experiment in a grid format, with each interaction represented in its own subplot. Each subplot contains two lines which show how the response (the transformed final average reward indicated on the vertical axis) changes with the four possible combinations of levels of the interacting factors. The horizontal axis indicates the level of the first of the interacting factors, while the colour of the line represents the level of the second factor in the interaction (see legend directly to the right of the plot). The blue line shows the effect that the first factor has on the response when the second factor is at its low level, while the red line shows the effect of the first factor when the second factor is at its high setting. The four points in each interaction subplot effectively show the four possible combinations for the two observations in question: off-off, off-on, on-off and on-on. The blue and red lines are parallel if there is no interaction present between the factors. An interaction is indicated by non-parallel lines in one of the subplots and, the stronger the interaction, the further the lines will be from parallel.

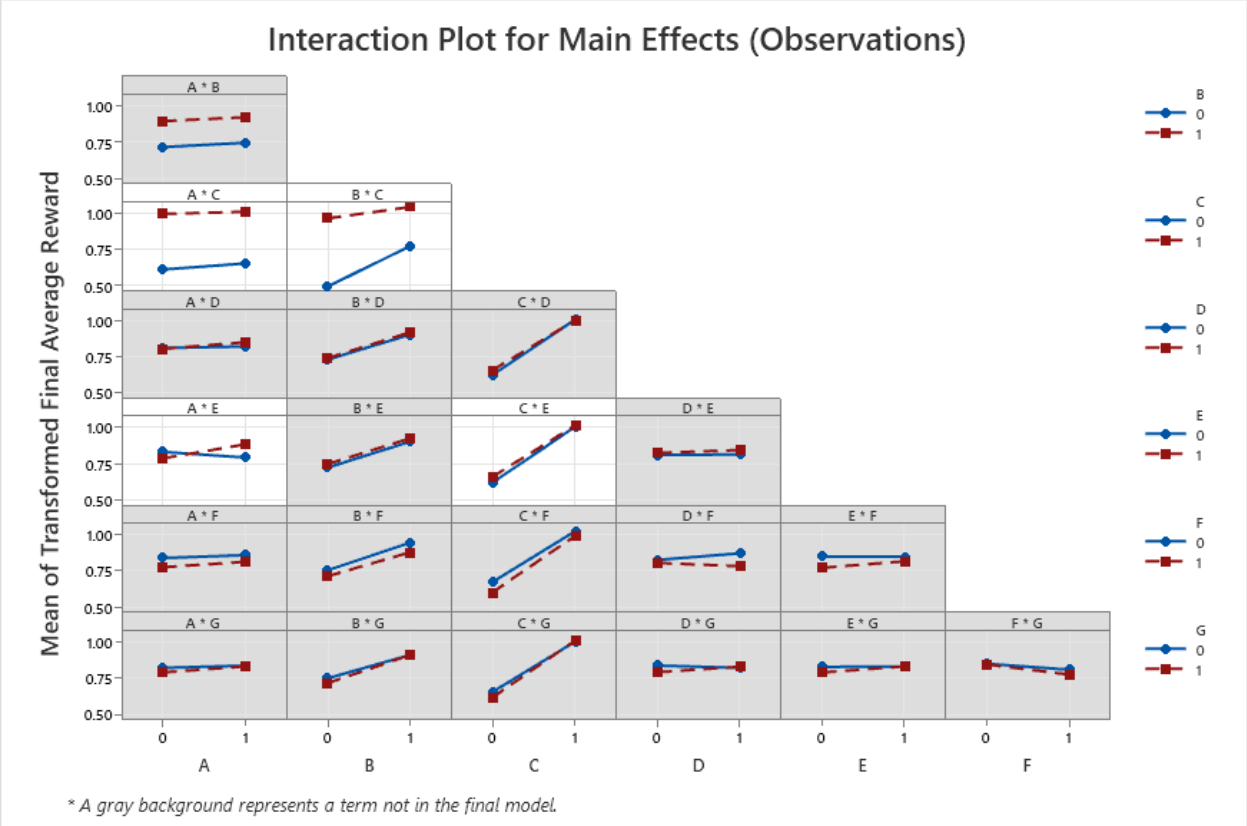



*Figure 46: Interaction plots for all possible first order interactions (those included in final model have subplots with white backgrounds)*

The four subplots with white backgrounds in Figure 46 are the first-order factors included in the final model, while those with grey backgrounds were eliminated during model reduction. Only two first-order interactions are significant with a P-value less than the 0.05 threshold: the B*C interaction between the body velocities (B) and body tilt (C) and the A*E interaction between joint torques (A) and body height (E). The other two first order interactions included in the model were retained to maintain model hierarchy as they are components of the significant A*C*E second-order interaction.

The B*C interaction plot shows that the positive effect of adding observation B, the body linear velocities, is reduced when the body tilt (C) is used as an observation. The body velocities and body tilt are very closely related – being measured at the same point on the hexapod body. It makes sense that this interaction could be present, as both observations could provide similar information to a reinforcement learning agent about the cyclic motion of the hexapod body

throughout the walking gait. There could be some overlap in the benefit provided by these two observations so adding the body velocities is not as impactful to the learning when the body tilt is already present, and vice versa. However, the main effect of the body tilt is so significant that the maximum response is still achieved when both observations are present.

The A*E first order interaction is an interesting case. The average response actually decreases with the addition of the joint torques (A) if the body height (E) is not present, but the response increases if the body height (E) is included. The A*E subplot in Figure 46 indicates that it is better to have either both observations turned on or both turned off than it is to use joint torques (A) or only body height (E) on their own. Note that a decision to include or exclude a certain observation(s) cannot be made based on a single interaction, as in this case both factors are also contained within the significant A*C*E second order interaction which must also be considered.

The A*C and C*E interactions do not display any significance on their own, but, are included in the model as they may be components of the significant A*C*E interaction term. The nature of the A*C*E second order interaction is not readily apparent, but its significance is most like attributed to the body tilt (C) as it is the main factor with the largest effect on the response. It is hypothesized that the second-order interaction is between the body tilt (C) and the A*E first order interaction, but more data is needed to fully explore the A*C*E interaction. Future work could carry out additional simulation runs which might reveal more information about these higher-order interactions. However, for the scope of this research in which the focus is the observations as main factors, the next step is to generate the final regression model and validate its accuracy.

## 5.4. Validation of the Designed Experiment Regression Model

The fourth objective of this thesis is to demonstrate the potential for using a fractional factorial designed experiment to select the observations for a specific hexapod reinforcement learning case that will maximize hexapod performance. To be able to validate the model identified using the designed experiment, it must first be confirmed that the reward metric used to quantify performance corresponds to the desired hexapod behaviour. The two main

characteristics of the desired hexapod behaviour are: the ability for the hexapod to steer towards and track along the goal trajectory line, and for the hexapod to learn to walk as fast as possible along the aforementioned path. To quantify the success of each combination of observations during deployment, a set of performance metrics based on the desired hexapod behaviour are created to classify each trial as a success or failure (see Table 24). To be considered a successful trained agent, the hexapod must be able to walk 4 meters in the direction parallel to the goal trajectory and reduce its starting offset to within ± 0.25 meters of the goal trajectory line – both within 30 seconds of a random initialization. The 0.25 meter offset limit was determined to be an appropriate range based on experience with the hexapod learning results during testing. The combination of the 30 second time limit and minimum displacement in the x-direction of 4 m originates from the maximum speed of the hexapod using a nominal gait (parameterized gait with no reinforcement learning), with a roughly 20 % reduction in required speed to account for the starting offset.

*Table 24: Performance conditions to be deemed a "successful" learning run*

| **Performance Metric** | **Condition for Success** |
|---|---|
| Time Limit | $t \leq 30\ s$ |
| X Displacement | $X \geq 4\ m$ |
| Y Offset | $-0.25\ m \leq Y \leq 0.25\ m$ |

When the final learning agent of each training routine is tested, the performance metrics are used to determine if the hexapod was successful at each test point. Figure 47 shows, for example, the test paths for all 10 replicants of run 8 from the designed experiment. The final agent of each replicant is tested at the seven different starting offsets and the total number of paths that meet the performance metrics are recorded. The performance metrics lead to a “success area” as shown in Figure 47 that the hexapod should reach in the 30 second test length.

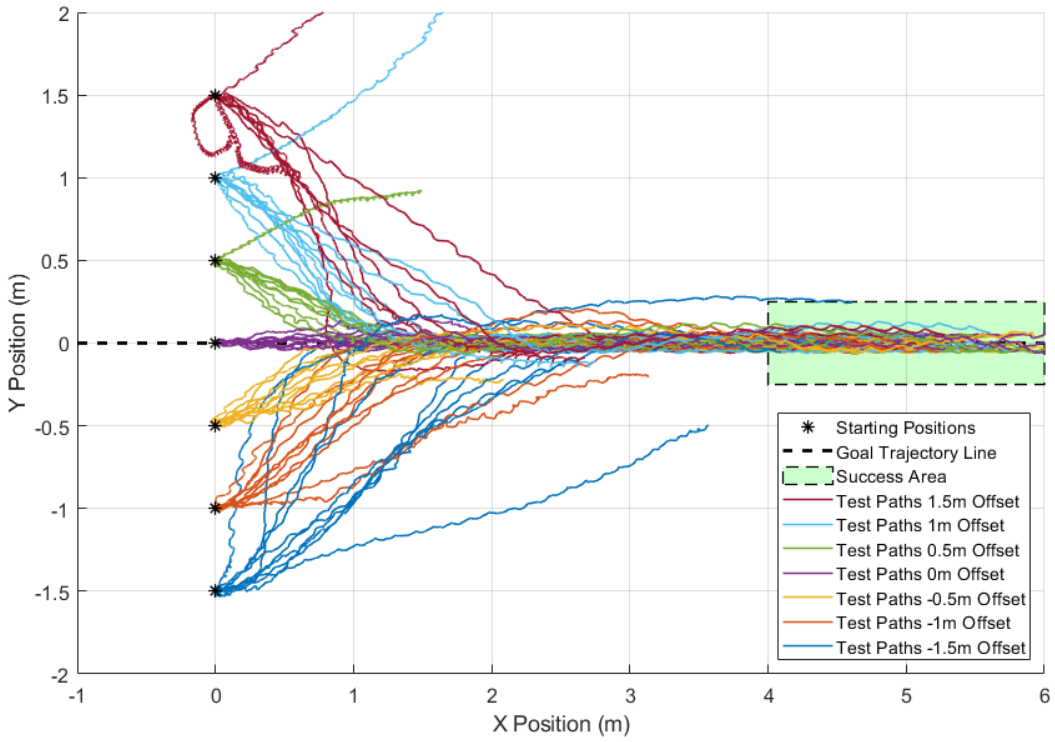



*Figure 47: The "success area" set by the performance conditions shown over an example test paths plot*

The number of successful paths from all 70 tests for each run are recorded (7 starting offsets for each of 10 replicants). They can then be compared to the final average rewards obtained during training for the agents in each run. Figure 48 plots a comparison between the final average reward for all 10 replicants of a run, the final average reward for the best 7 replicants of a run, and the number of successful paths from a run's ten replicants all together on a single graph. The runs are plotted in ascending order of best 7 final average reward along the x-axis. The final average reward for all 10 replicants, plotted in green, and the average reward for the best 7 replicants, plotted as a blue line, correspond to the y-axis located on the left of the plot. The number of successful paths from all ten replicants of each run is plotted using the orange line, corresponding to the secondary y-axis to the right side of Figure 48. The plotted path counts (orange curve) cannot be numerically compared to the reward data (green and blue curves), but the overall shape of the curves can be examined.

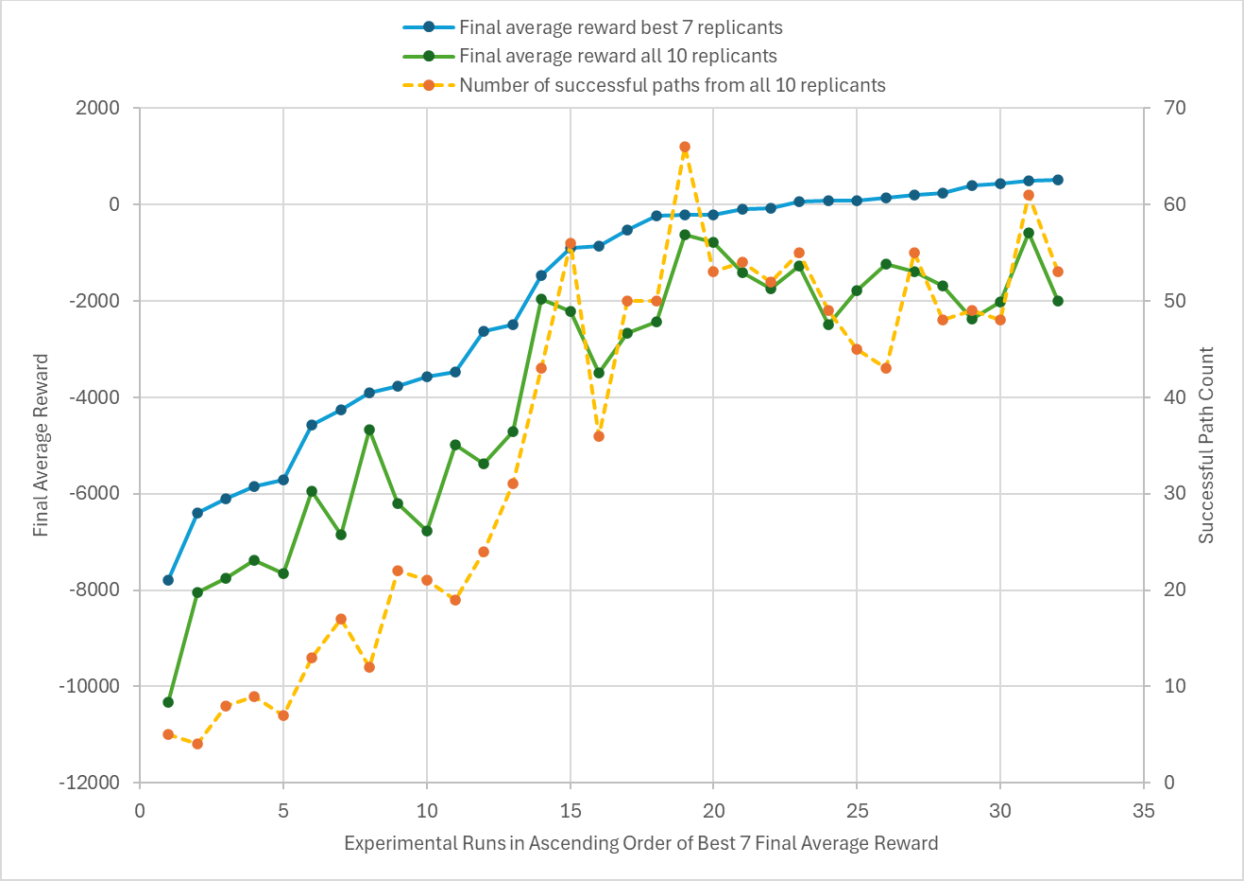



*Figure 48: Comparison across all experimental runs of the mean final average reward, the mean reward of the best seven cases, and the number of successful test paths*

Both of the reward curves follow a similar logarithmic shape, with the blue best seven data resulting in both higher rewards and a smoother curve than the green curve which uses all ten replicants. This result is as to be expected from dropping the three lowest outliers in each run. The orange curve, which represents the number of successful paths for each run, follows a similar shape to the reward curves. Neither the path count nor the reward data using all 10 replicants are smooth curves because no outliers have been dropped from both datasets. Given the increasing trends in the data, Figure 48 confirms that the reward function used during training quantifies the hexapod performance in such a way that it can directly be correlated with the desired hexapod behaviour. This correlation allows for more confidence in the results of the designed experiment – that the final model will be able to correctly provide numerical outputs with direct correlation to the hexapod performance.

The final model is validated by predicting the best possible combination of observations using the model and then testing the predicted case in simulation. The final linear regression model is shown again, for convenience, in Equation 26.

$$\begin{aligned} Y' = \; & 0.82132 + 0.01430\,A + 0.08893\,B + 0.18580\,C \\ & + 0.00557\,D + 0.01102\,E - 0.02781\,F - 0.00955\,G \\ & - 0.00642\,A*C + 0.03372\,A*E - 0.04969\,B*C - 0.00715\,C*E \\ & - 0.03979\,A*C*E \end{aligned} \tag{26}$$

The optimal set of observations is selected to maximize the transformed response Y’ in Equation 26. Minitab is used to calculate the predicted transformed response for all combinations of observations and the three best solutions are listed in Table 25. Listed in the table for each solution are the levels for each of the seven main factors (observations) and the predicted transformed response Y’ from the model with a 90 % confidence interval calculated using the experimental data in Minitab. The transformed data is then converted back into a predicted final average reward again with a 90 % confidence interval. There is overlap in the confidence intervals for final average reward of the top few optimal solutions. The body velocities (B) and body tilt (C) were found to be the observations with the greatest effect on final average reward, so it can be seen that the three optimal solutions contain both these observations. The overlap in confidence intervals exists between these three solutions because they differ by observations that have a lesser effect on the final average reward. Note that solution 3 is the same combination of observations as run 31 of the designed experiment.

*Table 25: Top three predicted solutions (combinations of observations) to maximize the final average reward*

| Optimal Solutions | Factors | | | | | | | Transformed Response | | Final Average Reward | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | Prediction | 90% Confidence Interval | Prediction | 90% Confidence Interval |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1.099 | (1.047, 1.152) | 711 | (343, 1061) |
| 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1.095 | (1.042, 1.148) | 681 | (311, 1032) |
| 3 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1.091 | (1.039, 1.144) | 656 | (285, 1009) |

According to the model built using the designed experiment data, for the simulation conditions used in this research the optimal observations to include from the seven options to maximize the final average reward achieved by the hexapod agent are the joint torques (A), the body velocities (B), the body tilt (C) and the body angular velocities (D). The model predicts this combination of observations would result in a final average reward of 711, with a 90 % confidence interval of 343 to 1061. The confidence intervals are fairly wide as the experimental data had a lower level of precision due to the stochastic nature of reinforcement learning, and a similar variance in learning? results has been shown in the literature [35], [41]. Due to the overlap in confidence intervals of all three optimal solutions, during validation testing one could expect any of the three solutions to produce the highest final average reward and be well within the confidence intervals. The model will be considered a success if one of these three solutions produces a final average reward equal to or higher than all other combinations of observations tested during the designed experiment.

The three optimal solutions are tested for the full ten replicants, and the lowest three results are removed (as done previously). The model was fit to the best seven replicants of the experimental runs so, for a fair comparison, the validation run must maintain the same format. Since solution 3 is the same as run 31 of the designed experiment, this case was not needed to be run again. The average final reward taken over the best 7 replicants for all 32 experimental runs as well as the predicted three optimal solutions are listed in Table 26.

*Table 26: Summary and ranking of the mean best seven reward for all cases tested*

| Run | Observations | | | | | | | Average Final Reward Over Best 7 Replicants | Ranking by Final Reward |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | | |
| 1 | - | - | - | - | - | + | + | -6407 | 33 |
| 2 | + | - | - | - | - | - | - | -5850 | 31 |
| 3 | - | + | - | - | - | - | - | -858 | 19 |
| 4 | + | + | - | - | - | + | + | -3572 | 25 |
| 5 | - | - | + | - | - | - | + | -241 | 17 |
| 6 | + | - | + | - | - | + | - | -80 | 13 |
| 7 | - | + | + | - | - | + | - | 75 | 11 |
| 8 | + | + | + | - | - | - | + | 196 | 8 |
| 9 | - | - | - | + | - | - | - | -3764 | 26 |
| 10 | + | - | - | + | - | + | + | -7793 | 34 |
| 11 | - | + | - | + | - | + | + | -2498 | 22 |
| 12 | + | + | - | + | - | - | - | -2638 | 23 |
| 13 | - | - | + | + | - | + | - | -909 | 20 |
| 14 | + | - | + | + | - | - | + | 71 | 12 |
| 15 | - | + | + | + | - | - | + | 514 | 2 |
| 16 | + | + | + | + | - | + | - | 232 | 7 |
| 17 | - | - | - | - | + | + | - | -5716 | 30 |
| 18 | + | - | - | - | + | - | + | -4577 | 29 |
| 19 | - | + | - | - | + | - | + | -4265 | 28 |
| 20 | + | + | - | - | + | + | - | -1471 | 21 |
| 21 | - | - | + | - | + | - | - | -212 | 16 |
| 22 | + | - | + | - | + | + | + | -211 | 15 |
| 23 | - | + | + | - | + | + | + | 397 | 6 |
| 24 | + | + | + | - | + | - | - | 435 | 5 |
| 25 | - | - | - | + | + | - | + | -6096 | 32 |
| 26 | + | - | - | + | + | + | - | -3900 | 27 |
| 27 | - | + | - | + | + | + | - | -3469 | 24 |
| 28 | + | + | - | + | + | - | + | 131 | 9 |

| Run | Observations | | | | | | | Average Final Reward Over Best 7 Replicants | Ranking by Final Reward |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | | |
| 29 | - | - | + | + | + | + | + | -520 | 18 |
| 30 | + | - | + | + | + | - | - | -87 | 14 |
| 31 | - | + | + | + | + | - | - | 488 | 3 |
| 32 | + | + | + | + | + | + | + | 83 | 10 |
| Sol. 1 | + | + | + | + | - | - | - | 449 | 4 |
| Sol. 2 | + | + | + | + | + | - | - | 585 | 1 |
| Sol. 3 | - | + | + | + | + | - | - | 488 | 3 |

The results from all three validation runs (last three rows of Table 26) fall within the confidence intervals listed in Table 25. The second solution resulted in the highest final average reward of the three at 585, followed by the third solution at 488, and finally the first solution with a reward of 449. The second solution was successful in producing the highest final average reward of any set of observations tested during this work. The second highest reward was 514 from run 15 of the designed experiment, with the third and fourth highest rewards from Solutions 3 and 1, respectively.

The model determined using the designed experiment was able to predict the combination of observations that would result in the highest average final reward as well as three of the top four runs. Since the confidence intervals are relatively wide and the learning results will always contain some level of noise, any one of the three top solutions could have produced the highest average final reward – in this instance it was Solution 2. These results validate the use of a designed experiment as a potentially valuable tool to be used in the selection and optimization of observations for a hexapod locomotion problem. Based on these validation tests, the recommended observations to obtain maximum final average reward for a hexapod applied to the conditions simulated in this work are the joint torques (A), body velocities (B), body tilt (C), body angular velocities (D), and body height (E).

## 5.5. Demonstration of Hexapod Path Following Capabilities

The final thesis objective is to demonstrate the potential of the trajectory following learning used in this work as a method for more complex hexapod directional control during deployment. The concept behind the goal trajectory line and offset observation is for the trajectory to be provided by a higher-level path-planning controller that the trained reinforcement learning agent then follows. The higher-level planning algorithm would discretize the desired path into a series of goal trajectories that do not need to be updated at as fast a sample rate as the reinforcement learning agent, reducing the processing required by the hexapod. The goal trajectory provided to the hexapod agent can then be updated only when a change in direction is required.

A trained reinforcement learning agent is deployed within the simulator for the following example cases to demonstrate the ability of the hexapod to follow a moving goal trajectory line. Figure 49 shows a case where the goal trajectory line alternates at regular intervals between $x = -1$ and $x = 0$ corresponding to Trajectory Line 1 and Trajectory Line 2, respectively. It can be seen in this figure that the hexapod is able to move between the two goal trajectory lines resulting in a sinusoidal path of motion. The red points along the hexapod's path (blue curve) indicate the moment the goal trajectory line switches position. When the hexapod reached points A, C and E the goal was set to Trajectory Line 1, and points B and D indicate the moments the goal was switched to Trajectory Line 2. This scenario is analogous to the hexapod circumventing obstacles while still moving in the general x-direction.

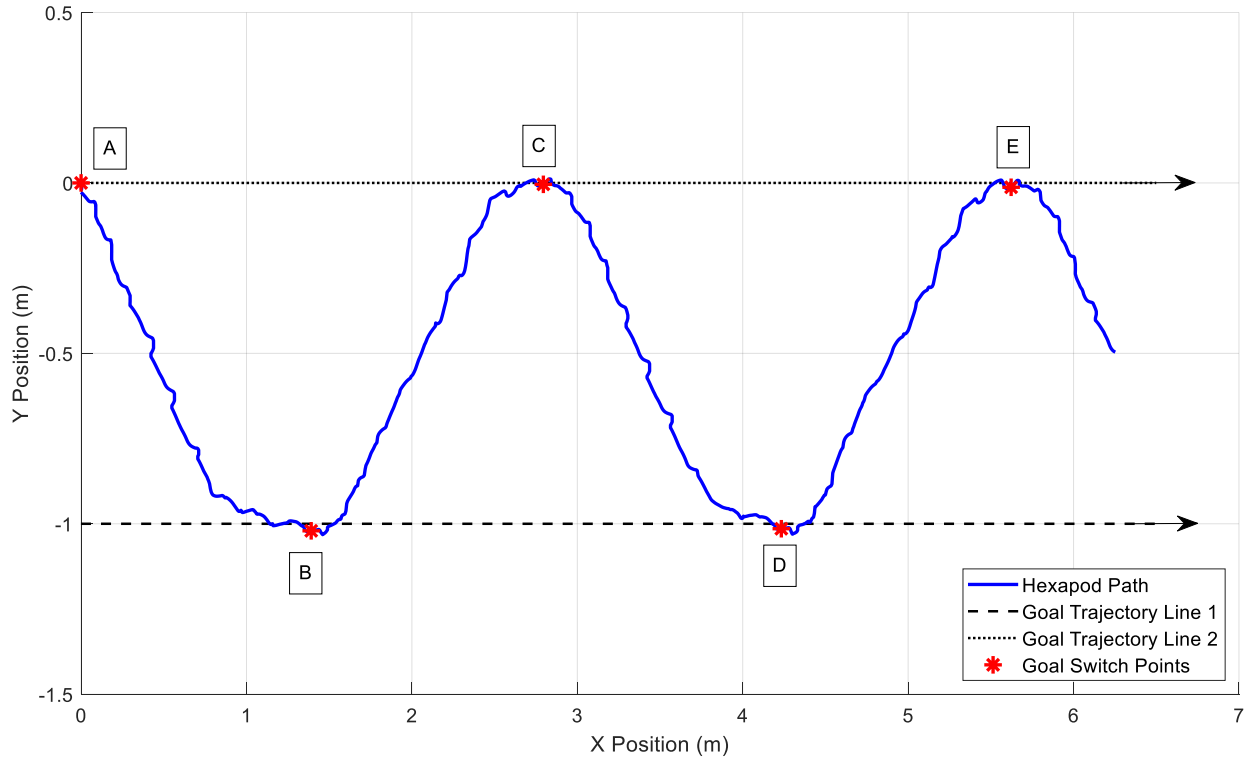



*Figure 49: Hexapod path demonstrating turning ability for the case of an alternating goal trajectory*

Figure 50 shows a second case where the path of the hexapod follows a sweeping turn which has been discretized using a series of goal trajectory lines. The hexapod starts at the origin and follows the sequence of trajectories in a counterclockwise manner. The hexapod follows the goal trajectories in alphabetical order, switching to the next trajectory at the points indicated along the blue hexapod path. The switch points are coloured to match the associated goal trajectory. From Figure 50 one can see that the first two switch points occurred near the crossover points of two goal trajectories so no major corrections in offset were required by the hexapod, just the resulting change in direction. The change in goal trajectory direction was achieved by moving the coordinate system from which the hexapod measures its heading to the reference frame of the current goal trajectory. The third and fourth trajectory changes, occurring at points d and e, take place after the hexapod has already crossed over and overshot the next goal trajectory, showcasing the hexapod's ability to correct both an instantaneous angle and offset change during continuous movement.

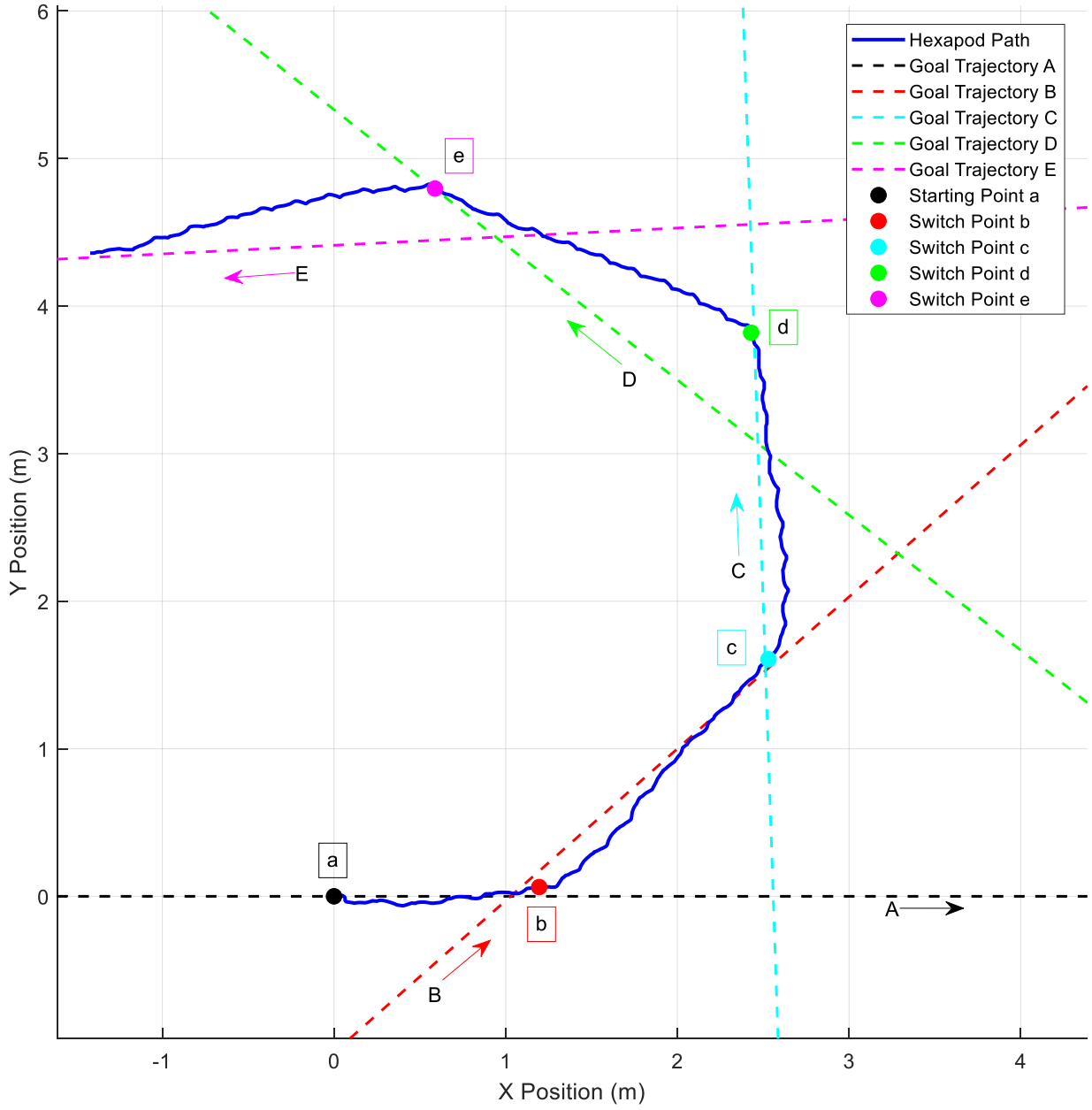



*Figure 50: Hexapod path for a sweeping turn generated by segmenting the desired path into a series of goal trajectories*

The switching between trajectories is set manually based on the simulation time for this validation test, but in a real deployment scenario the change in trajectory could be triggered automatically based on the hexapod's current position along the path. The smoothness of curves

that can be achieved depends on the update frequency of the goal trajectory and/or resolution of the curve segmentation. These validation tests demonstrate the ability for a hexapod trained using a random offset from a straight-line goal to navigate more complex movement paths using a discretization technique.

# Chapter 6: Conclusions and Recommendations

A thorough examination and survey of the current literature in the area of reinforcement learning applied to hexapod robots revealed a wide range of selected observations, without a clear method of justification for these choices. The selection of observations is important to not only maximize the hexapod performance, but also to affect other considerations such as hardware cost, form factor, and both power and computational requirements. The objectives of this thesis listed in Chapter 1 were set in order to build up a hexapod simulator and study the effect of observations on hexapod performance.

Building on the work of Wang *et al.* [23], a hexapod simulator was developed using a central pattern generator control scheme consisting of 6 coupled Hopf oscillators and associated mapping functions. The control scheme used in this work aligns with the cutting edge of central pattern generator control found in the literature, meeting the first thesis objective. The second thesis objective was met through the application of a reinforcement learning agent able to modify the hexapod gait in real-time by updating parameters of the CPG mapping functions. DDPG learning was used to train the hexapod to follow a goal trajectory line by giving the hexapod a random initial offset each training episode.

A designed experiment was carried out using the hexapod simulator to explore the effect of seven potential observations on the hexapod's walking performance: the joint torques, hexapod body linear velocities, body tilt, body angular velocities, body height from the ground plane, and two versions of ground contact at the hexapod leg tips (binary and force measurement). A quarter fraction factorial experiment was completed consisting of 32 unique runs with 10 replicants each in order to meet the third and fourth thesis objectives. The single observation (measurement) with the greatest effect on hexapod performance was found to be the body tilt. The factorial designed experiment was shown to have potential in this novel application and was able to generate a linear regression model providing insight about the seven studied observations. The optimal set of observations to use in order to maximize the final learning reward, for the specific simulation conditions outlined in this thesis, was found to include the joint torques, body velocities, body tilt, body angular velocities, and body height.

The designed experiment was shown for the hexapod case used in this thesis to a be a viable alternative to the existing methods of feature selection used in the field of machine learning. An advantage of the designed experiment method is the ability to fit a mathematical model to the resulting data which can provide more generalized insight into the relationships between the different observations and the hexapod performance. The resulting model could be used to estimate the hexapod's performance using different combinations of observations without the need for additional testing. The designed experiment also provides a defined run plan for generating the experimental data, which is helpful for application to a reinforcement learning case where there is no existing dataset before training.

Finally, to meet the fifth thesis objective, a trained reinforcement learning agent was deployed to the hexapod simulator with the additional complexity of a moving goal trajectory. The robustness and versatility of the learned behaviour was demonstrated as the hexapod was able to follow complex paths which had been discretized into a series of straight-line goal trajectories.

## 6.1. Contributions

The contributions of this work are as follows:

- The development of a MATLAB Simscape simulator for studying reinforcement learning applied to a hexapod robot, which could be used in future work to train and test algorithms before deployment to the Fire Ant hexapod platform.
- The successful application to a hexapod of a novel central pattern generator control architecture for reinforcement learning; the combination of DDPG reinforcement learning with the Hopf oscillator and mapping functions adapted from Wang *et al.* [23].
- A demonstration of the capability of a fractional factorial designed experiment as a tool to aid in the selection of observations in order to maximize performance for a particular reinforcement learning application.
- The utilization of an offset observation normalized using a sigmoid function to train a hexapod to walk along a goal trajectory line, as well as a demonstration that the learned

behaviour is robust, allowing the hexapod to follow more complex paths during deployment.

## 6.2. Future Work

Working toward the deployment of reinforcement learning to the physical Fire Ant hexapod robot platform, it is recommended that realistic sensor noise be added to the observations in simulation to determine how sensor noise affects the learning process. The complexity of the training environment for the hexapod could also be increased with the addition of uneven terrain or slopes. The robustness of the learned behaviour could also potentially be improved by randomly initializing the hexapod's orientation with respect to the goal trajectory as well as the staring offset. The scope of work in this thesis focused on lower-level path control of a hexapod; therefore, this work would need to be combined with a high-level planning algorithm which would make decisions on path planning and gait-type selection during deployment.

Although in this work the potential of a fractional factorial designed experiment was demonstrated for the selection of observation to optimize hexapod performance in a specific learning scenario, further study could be conducted to explore this potential. A smaller fraction factorial design could be tested to see if the required number of runs can be further reduced from the 32 used in this study without compromising the accuracy of the final regression model. The use of a designed experiment for observation optimization could be further studied by modifying the simulator environment to test the hexapod in a variety of scenarios, with potential examples including traversing rough terrain, climbing stairs, transporting a payload, and walking with a damaged leg. The optimal observations may differ between tasks for the same hexapod, so a designed experiment could help improve many areas of a hexapod robot's performance.

A potential future application of the designed experiment methodology used in this work is as a tool to assist with the selection of sensors for an autonomous mobile robot design that will be controlled using reinforcement learning. A designed experiment could be used in helping to quantify and compare the effect of different sensors on reinforcement learning performance, and to identify observations critical to the robot's performance for specific learning tasks.

# References


[1] J. Coelho, F. Ribeiro, B. Dias, G. Lopes, and P. Flores, “Trends in the Control of Hexapod Robots: A Survey,” *Robotics*, vol. 10, no. 3, p. 100, Aug. 2021, doi: 10.3390/robotics10030100.

[2] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, “How to train your robot with deep reinforcement learning: lessons we have learned,” *Int. J. Robot. Res.*, vol. 40, no. 4–5, pp. 698–721, Apr. 2021, doi: 10.1177/0278364920987859.

[3] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, *Feature Selection for High-Dimensional Data*. in Artificial Intelligence: Foundations, Theory, and Algorithms. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-21858-8.

[4] S. Whiteson, *Adaptive Representations for Reinforcement Learning*, vol. 291. in Studies in Computational Intelligence, vol. 291. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-13932-1.

[5] G. M. Clarke and R. E. Kempson, *Introduction to the Design and Analysis of Experiments*. London, UK: Arnold, 1997.

[6] F. Rubio, F. Valero, and C. Llopis-Albert, “A review of mobile robots: Concepts, methods, theoretical framework, and applications,” *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, p. 172988141983959, Mar. 2019, doi: 10.1177/1729881419839596.

[7] F. Delcomyn, “Neural Basis of Rhythmic Behavior in Animals,” *Science*, vol. 210, no. 4469, pp. 492–498, Oct. 1980, doi: 10.1126/science.7423199.

[8] A. J. Ijspeert, “Central pattern generators for locomotion control in animals and robots: A review,” *Neural Netw.*, vol. 21, no. 4, pp. 642–653, May 2008, doi: 10.1016/j.neunet.2008.03.014.

[9] Z. Ma, Y. Liang, and H. Tian, “Research on Gait Planning Algorithm of Quadruped Robot Based on Central Pattern Generator,” in *Proceedings of the 39th Chinese Control Conference*, Shenyang, China: IEEE Xplore, Jul. 2020, pp. 3948–3953.

[10] M. Wang, Z. Tang, B. Chen, and J. Zhang, “Locomotion control for quadruped robot based on Central Pattern Generators,” in *2016 35th Chinese Control Conference (CCC)*, Chengdu: IEEE, Jul. 2016, pp. 6335–6339. doi: 10.1109/ChiCC.2016.7554352.

[11] G. Zhong, L. Chen, Z. Jiao, J. Li, and H. Deng, “Locomotion Control and Gait Planning of a Novel Hexapod Robot Using Biomimetic Neurons,” *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 2, pp. 624–636, Mar. 2018, doi: 10.1109/TCST.2017.2692727.

[12] B. Wang, K. Zhang, X. Yang, and X. Cui, “The gait planning of hexapod robot based on CPG with feedback,” *Int. J. Adv. Robot. Syst.*, vol. 17, no. 3, p. 172988142093050, May 2020, doi: 10.1177/1729881420930503.

[13] W. Chen, G. Ren, J. Zhang, and J. Wang, “Smooth transition between different gaits of a hexapod robot via a central pattern generators algorithm,” *J. Intell. Robot. Syst.*, vol. 67, no. 3–4, pp. 255–270, Sep. 2012, doi: 10.1007/s10846-012-9661-1.

[14] R. Campos, V. Matos, and C. Santos, “Hexapod locomotion: A nonlinear dynamical systems approach,” in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, Glendale, AZ, USA: IEEE, Nov. 2010, pp. 1546–1551. doi: 10.1109/IECON.2010.5675454.

[15] C. Bal, “Neural coupled central pattern generator based smooth gait transition of a biomimetic hexapod robot,” *Neurocomputing*, vol. 420, pp. 210–226, Jan. 2021, doi: 10.1016/j.neucom.2020.07.114.

[16] D. T. Tran *et al.*, “Central pattern generator based reflexive control of quadruped walking robots using a recurrent neural network,” *Robot. Auton. Syst.*, vol. 62, no. 10, pp. 1497–1516, Oct. 2014, doi: 10.1016/j.robot.2014.05.011.

[17] H.-Y. Chung, C.-C. Hou, and S.-Y. Hsu, “Hexapod moving in complex terrains via a new adaptive CPG gait design,” *Ind. Robot Int. J.*, vol. 42, no. 2, pp. 129–141, Mar. 2015, doi: 10.1108/IR-10-2014-0403.

[18] G. Sartoretti, S. Shaw, K. Lam, N. Fan, M. Travers, and H. Choset, “Central Pattern Generator With Inertial Feedback for Stable Locomotion and Climbing in Unstructured Terrain,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD: IEEE, May 2018, pp. 5769–5775. doi: 10.1109/ICRA.2018.8461013.

[19] H. Liu, W. Jia, and L. Bi, “Hopf oscillator based adaptive locomotion control for a bionic quadruped robot,” in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, Takamatsu, Japan: IEEE, Aug. 2017, pp. 949–954. doi: 10.1109/ICMA.2017.8015944.

[20] H. Yu, H. Gao, and Z. Deng, “Enhancing adaptability with local reactive behaviors for hexapod walking robot via sensory feedback integrated central pattern generator,” *Robot. Auton. Syst.*, vol. 124, p. 103401, Feb. 2020, doi: 10.1016/j.robot.2019.103401.

[21] D. Trivun, H. Dindo, and B. Lacevic, “Resilient hexapod robot,” in *2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT)*, Sarajevo: IEEE, Oct. 2017, pp. 1–6. doi: 10.1109/ICAT.2017.8171613.

[22] J. Kon and F. Sahin, “Gait Generation for Damaged Hexapods using a Genetic Algorithm,” in *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, Budapest, Hungary: IEEE, Jun. 2020, pp. 451–456. doi: 10.1109/SoSE50414.2020.9130561.

[23] B. Wang, X. Cui, J. Sun, and Y. Gao, “Parameters optimization of central pattern generators for hexapod robot based on multi-objective genetic algorithm,” *Int. J. Adv. Robot. Syst.*, vol. 18, no. 5, p. 172988142110449, Sep. 2021, doi: 10.1177/17298814211044934.

[24] F. Borrett and M. Beckerleg, “A comparison of an evolvable hardware controller with an artificial neural network used for evolving the gait of a hexapod robot,” *Genet. Program. Evolvable Mach.*, vol. 24, no. 1, p. 5, Jun. 2023, doi: 10.1007/s10710-023-09452-4.

[25] N. Heess *et al.*, “Emergence of Locomotion Behaviours in Rich Environments,” Jul. 10, 2017, *arXiv*: arXiv:1707.02286. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/1707.02286

[26] Q. Yang, Y. Gao, and S. Li, “Terrain-adaptive Central Pattern Generators with Reinforcement Learning for Hexapod Locomotion,” Oct. 11, 2023, *arXiv*: arXiv:2310.07744. Accessed: Nov. 03, 2023. [Online]. Available: http://arxiv.org/abs/2310.07744

[27] L. Wang, R. Li, Z. Huangfu, Y. Feng, and Y. Chen, “A Soft Actor-Critic Approach for a Blind Walking Hexapod Robot with Obstacle Avoidance,” *Actuators*, vol. 12, no. 10, p. 393, Oct. 2023, doi: 10.3390/act12100393.

[28] Z. Qiu, W. Wei, and X. Liu, “Adaptive Gait Generation for Hexapod Robots Based on Reinforcement Learning and Hierarchical Framework,” *Actuators*, vol. 12, no. 2, p. 75, Feb. 2023, doi: 10.3390/act12020075.

[29] D. Li, W. Wei, and Z. Qiu, “Combined Reinforcement Learning and CPG Algorithm to Generate Terrain-Adaptive Gait of Hexapod Robots,” *Actuators*, vol. 12, no. 4, p. 157, Apr. 2023, doi: 10.3390/act12040157.

[30] R. Liu, N. Sontakke, and S. Ha, “PM-FSM: Policies Modulating Finite State Machine for Robust Quadrupedal Locomotion,” Aug. 01, 2022, *arXiv*: arXiv:2109.12696. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/2109.12696

[31] A. Issa and A. Aldair, “Learning the Quadruped Robot by Reinforcement Learning (RL),” *Iraqi J. Electr. Electron. Eng.*, vol. 18, no. 2, pp. 117–126, Dec. 2022, doi: 10.37917/ijeee.18.2.15.

[32] M. Zeng, Y. Ma, Z. Wang, and Q. Li, “Gait Self-learning for Damaged Robots Combining Bionic Inspiration and Deep Reinforcement Learning,” in *2021 40th Chinese Control Conference (CCC)*, Jul. 2021, pp. 3978–3983. doi: 10.23919/CCC52363.2021.9549968.

[33] H. Sun, T. Fu, Y. Ling, and C. He, “Adaptive Quadruped Balance Control for Dynamic Environments Using Maximum-Entropy Reinforcement Learning,” *Sensors*, vol. 21, no. 17, p. 5907, Sep. 2021, doi: 10.3390/s21175907.

[34] W. Ouyang, H. Chi, J. Pang, W. Liang, and Q. Ren, “Adaptive Locomotion Control of a Hexapod Robot via Bio-Inspired Learning,” *Front. Neurorobotics*, vol. 15, p. 627157, Jan. 2021, doi: 10.3389/fnbot.2021.627157.

[35] K. Naya, K. Kutsuzawa, D. Owaki, and M. Hayashibe, “Spiking Neural Network Discovers Energy-Efficient Hexapod Motion in Deep Reinforcement Learning,” *IEEE Access*, vol. 9, pp. 150345–150354, 2021, doi: 10.1109/ACCESS.2021.3126311.

[36] A. Kumar, Z. Fu, D. Pathak, and J. Malik, “RMA: Rapid Motor Adaptation for Legged Robots,” Jul. 08, 2021, *arXiv*: arXiv:2107.04034. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/2107.04034

[37] Y. Kim, B. Son, and D. Lee, “Learning multiple gaits of quadruped robot using hierarchical reinforcement learning,” Dec. 09, 2021, *arXiv*: arXiv:2112.04741. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/2112.04741

[38] H. Hu and Y. Liu, “Blind Adaptive Gait Planning on Non-stationary Environments via Continual Reinforcement Learning,” in *2021 IEEE International Conference on Unmanned Systems (ICUS)*, Oct. 2021, pp. 280–284. doi: 10.1109/ICUS52573.2021.9641095.

[39] T. Anne, J. Wilkinson, and Z. Li, “Meta-Reinforcement Learning for Adaptive Motor Control in Changing Robot Dynamics and Environments,” Jan. 19, 2021, *arXiv*: arXiv:2101.07599. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/2101.07599

[40] S. Verma, H. S. Nair, G. Agarwal, J. Dhar, and A. Shukla, “Deep Reinforcement Learning for Single-Shot Diagnosis and Adaptation in Damaged Robots,” in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, Hyderabad India: ACM, Jan. 2020, pp. 82–89. doi: 10.1145/3371158.3371168.

[41] M. Schilling, K. Konen, F. W. Ohl, and T. Korthals, “Decentralized Deep Reinforcement Learning for a Distributed and Adaptive Locomotion Controller of a Hexapod Robot,” May 21, 2020, *arXiv*: arXiv:2005.11164. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/2005.11164

[42] D. Jain, A. Iscen, and K. Caluwaerts, “Hierarchical Reinforcement Learning for Quadruped Locomotion,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, China: IEEE, Nov. 2019, pp. 7551–7557. doi: 10.1109/IROS40897.2019.8967913.

[43] D. Jain, A. Iscen, and K. Caluwaerts, “From Pixels to Legs: Hierarchical Learning of Quadruped Locomotion,” Nov. 23, 2020, *arXiv*: arXiv:2011.11722. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/2011.11722

[44] T. Azayev and K. Zimmerman, “Blind Hexapod Locomotion in Complex Terrain with Gait Adaptation Using Deep Reinforcement Learning and Classification,” *J. Intell. Robot. Syst.*, vol. 99, no. 3–4, pp. 659–671, Sep. 2020, doi: 10.1007/s10846-020-01162-8.

[45] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, “Data Efficient Reinforcement Learning for Legged Robots,” Oct. 06, 2019, *arXiv*: arXiv:1907.03613. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/1907.03613

[46] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, “Learning to Walk via Deep Reinforcement Learning,” Jun. 19, 2019, *arXiv*: arXiv:1812.11103. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/1812.11103

[47] A. Iscen *et al.*, “Policies Modulating Trajectory Generators,” p. 11, 2018.

[48] “Quadruped Robot Locomotion Using DDPG Agent - MATLAB & Simulink.” Accessed: Oct. 05, 2023. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/quadruped-robot-locomotion-using-ddpg-gent.html

[49] “Fire Ant Assembly Guide.” Orion Robotics Inc., 2012.

[50] J. Woodacre, “MODEL-PREDICTIVE CONTROL OF A HYDRAULIC ACTIVE HEAVE COMPENSATION SYSTEM WITH HEAVE PREDICTION,” Dalhousie University, Halifax, Nova Scotia, 2015.

[51] “Model spatial contact between two geometries - MATLAB.” Accessed: Oct. 24, 2023. [Online]. Available: https://www.mathworks.com/help/sm/ref/spatialcontactforce.html

[52] “Convert Simulink input signal into physical signal - MATLAB.” Accessed: Nov. 06, 2023. [Online]. Available: https://www.mathworks.com/help/simscape/ref/simulinkpsconverter.html

[53] C. Deng, S. Wang, Z. Chen, J. Wang, L. Ma, and J. Li, “CPG-Inspired Gait Generation and Transition Control for Six Wheel-legged Robot,” in *2021 China Automation Congress (CAC)*, Oct. 2021, pp. 2310–2315. doi: 10.1109/CAC53003.2021.9727252.

[54] “Reinforcement learning agent - Simulink.” Accessed: Jul. 03, 2024. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ref/rlagent.html

[55] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” Jul. 05, 2019, *arXiv*: arXiv:1509.02971. Accessed: Sep. 22, 2022. [Online]. Available: http://arxiv.org/abs/1509.02971

[56] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Jan. 29, 2017, *arXiv*: arXiv:1412.6980. Accessed: Nov. 07, 2023. [Online]. Available: http://arxiv.org/abs/1412.6980

[57] NIST/SEMATECH, “e-Handbook of Statistical Methods”, doi: https://doi.org/10.18434/M32189.

[58] P. Goos and B. Jones, *Optimal Design of Experiments: A Case Study Approach*. Hoboken, NJ: John Wiley & Sons, Inc., 2011.

[59] *Data Analysis, Statistical & Process Improvement Tools | Minitab*. Accessed: Nov. 07, 2023. [Online]. Available: https://www.minitab.com/en-us/

[60] I. Pardoe, L. Simon, and D. Young, “STAT 501: Regression Methods,” The Pennsylvania State University. Accessed: Jul. 02, 2024. [Online]. Available: https://online.stat.psu.edu/stat501/

[61] “Coefficient of Determination (R-Squared) - MATLAB & Simulink.” Accessed: Jul. 04, 2024. [Online]. Available: https://www.mathworks.com/help/stats/coefficient-of-determination-r-squared.html

[62] “Model summary table for Analyze Factorial Design.” Accessed: Jul. 03, 2024. [Online]. Available: https://support.minitab.com/en-us/minitab/help-and-how-to/statistical-modeling/doe/how-to/factorial/analyze-factorial-design/interpret-the-results/all-statistics-and-graphs/model-summary-table/#r-sq

[63] P. G. Mathews, *Design of experiments with MINITAB*. Milwaukee, Wis: ASQ Quality Press, 2005.

# Appendix A: Designed Experiment Alias Structure

The following table lists the aliased or confounded terms of the fractional factorial designed experiment used in the thesis. Because it is a quarter fraction design, each of the 32 terms in the initial model is confounded with three other higher order interaction terms (aliased groups of four).

| |
|---|
| I + CEFG + ABCDF + ABDEG |
| A + BCDF + BDEG + ACEFG |
| B + ACDF + ADEG + BCEFG |
| C + EFG + ABDF + ABCDEG |
| D + ABCF + ABEG + CDEFG |
| E + CFG + ABDG + ABCDEF |
| F + CEG + ABCD + ABDEFG |
| G + CEF + ABDE + ABCDFG |
| AB + CDF + DEG + ABCEFG |
| AC + BDF + AEFG + BCDEG |
| AD + BCF + BEG + ACDEFG |
| AE + BDG + ACFG + BCDEF |
| AF + BCD + ACEG + BDEFG |
| AG + BDE + ACEF + BCDFG |
| BC + ADF + BEFG + ACDEG |
| BD + ACF + AEG + BCDEFG |
| BE + ADG + BCFG + ACDEF |

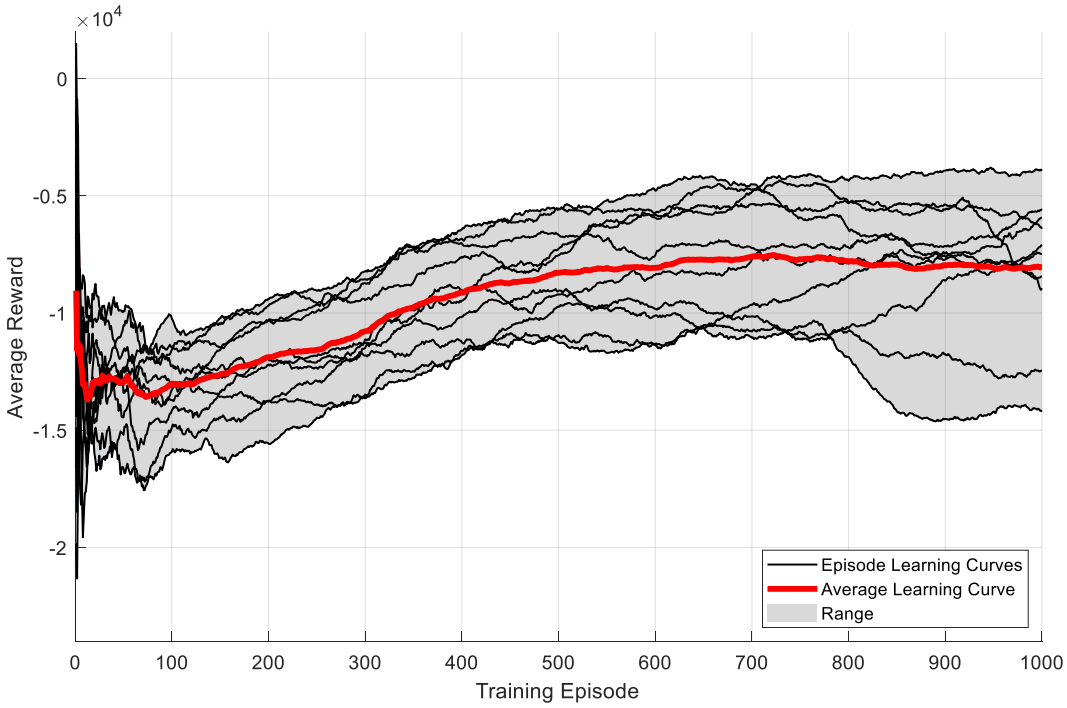
| |
|---|
| BF + ACD + BCEG + ADEFG |
| BG + ADE + BCEF + ACDFG |
| CD + ABF + DEFG + ABCEG |
| CE + FG + ABCDG + ABDEF |
| CF + EG + ABD + ABCDEFG |
| CG + EF + ABCDE + ABDFG |
| DE + ABG + CDFG + ABCEF |
| DF + ABC + CDEG + ABEFG |
| DG + ABE + CDEF + ABCFG |
| ACE + AFG + BCDG + BDEF |
| ACG + AEF + BCDE + BDFG |
| BCE + BFG + ACDG + ADEF |
| BCG + BEF + ACDE + ADFG |
| CDE + DFG + ABCG + ABEF |
| CDG + DEF + ABCE + ABFG |
| |

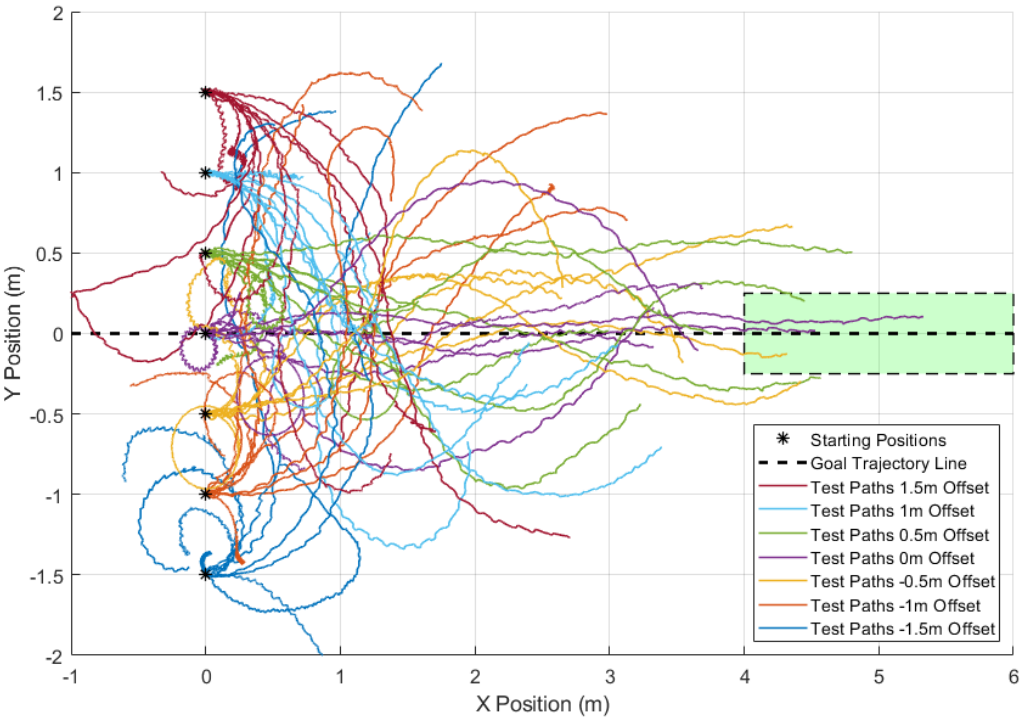
A = joint torques
B = body velocities (linear)
C = body tilt (orientation)
D = body angular velocities
E = body height
F = ground contact binary
G = ground contact forces
I = constant term

# Appendix B: Designed Experiment Result Plots

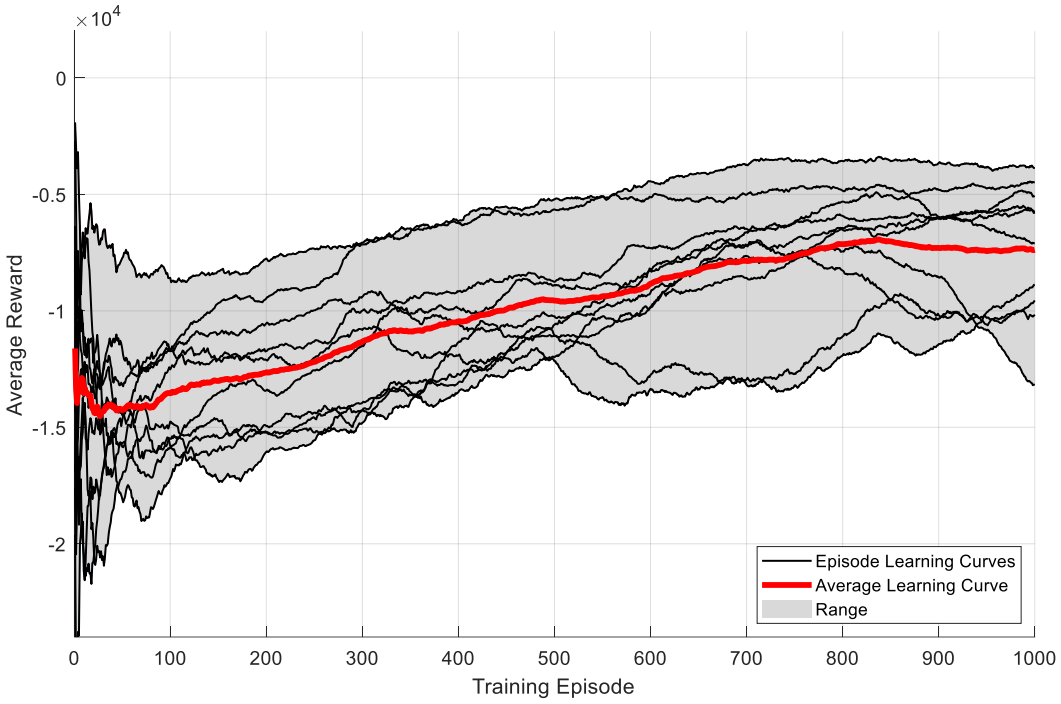
- This appendix contains two plots for each of the 32 runs from the designed experiment.
- The first plot for each run shows the learning curves for each of the ten replicants (black lines) as well as the average learning curve across all replicants shown in red.
- The second plot shows the test paths for all ten replicants of the run at the seven different starting offsets, as well as illustrating the "success area" set by the performance conditions described in Section 5.4 (shaded in green).
- The plots in this appendix demonstrate the variety of results obtained during the designed experiment.

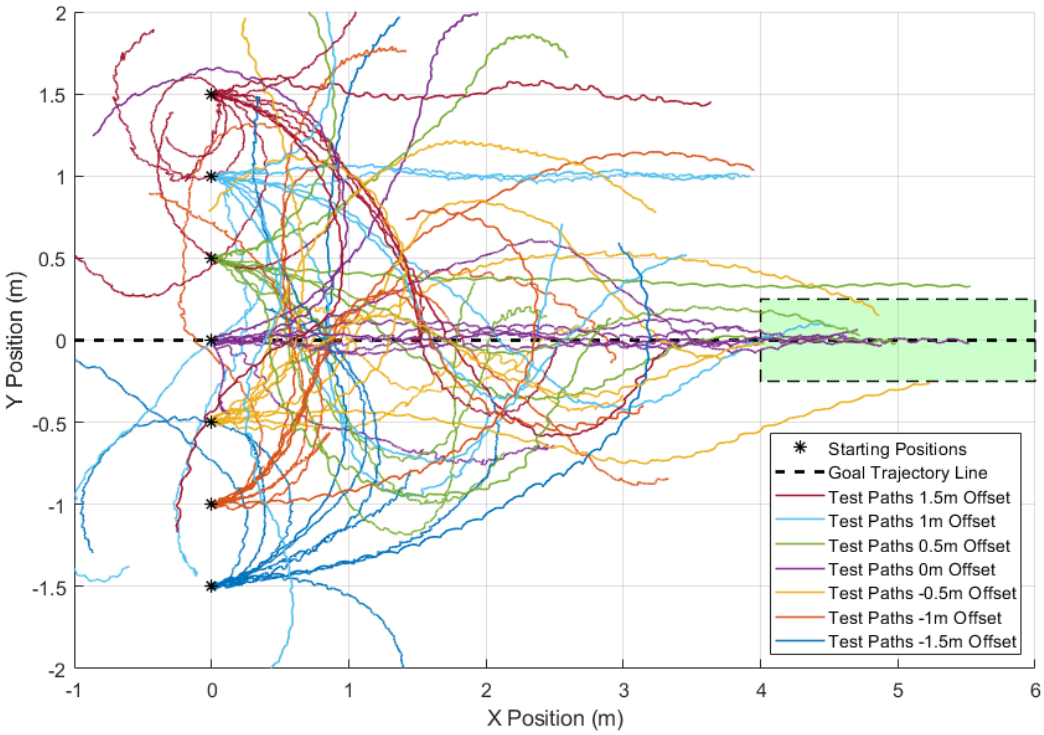
Run 1


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

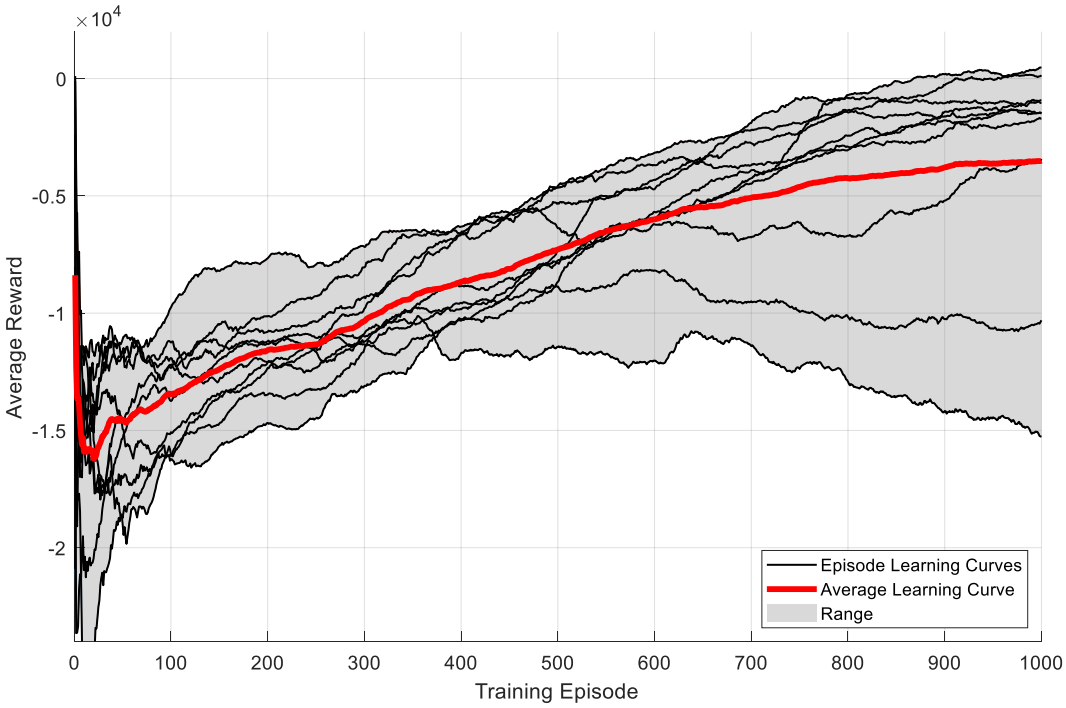

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

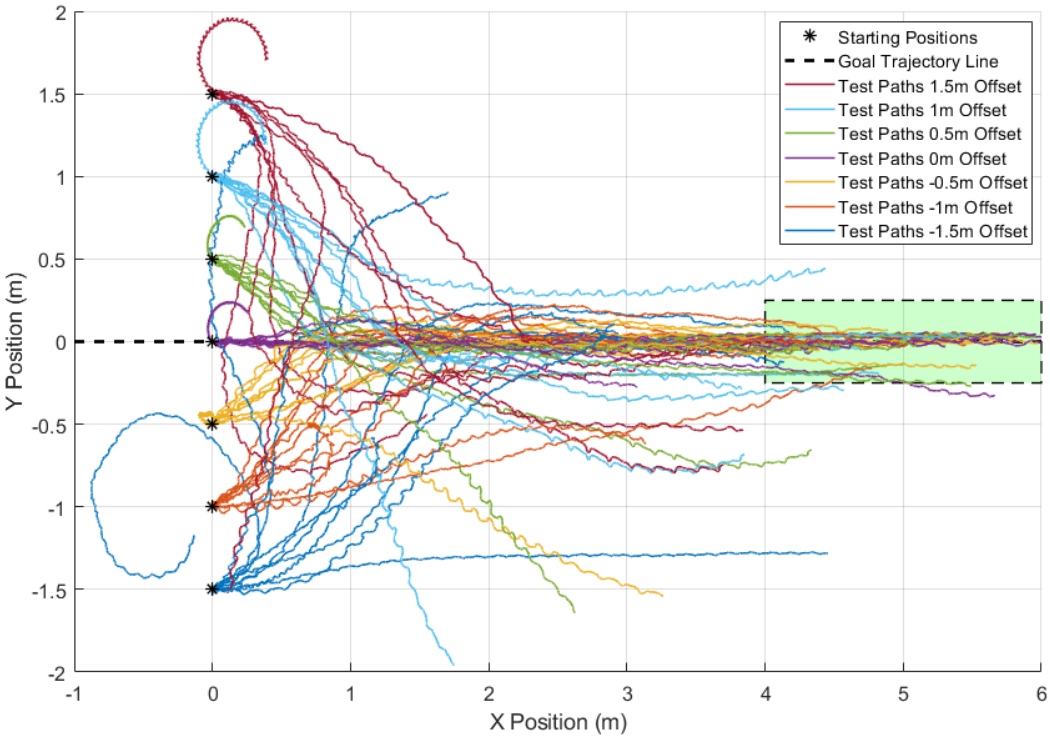
Run 2


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

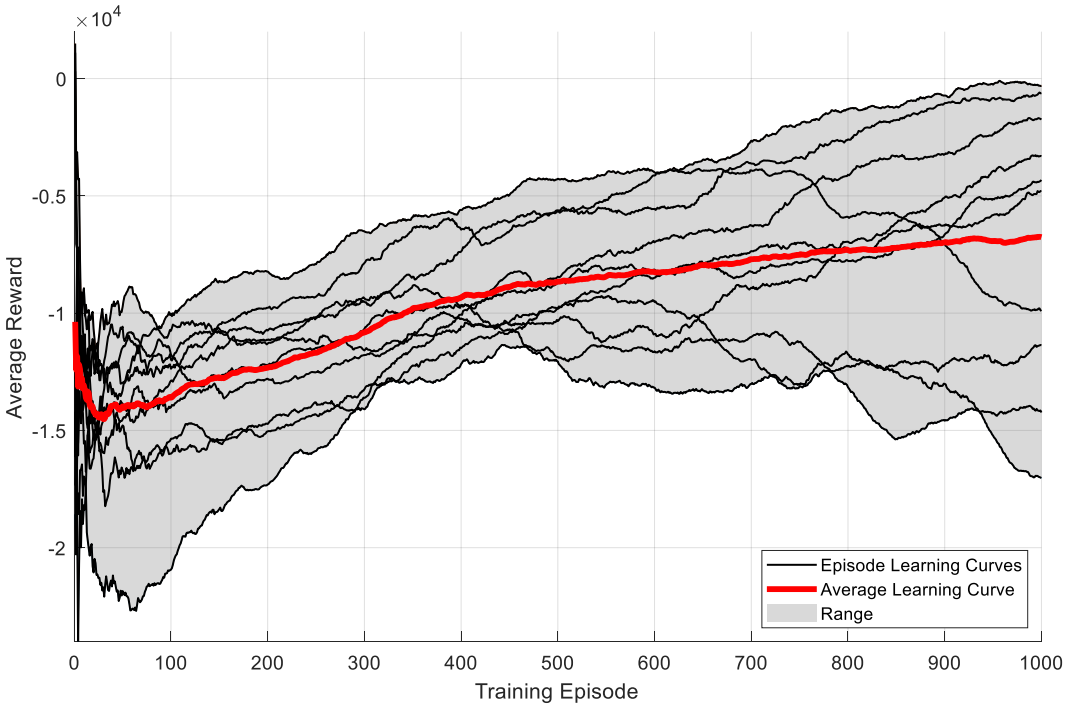

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

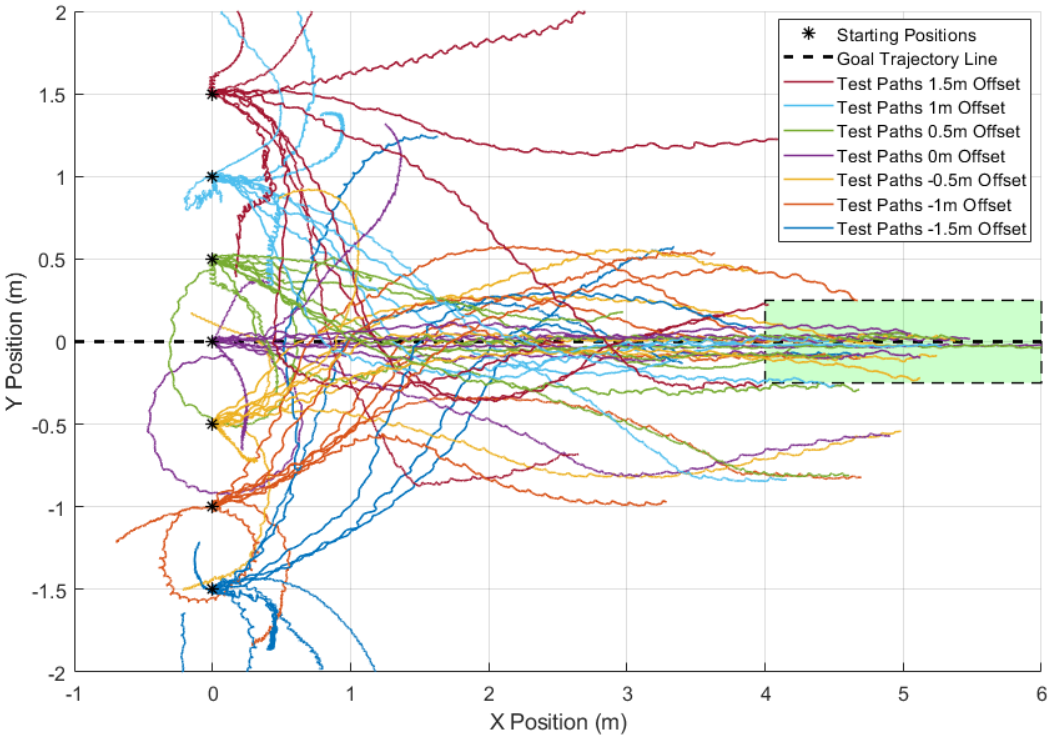
Run 3


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

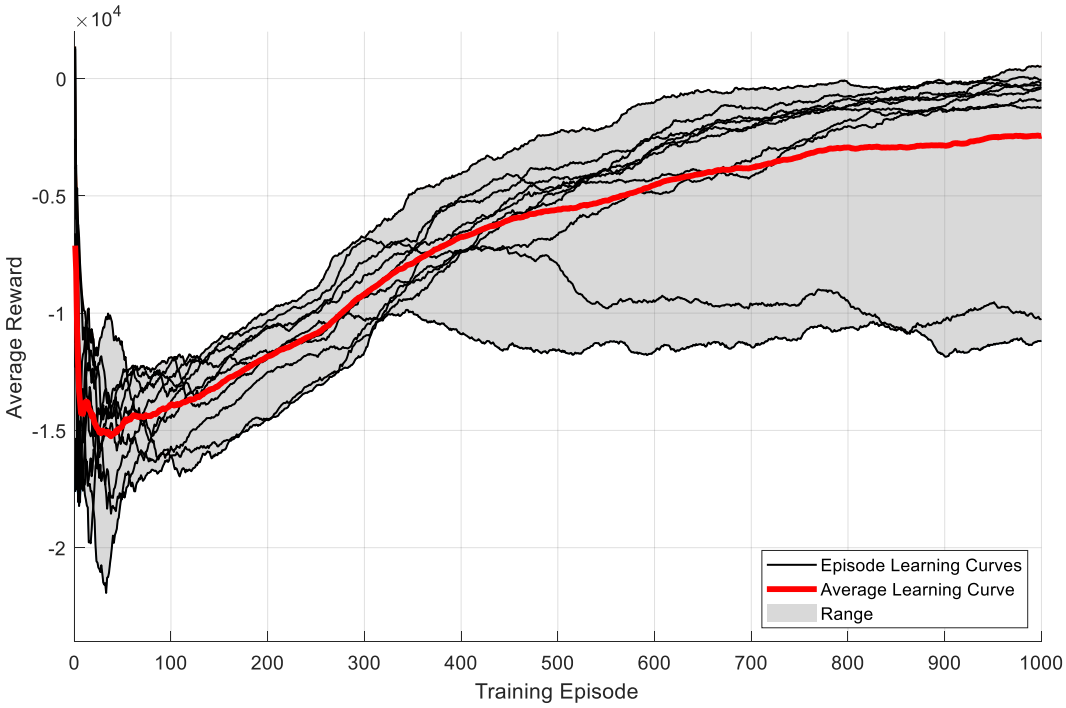

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

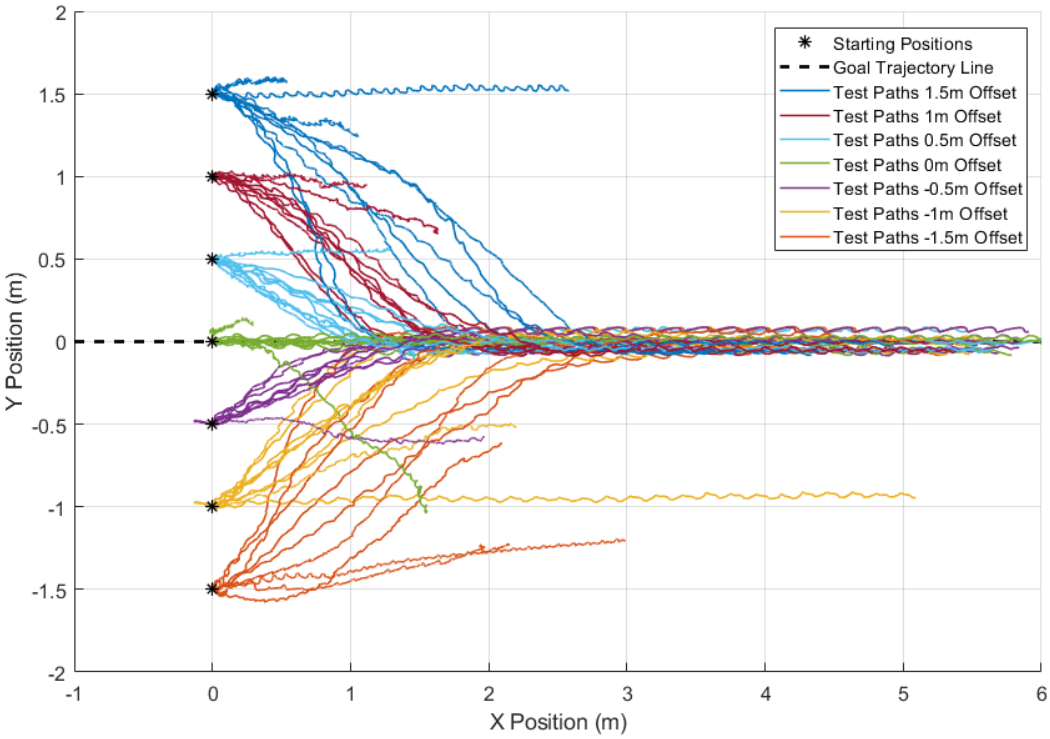
Run 4


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

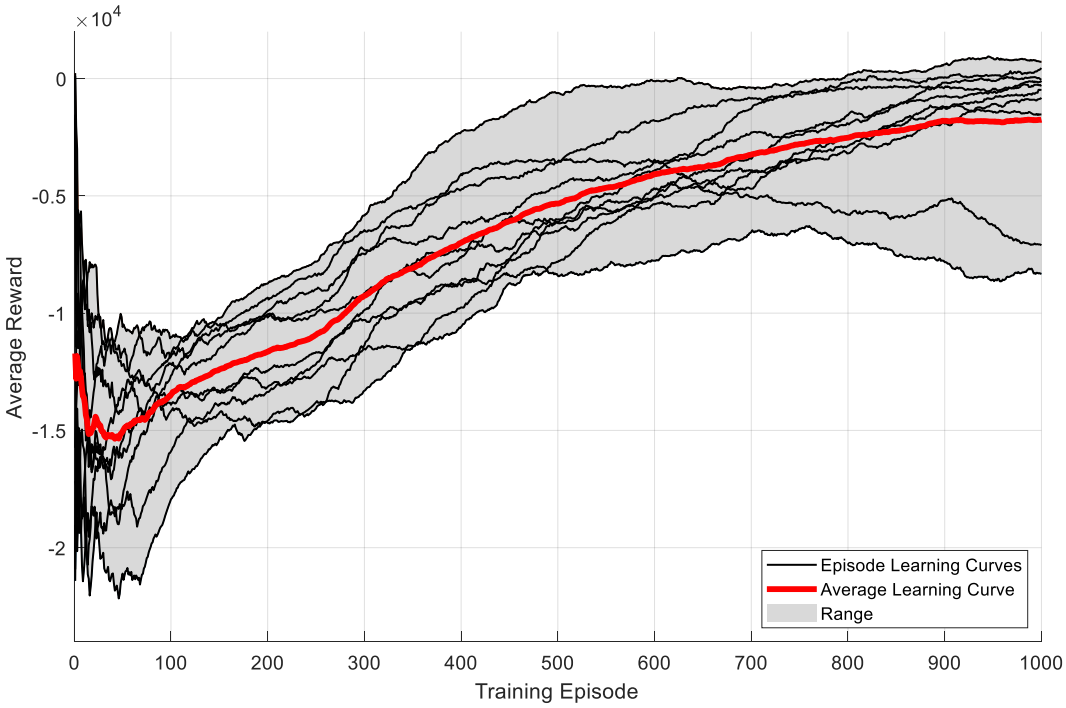

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

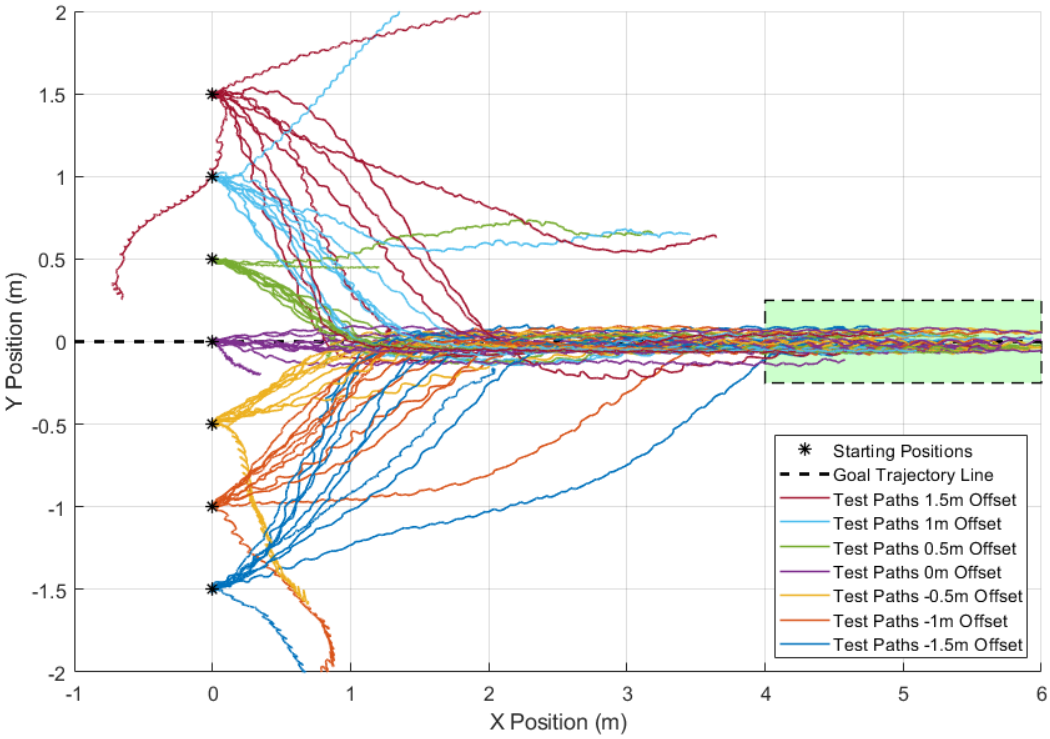
Run 5


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

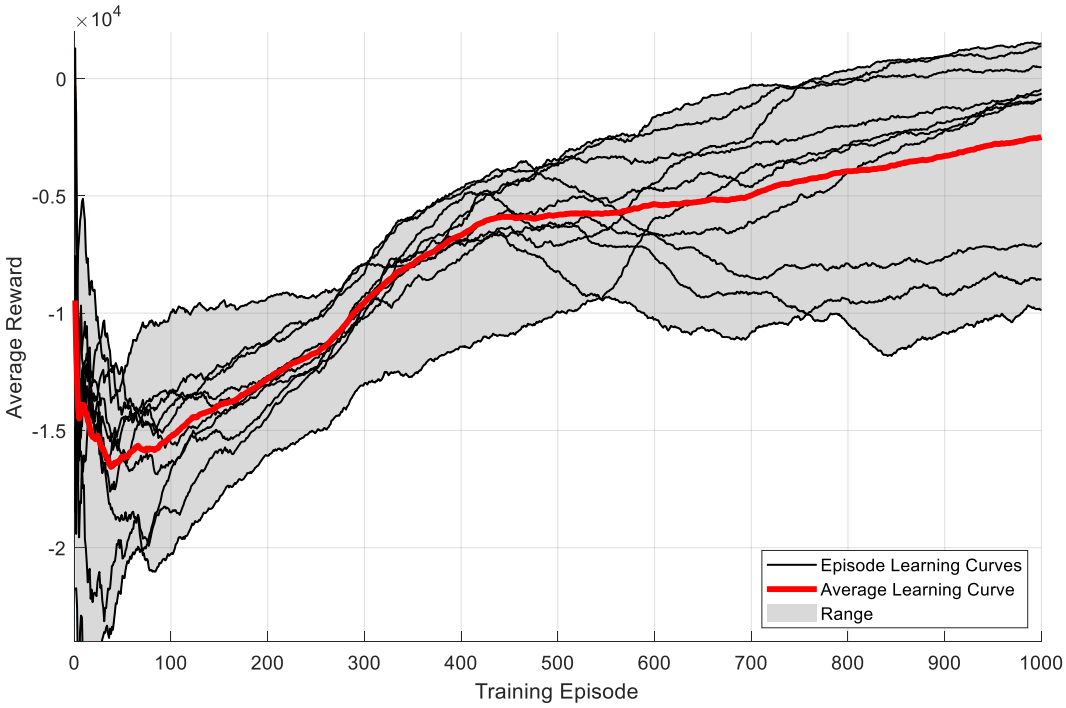

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

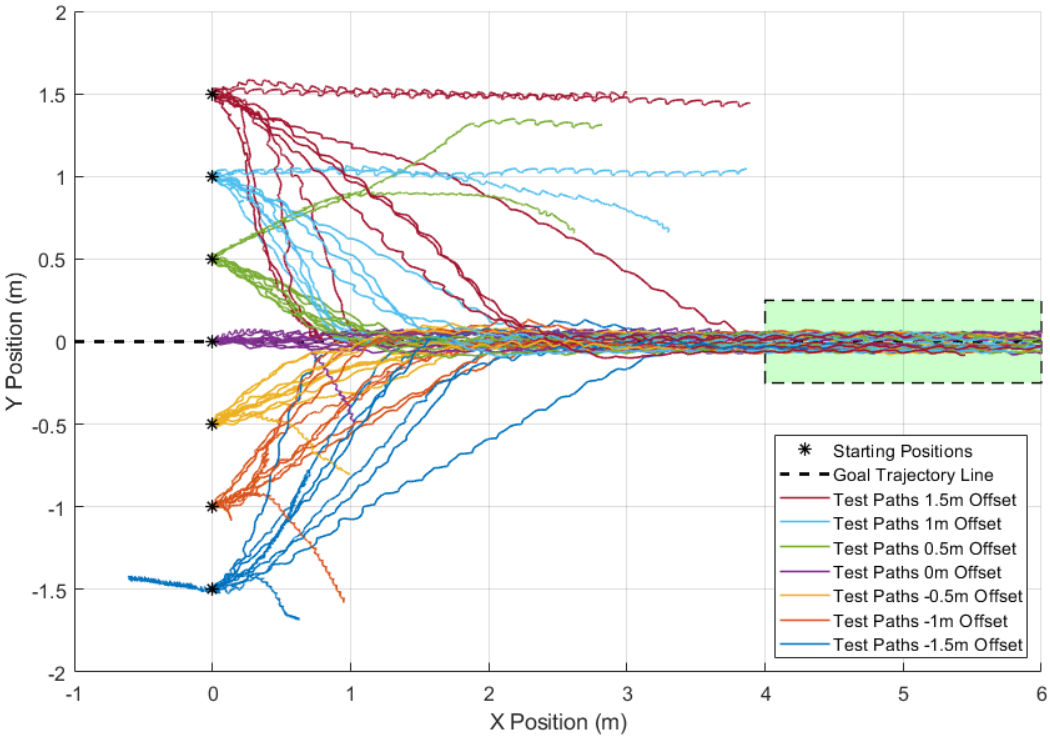
Run 6


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

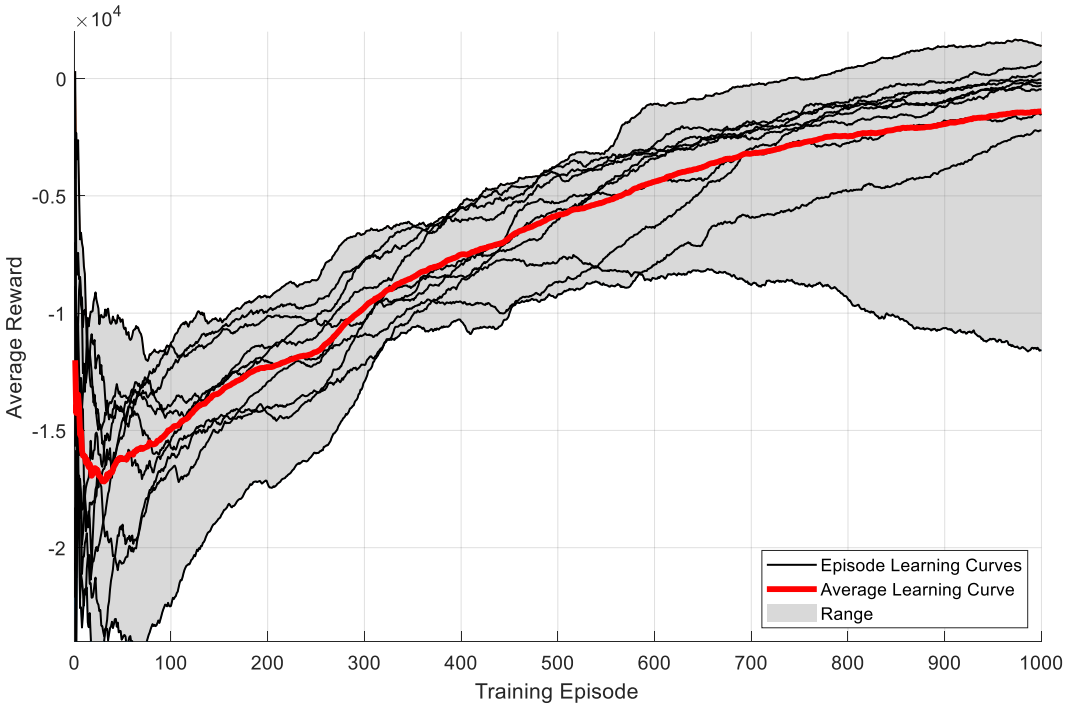

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

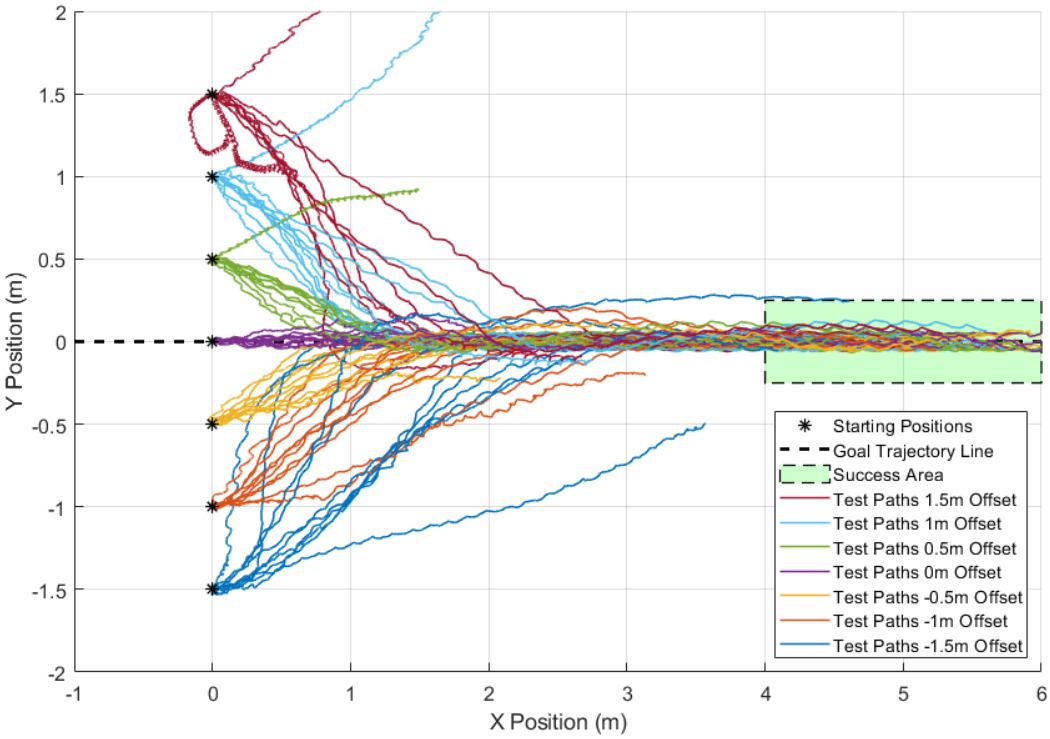
Run 7


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

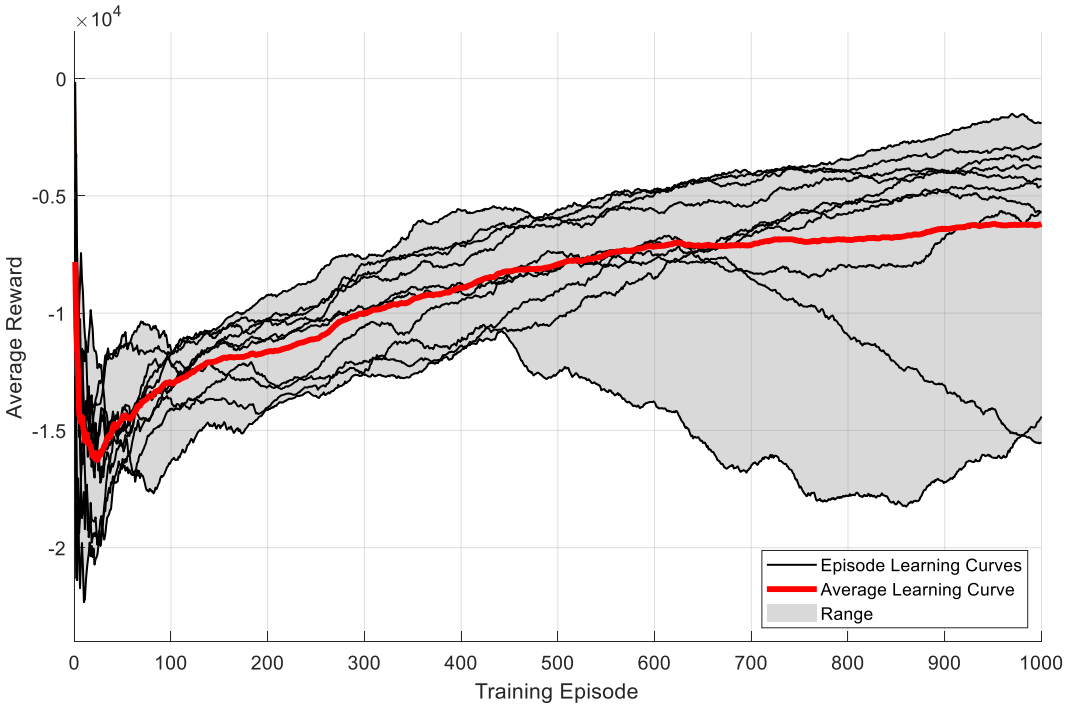

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

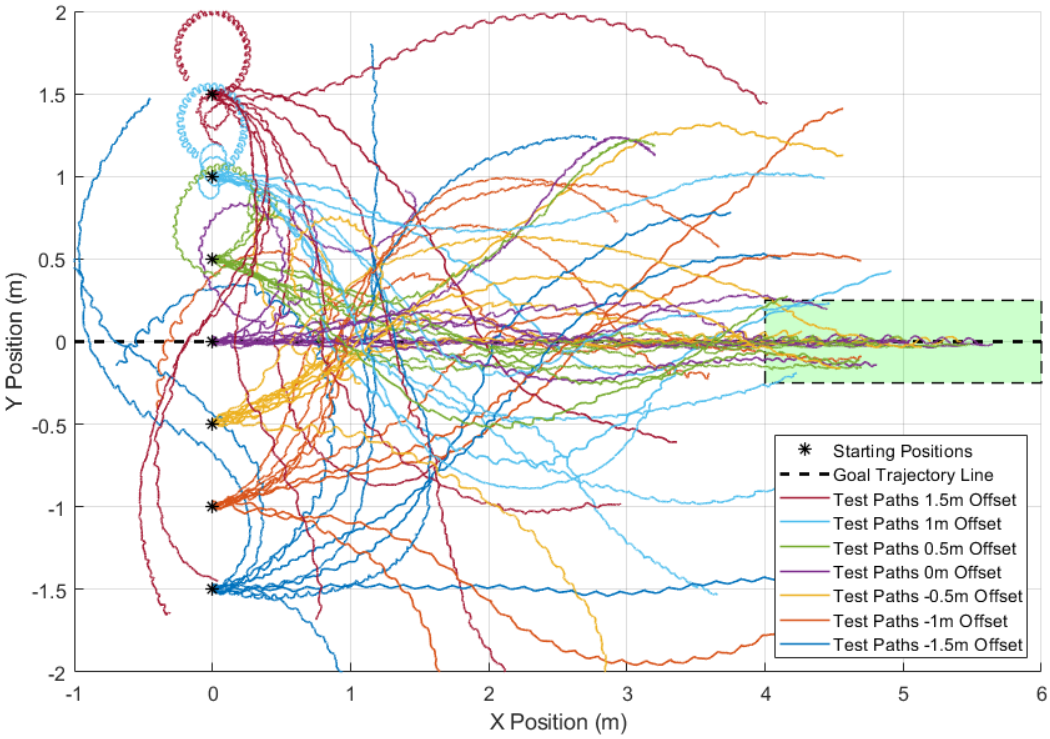
Run 8


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

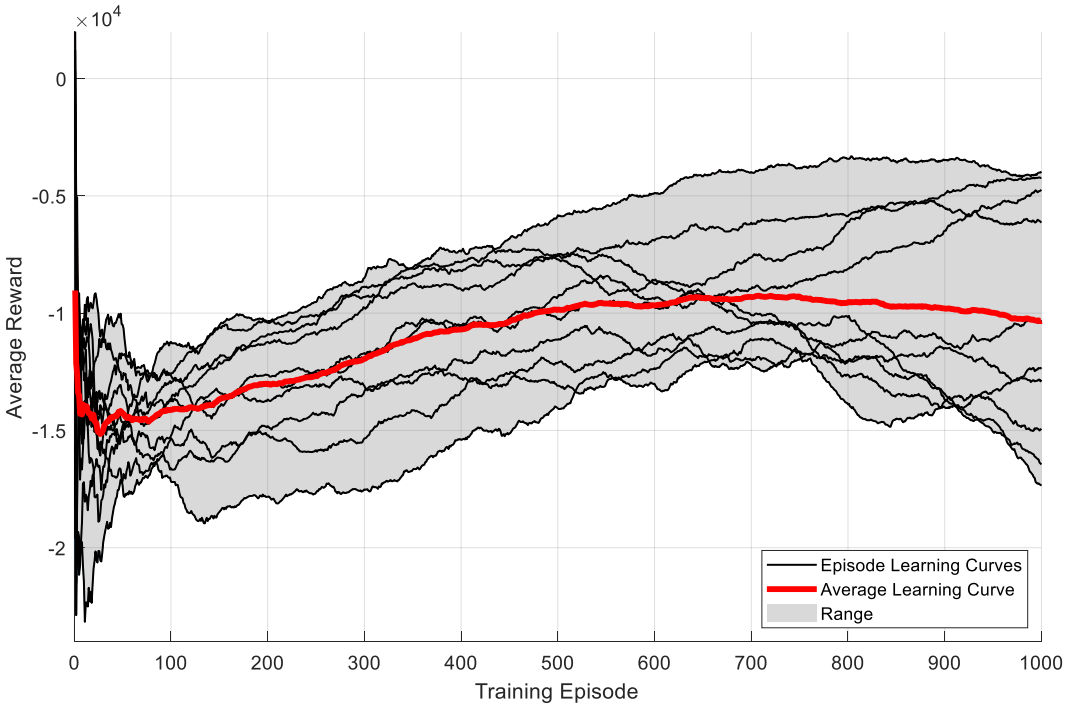

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Success Area
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

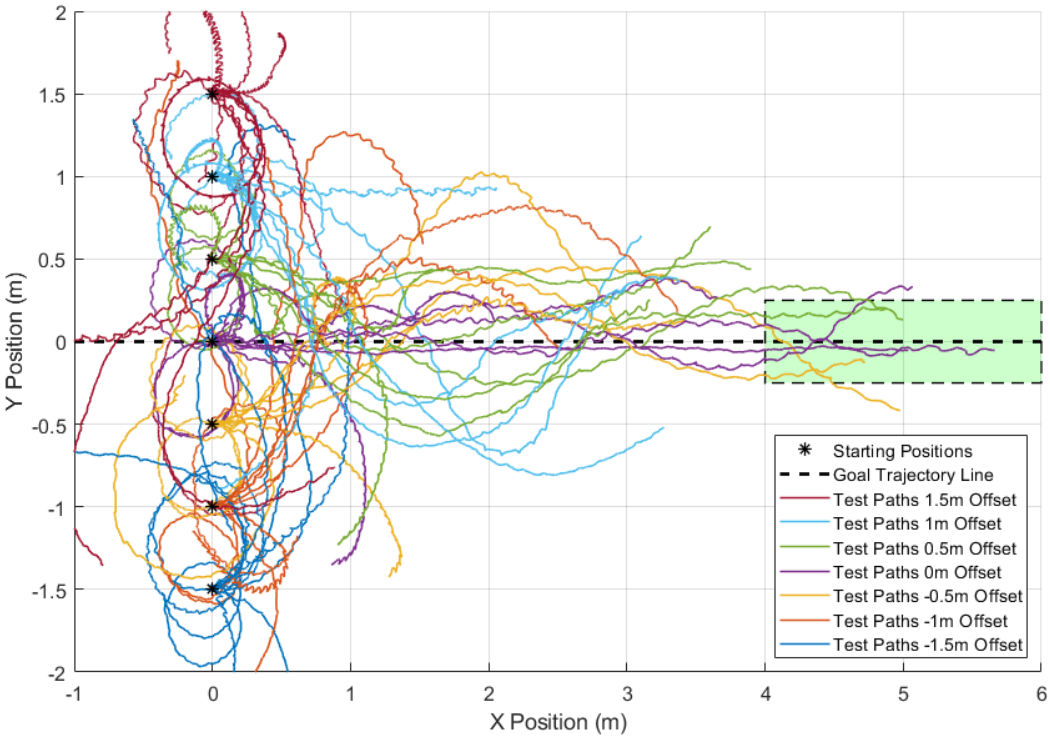
Run 9


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

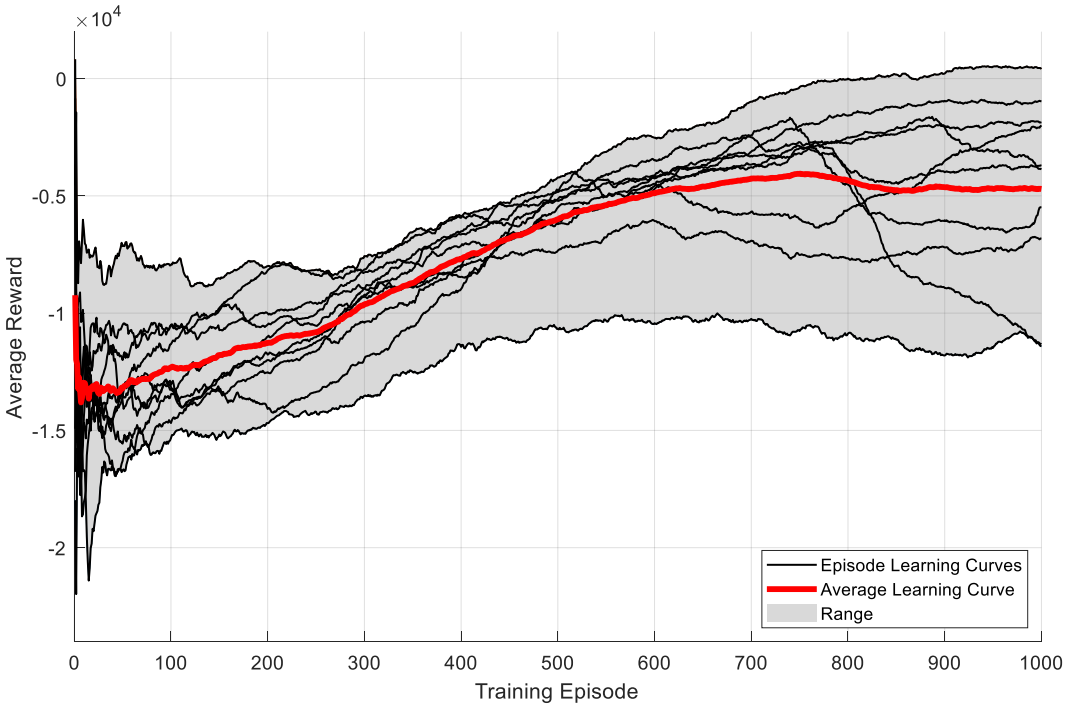

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

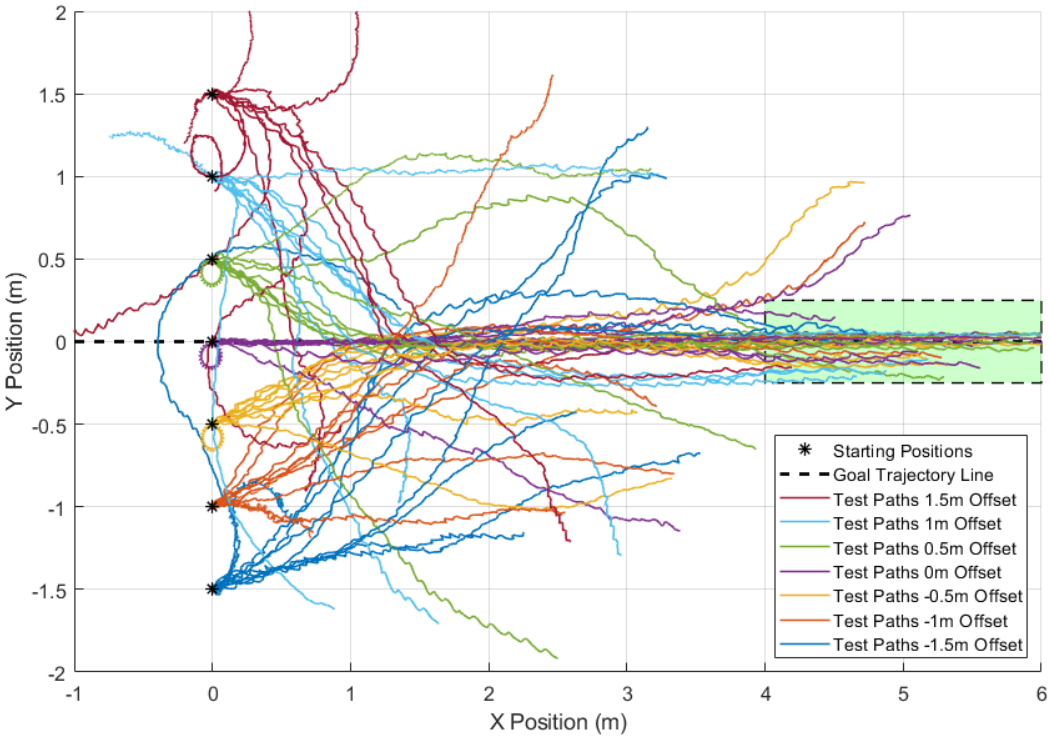
Run 10


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

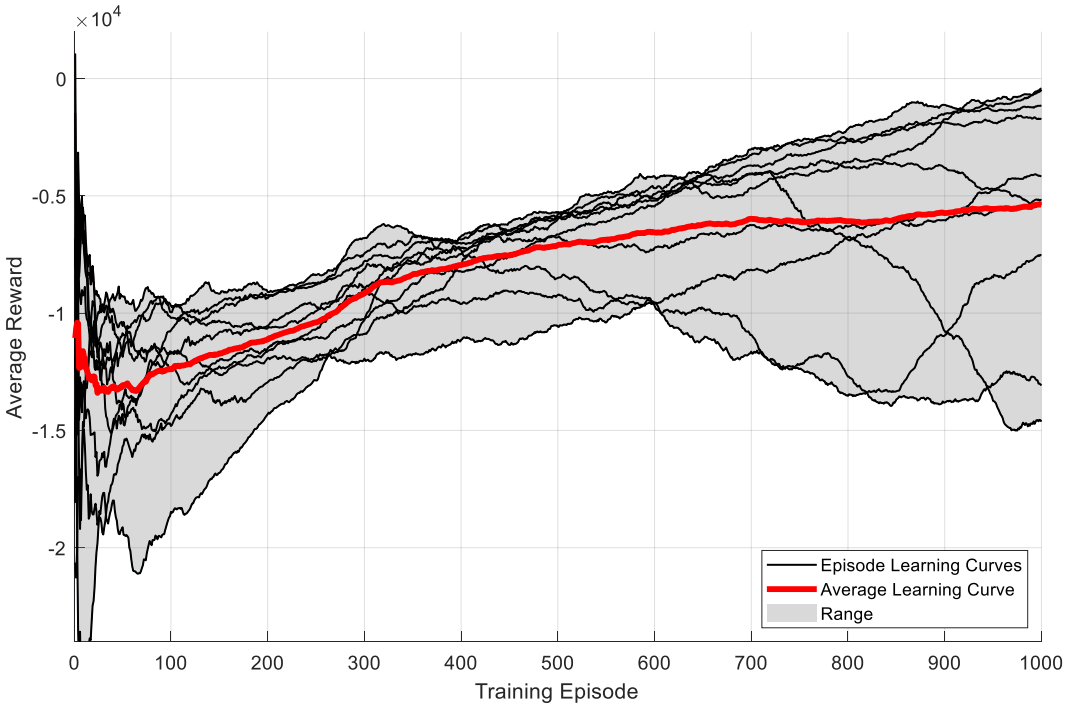

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

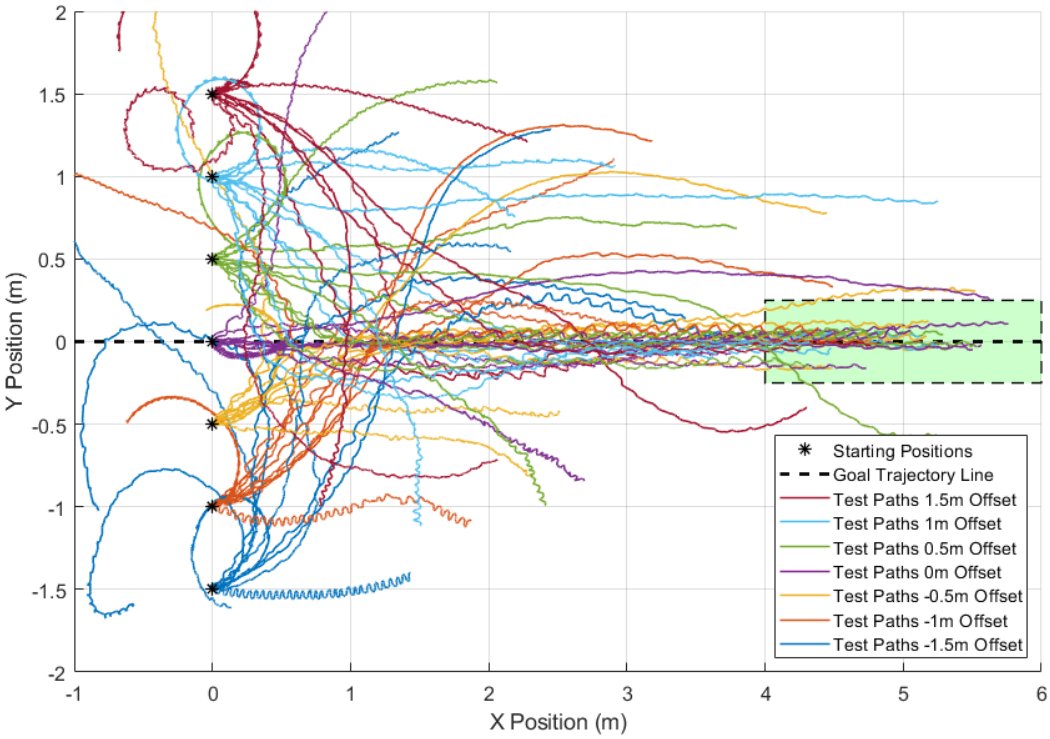
Run 11


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

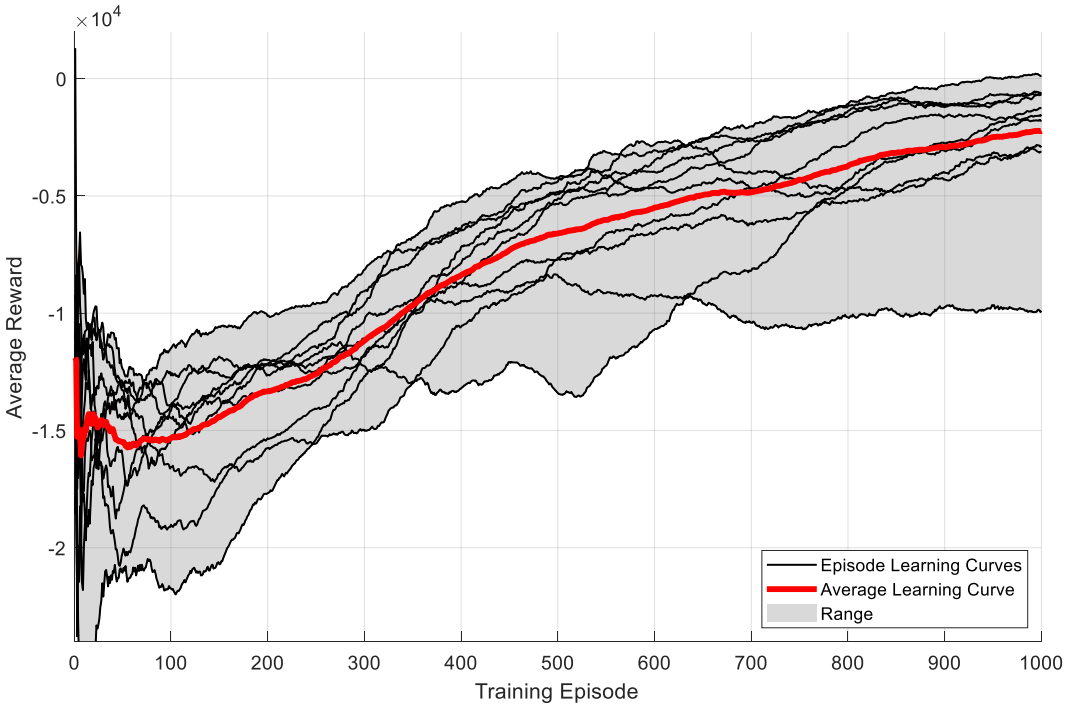

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

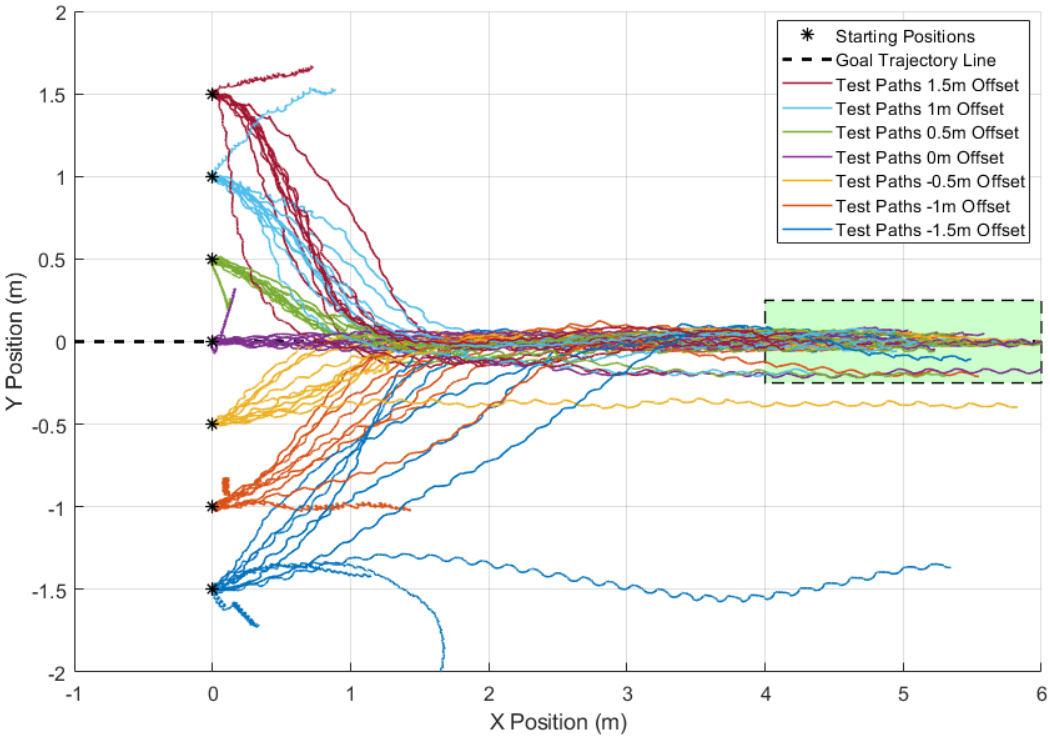
Run 12


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

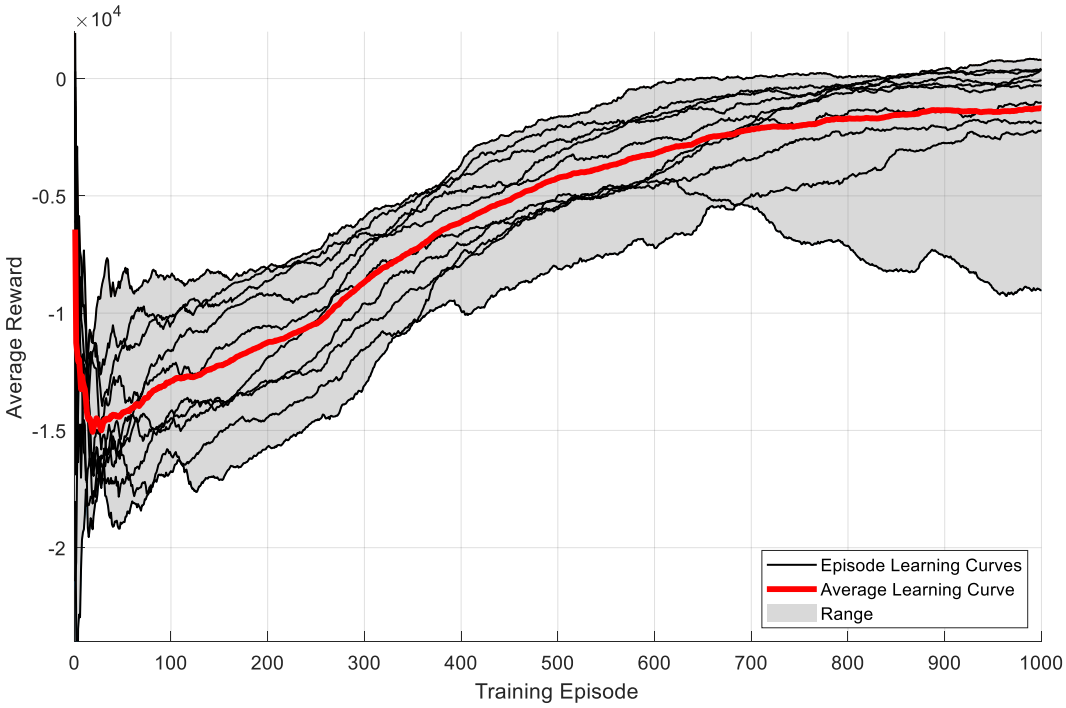

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

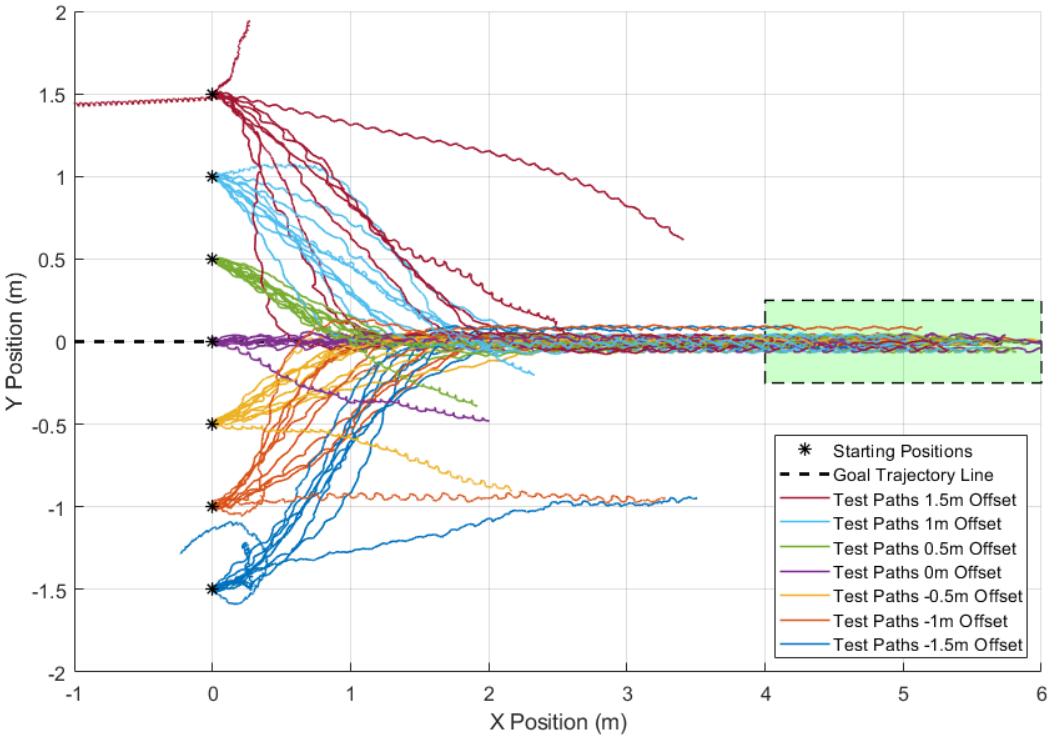
Run 13


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

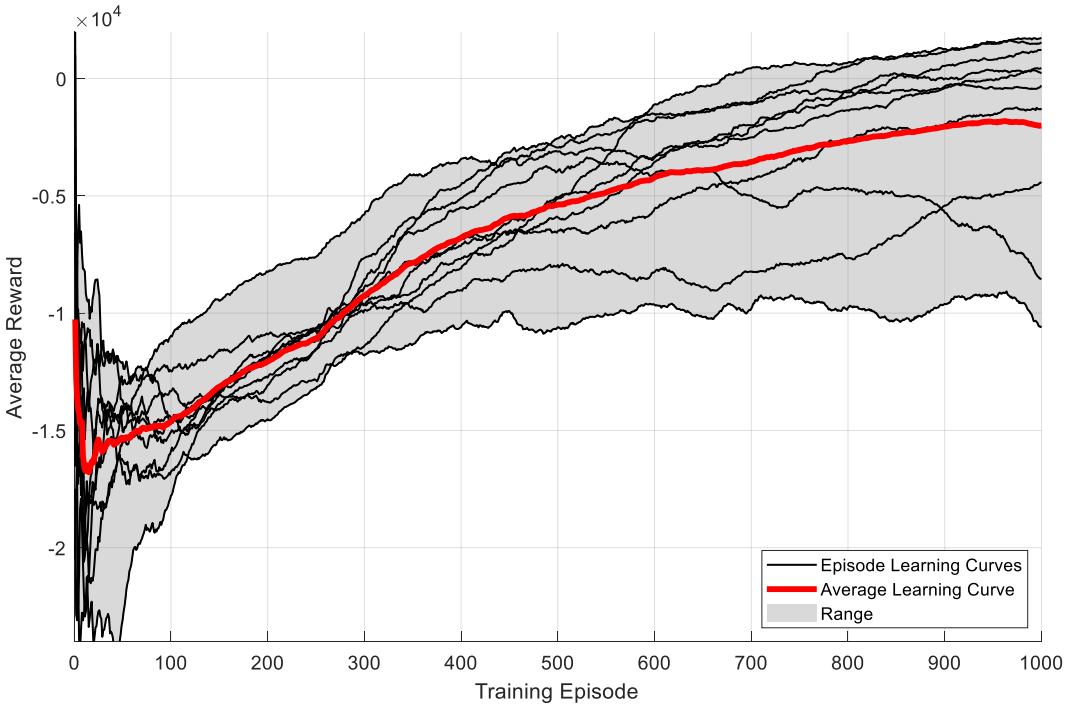

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

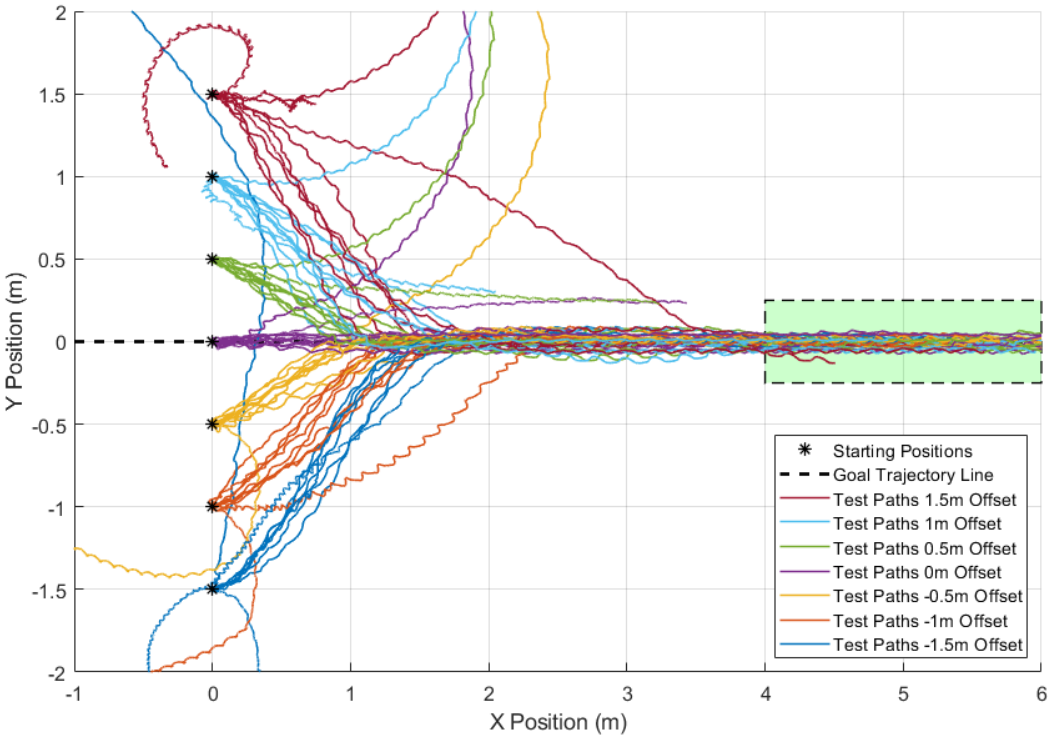
Run 14


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

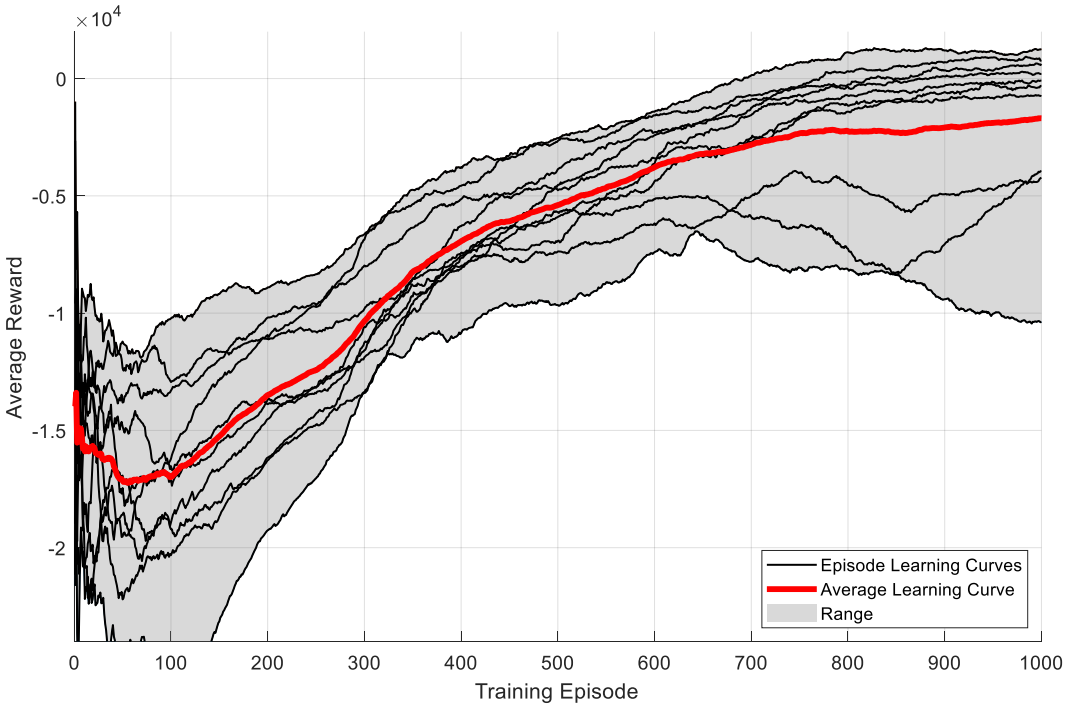

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

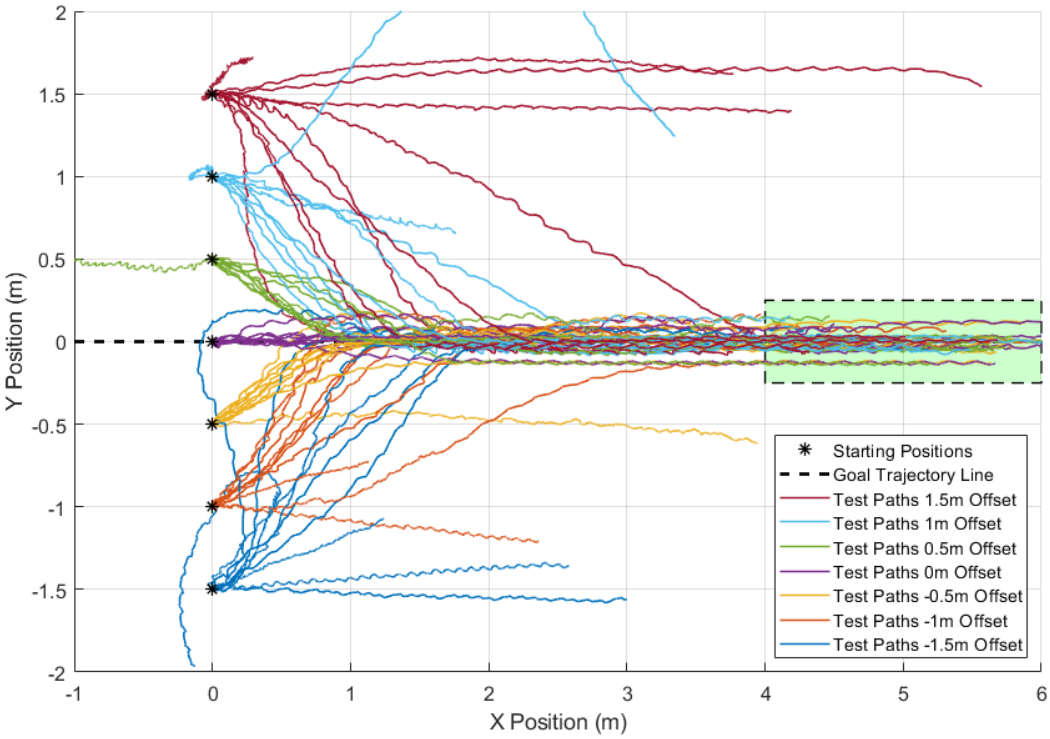
Run 15


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

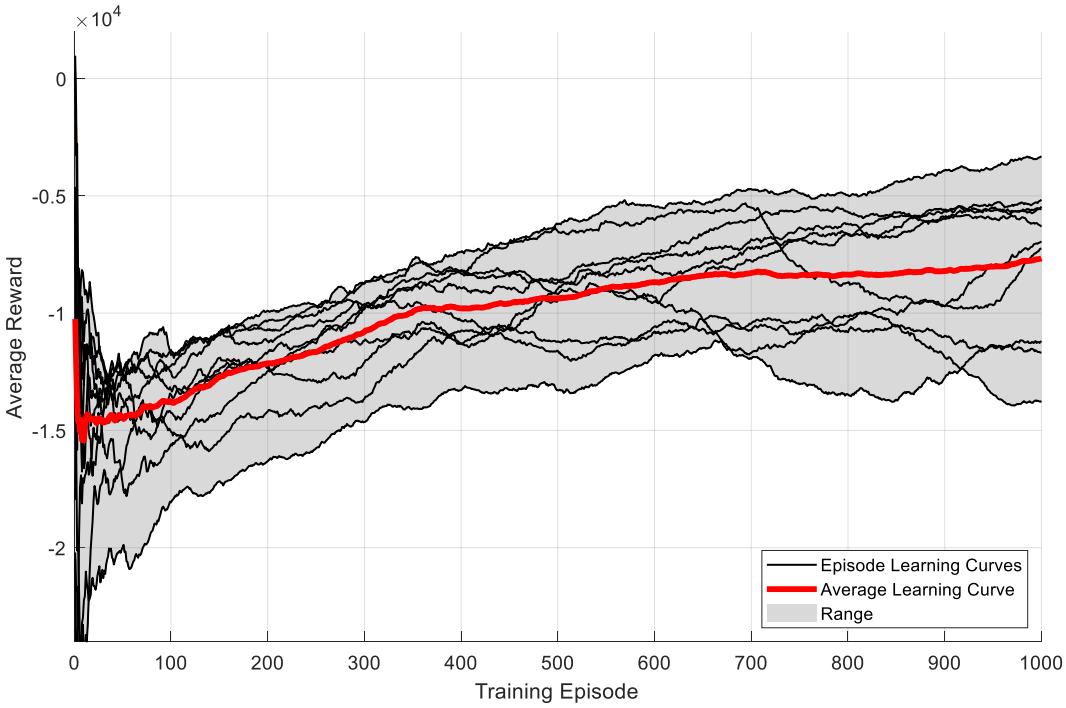

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

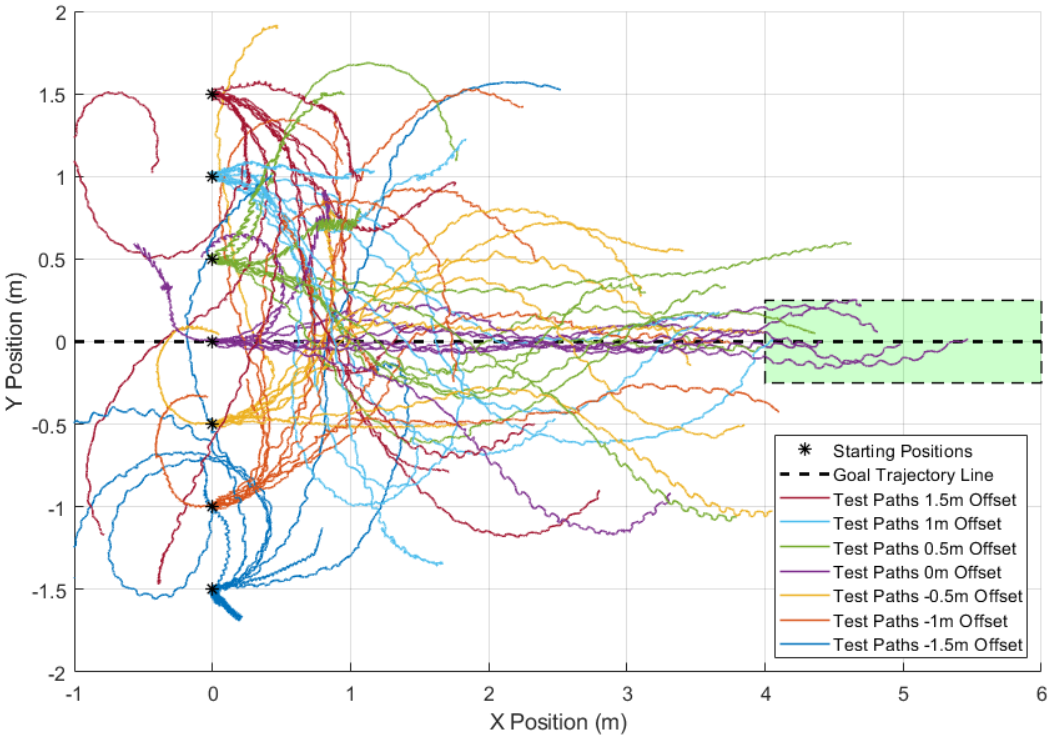
Run 16


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

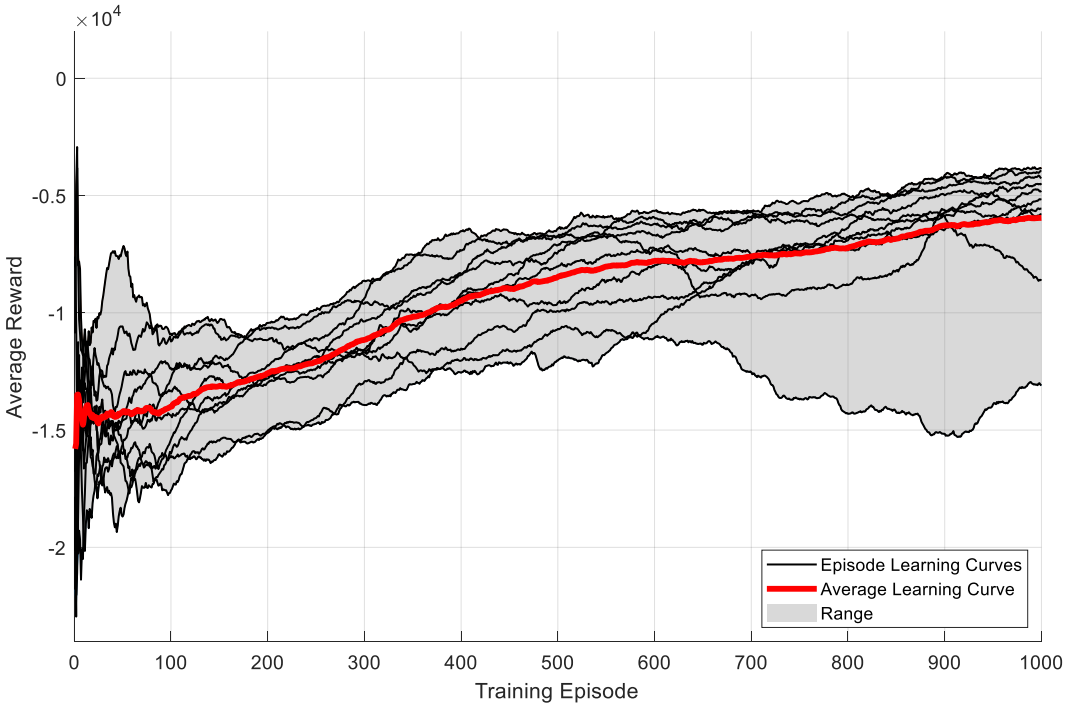

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

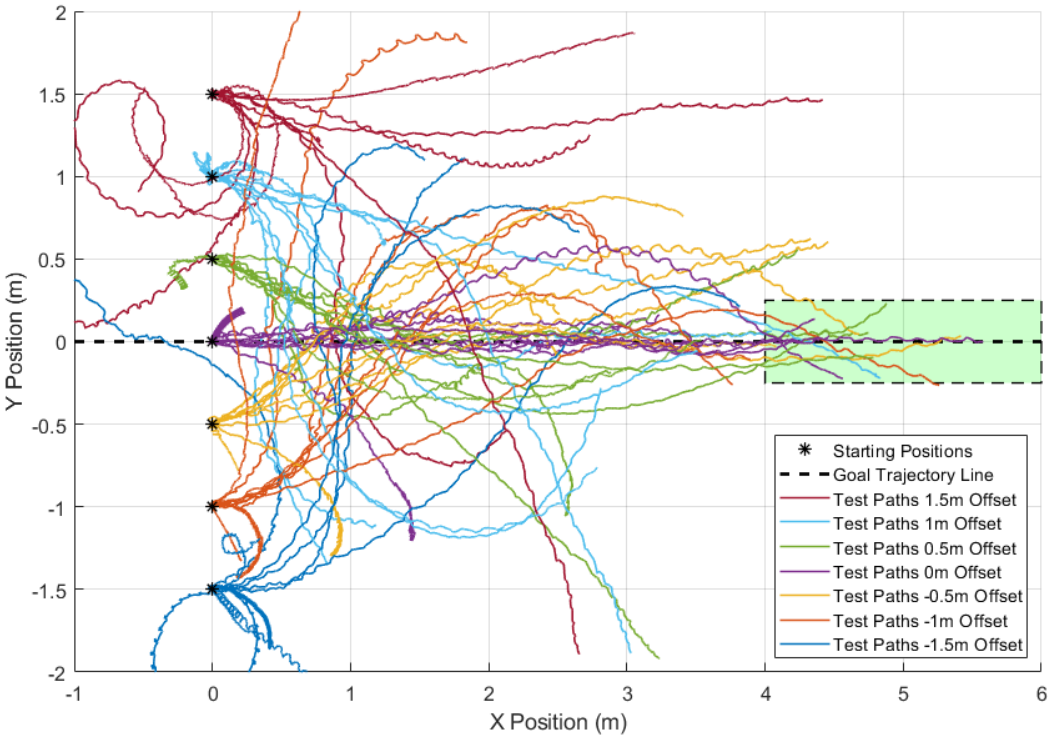
Run 17


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

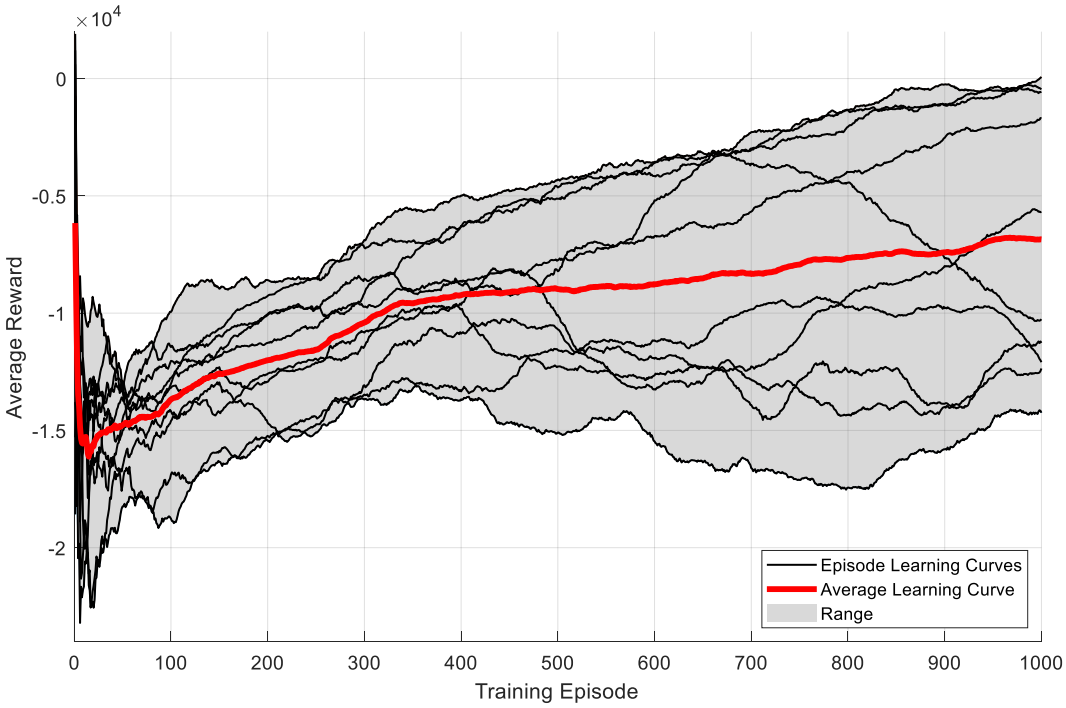

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

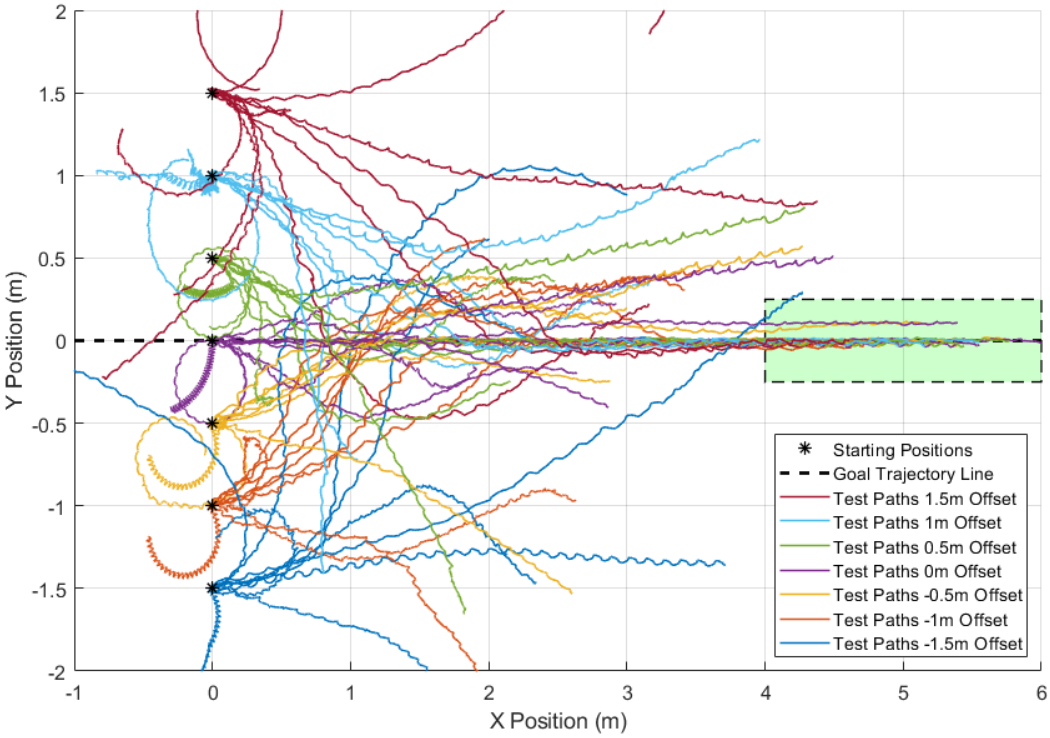
Run 18


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

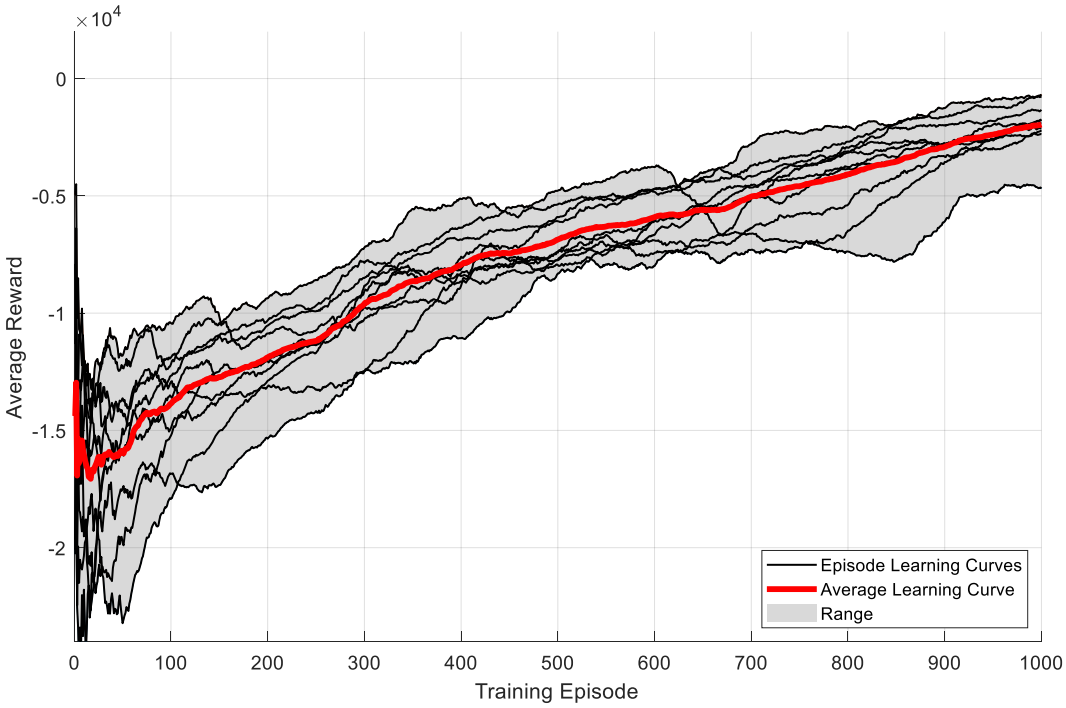

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

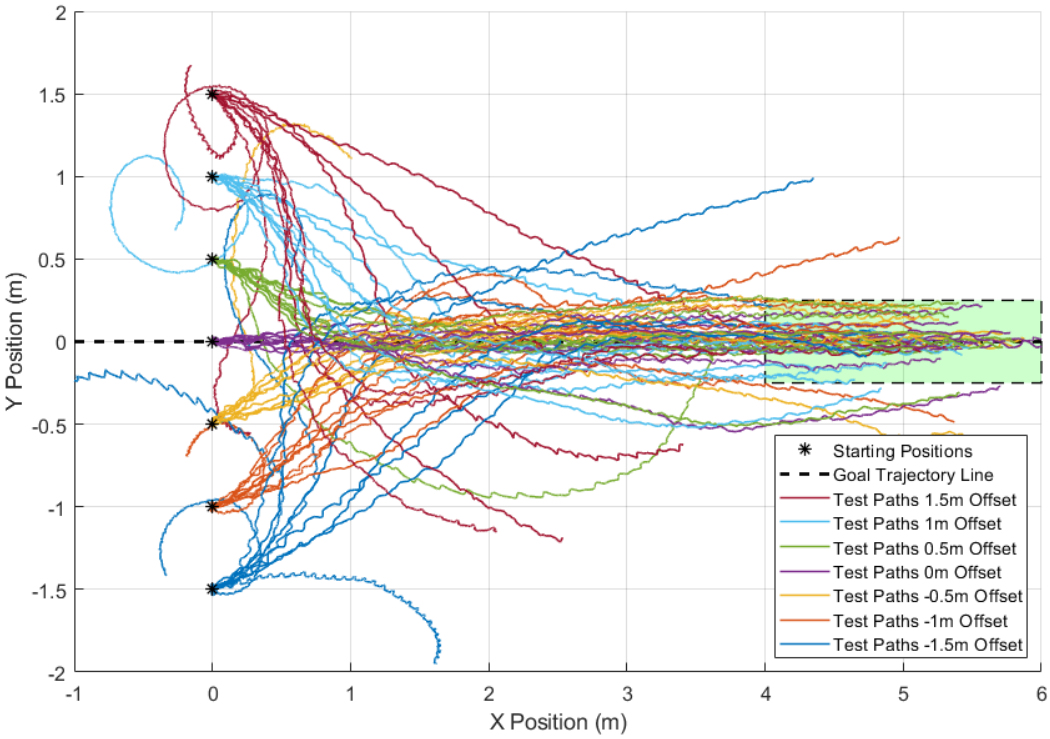
Run 19


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

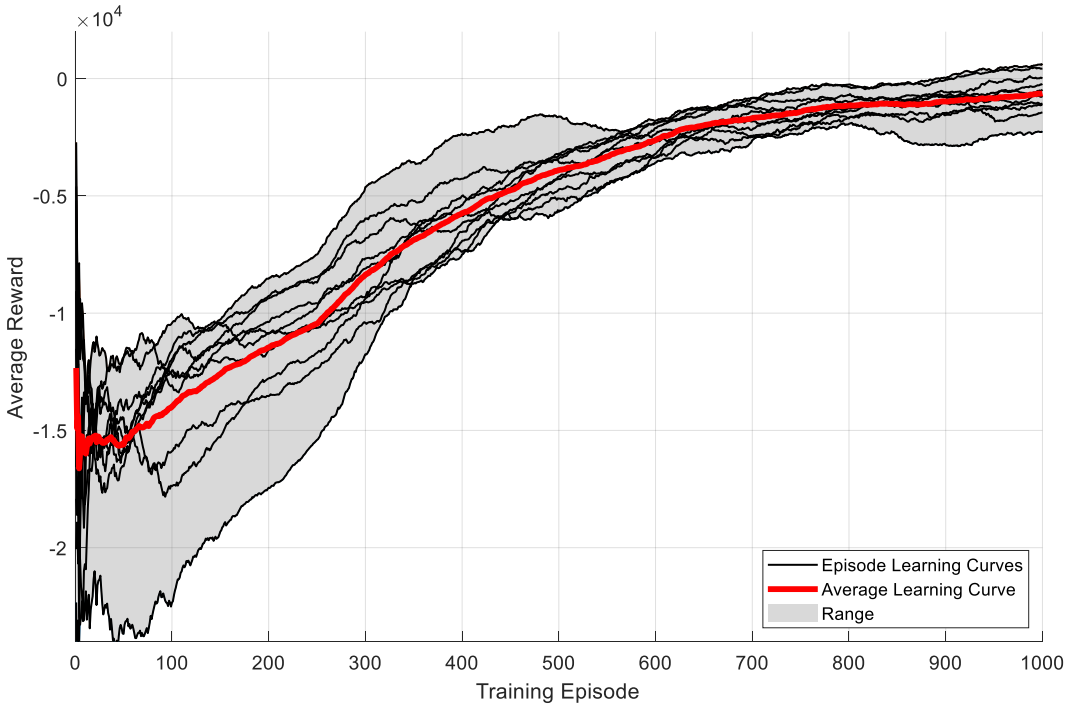

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

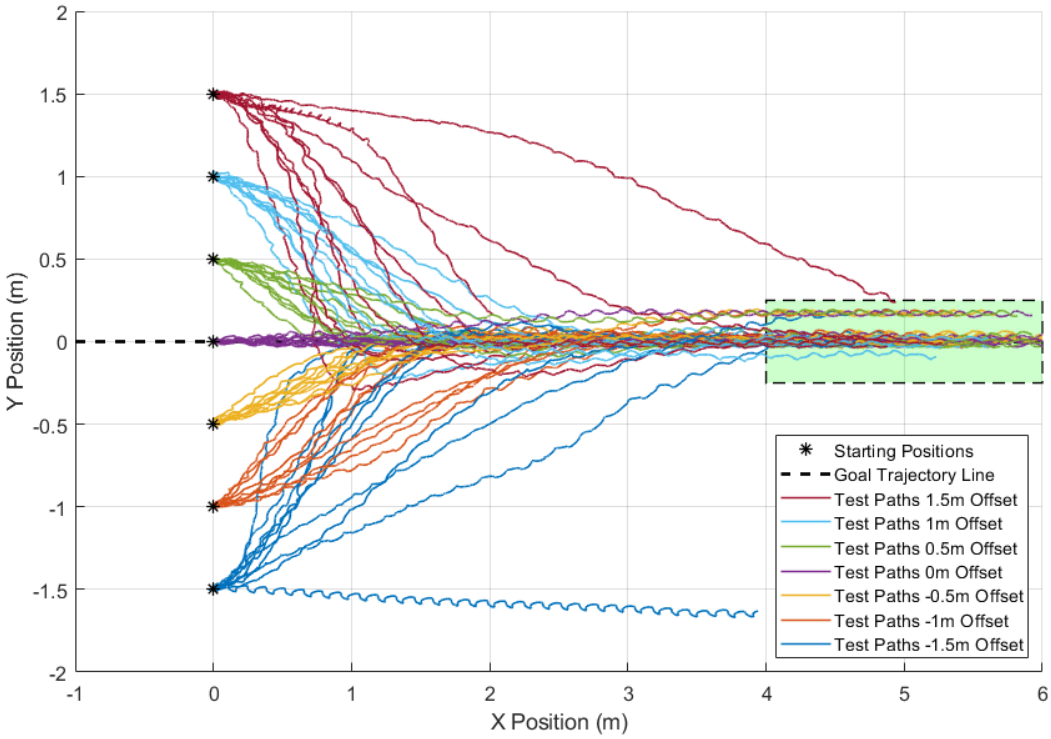
Run 20


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

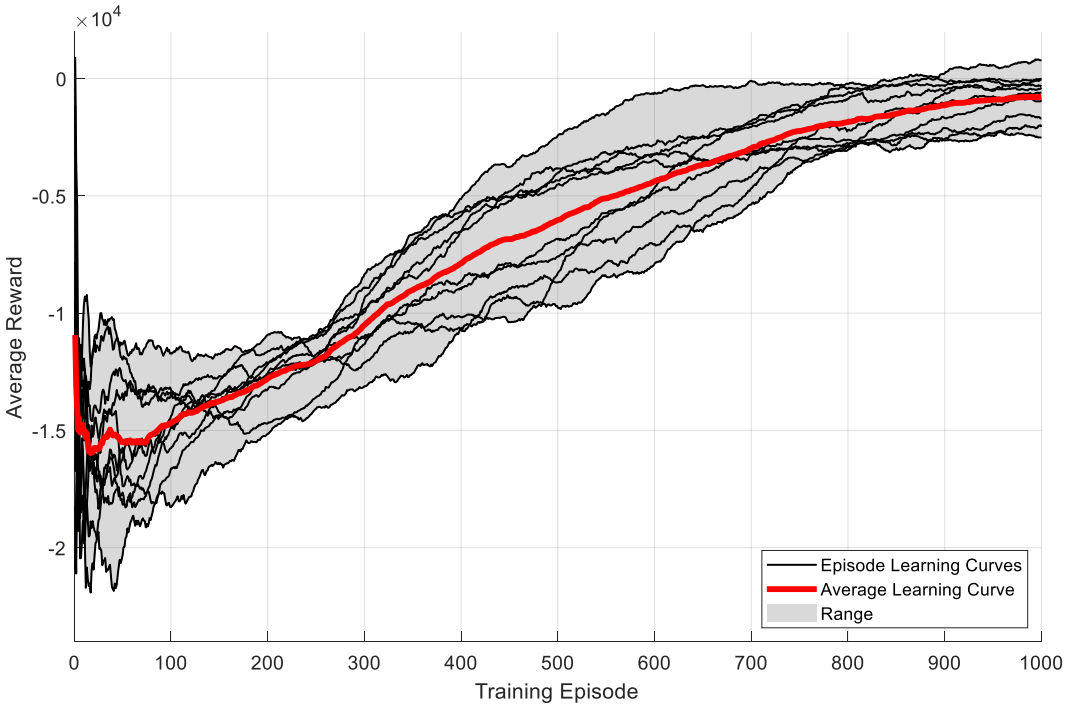

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

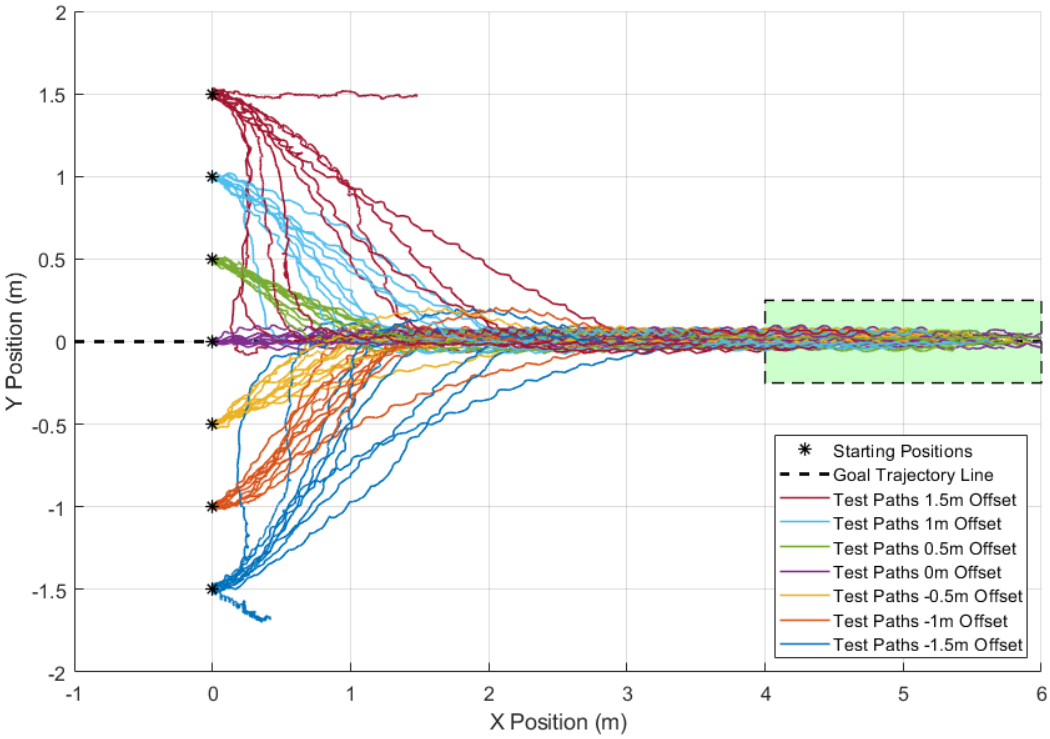
Run 21


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

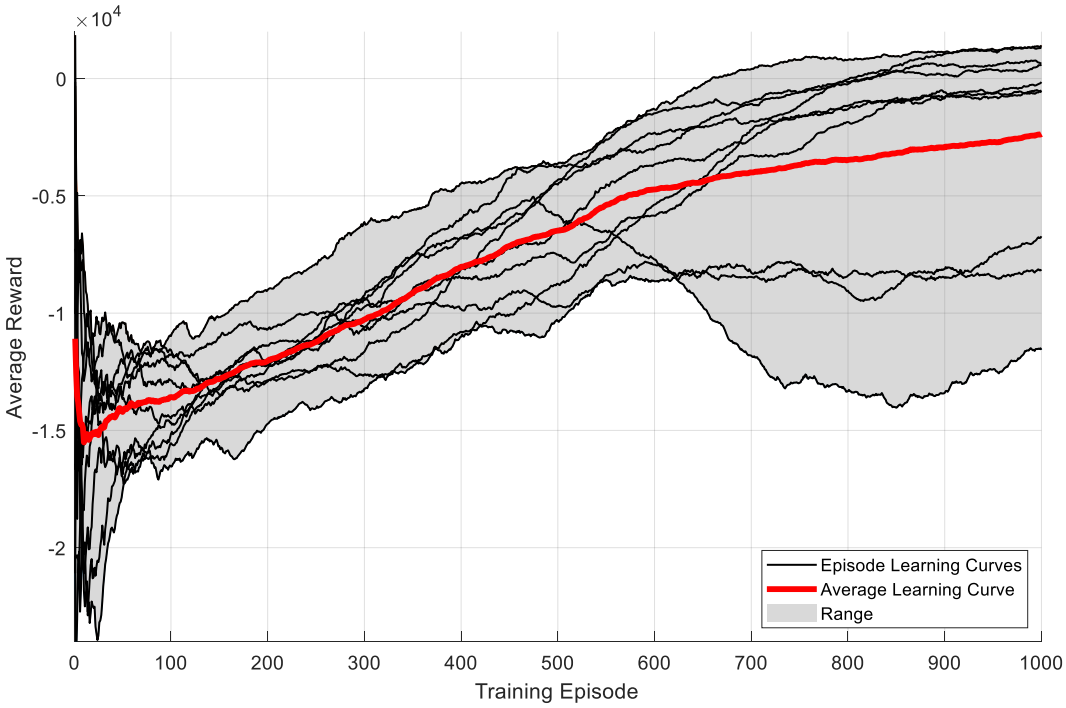

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

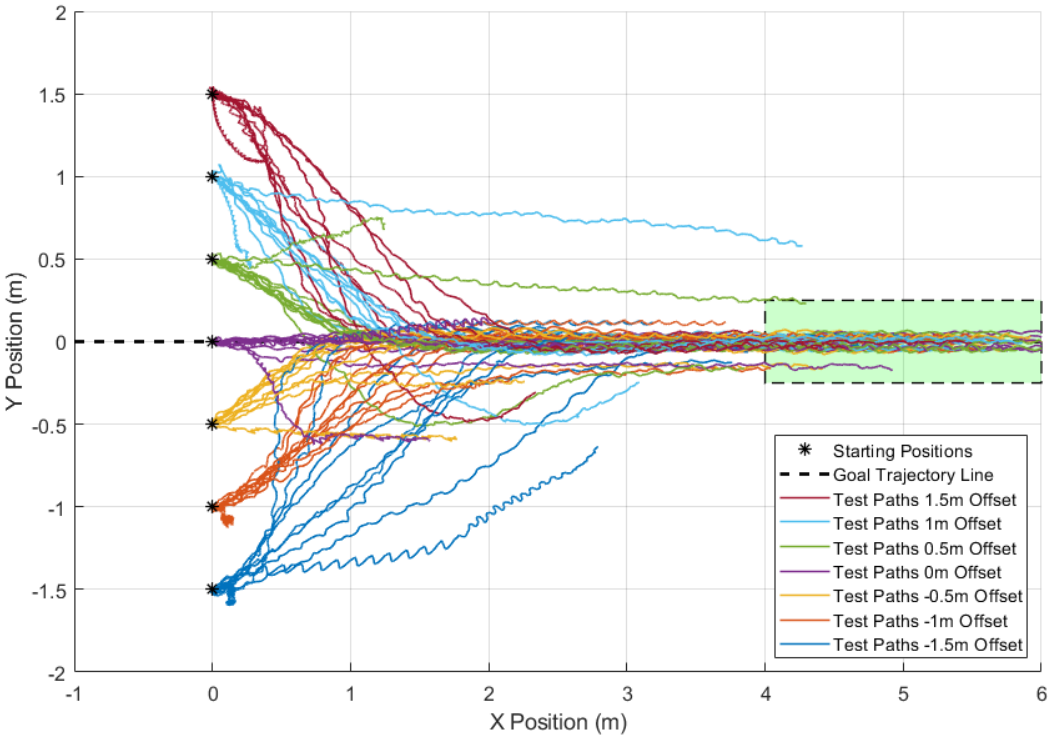
Run 22


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

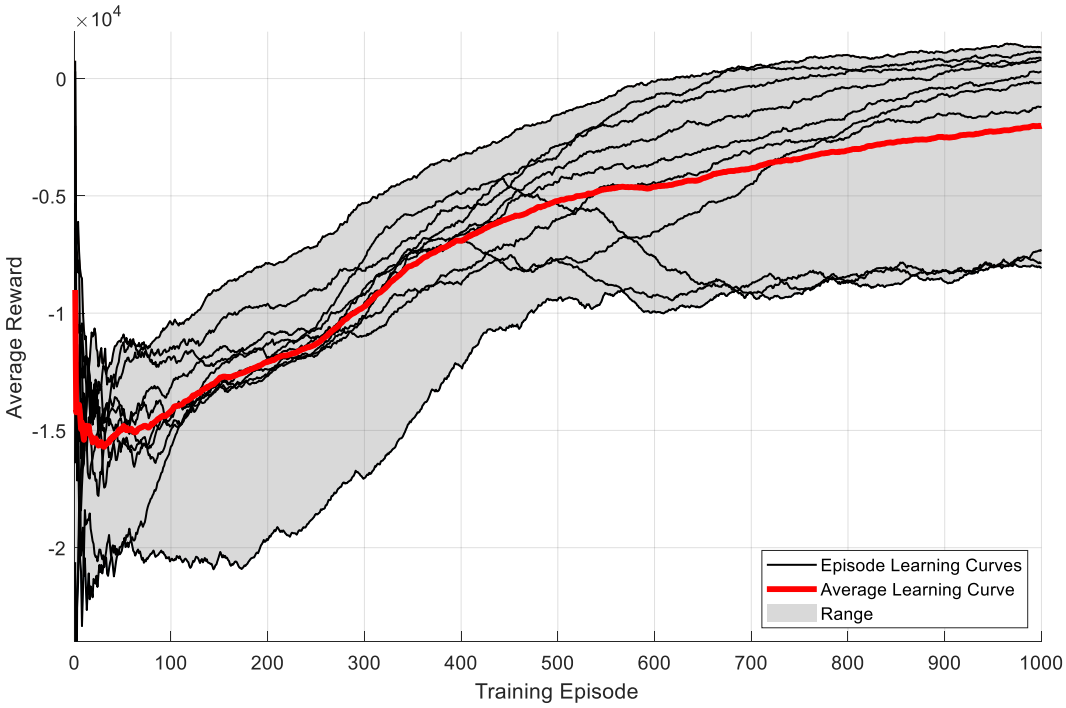

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

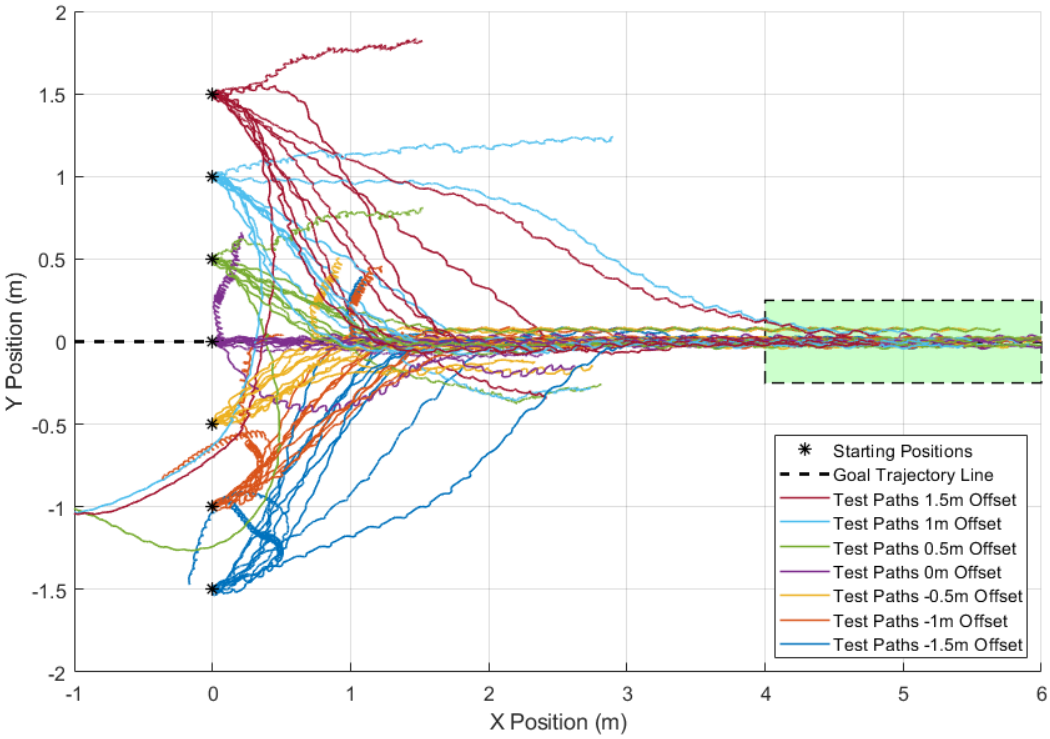
Run 23


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

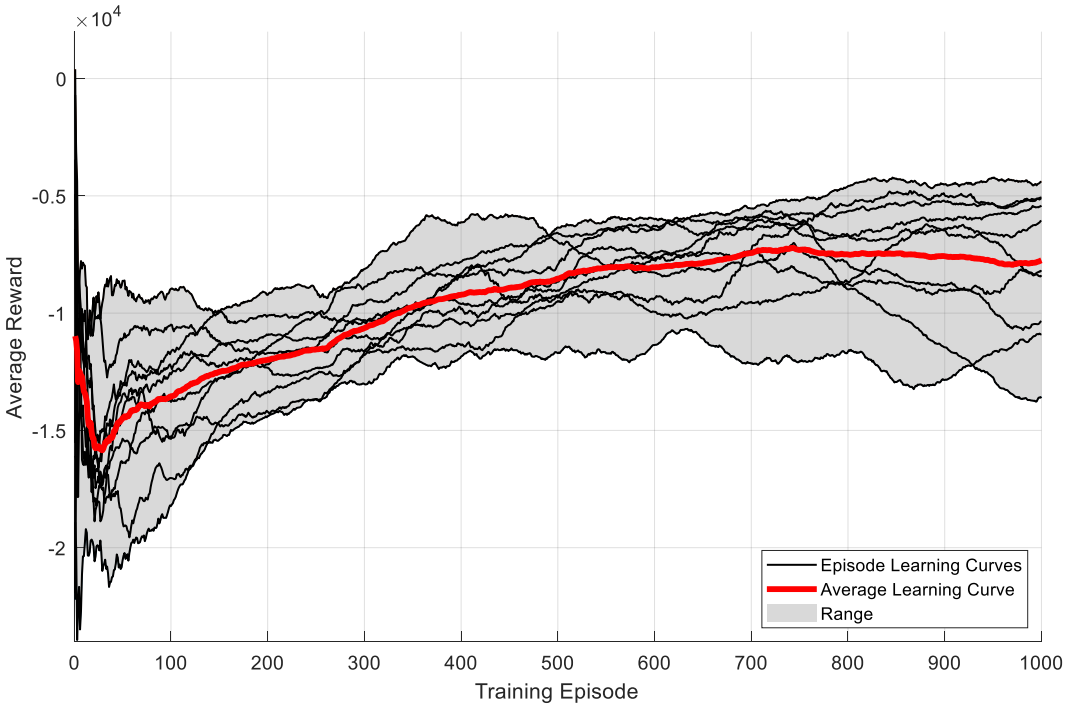

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

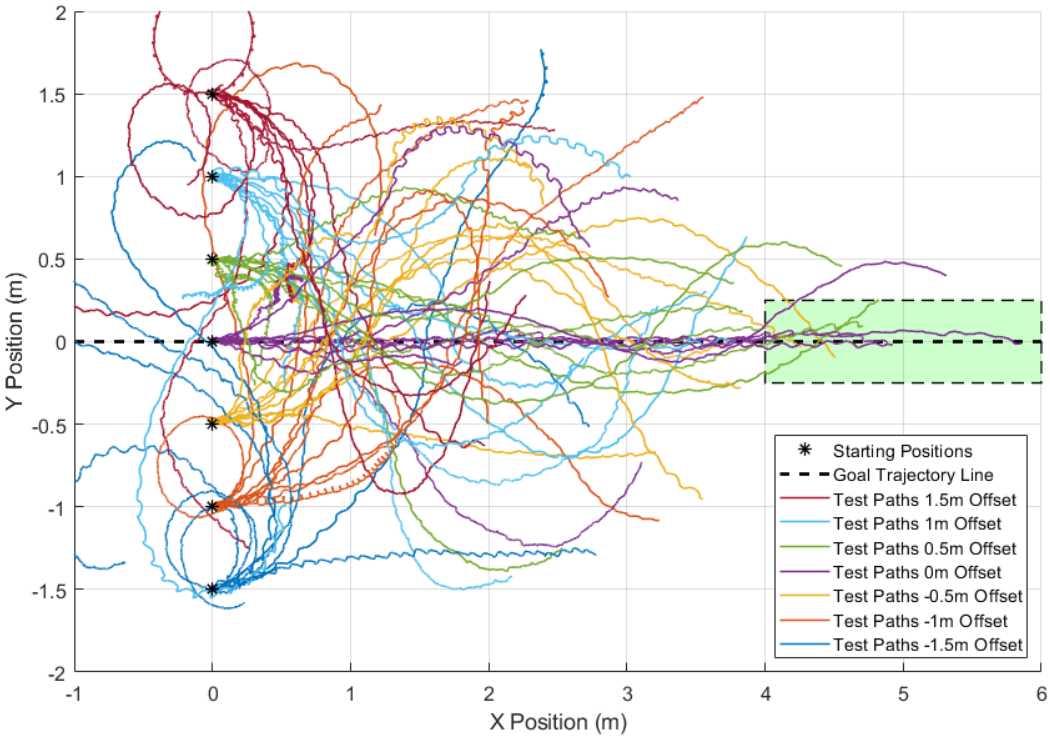
Run 24


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

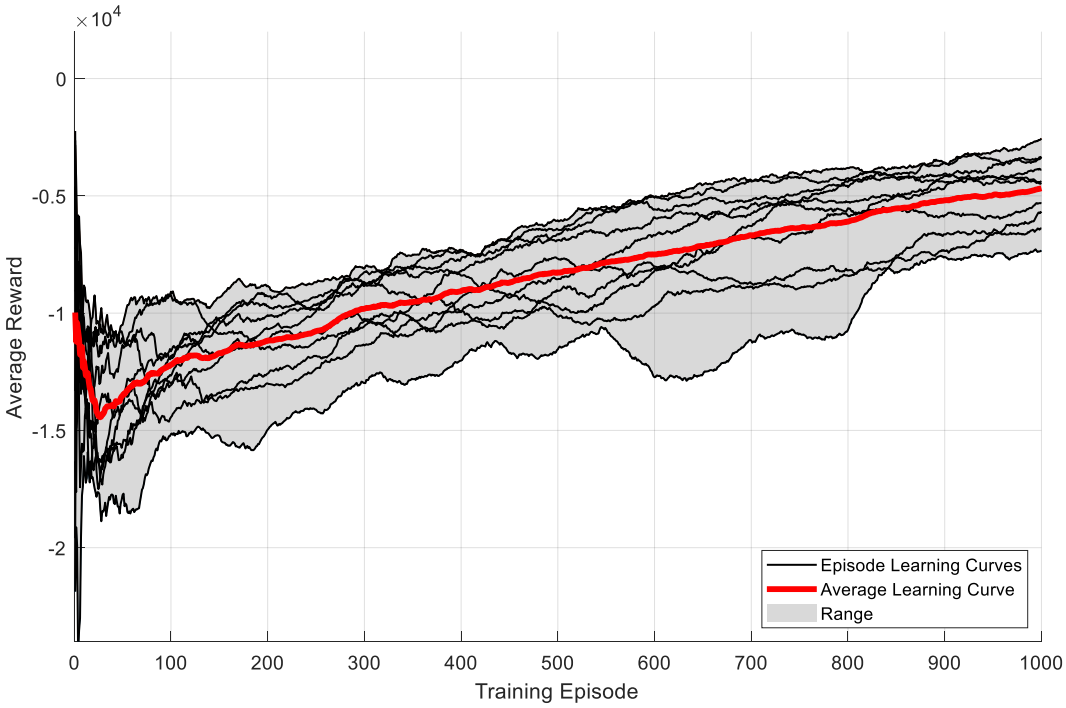

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

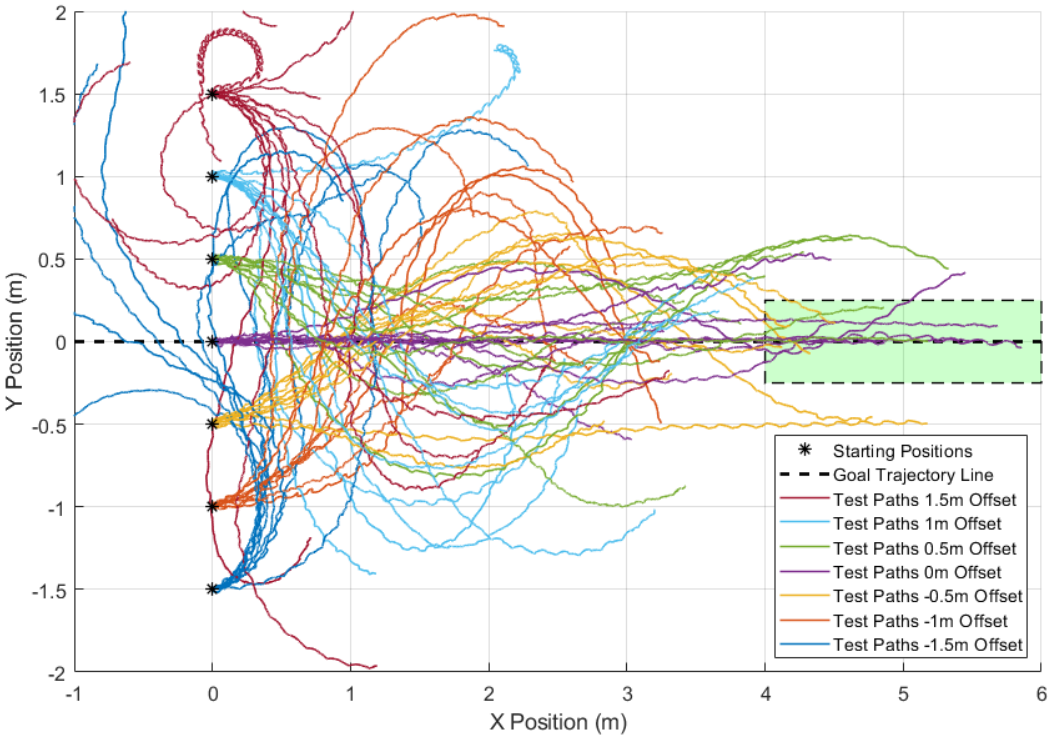
Run 25


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

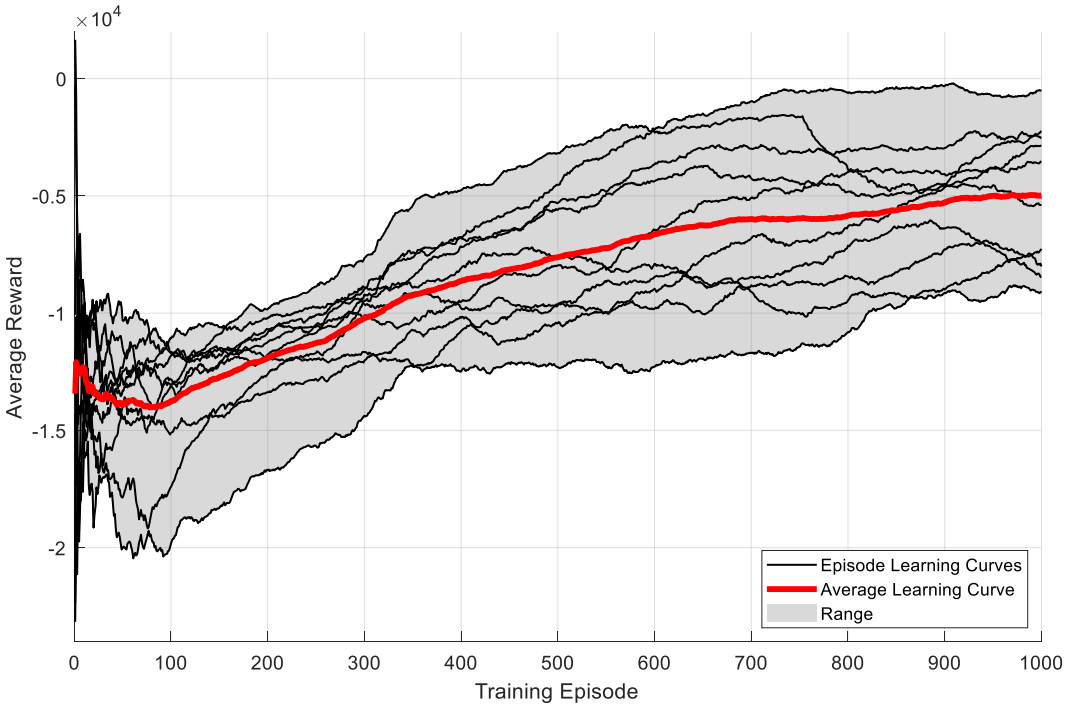

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

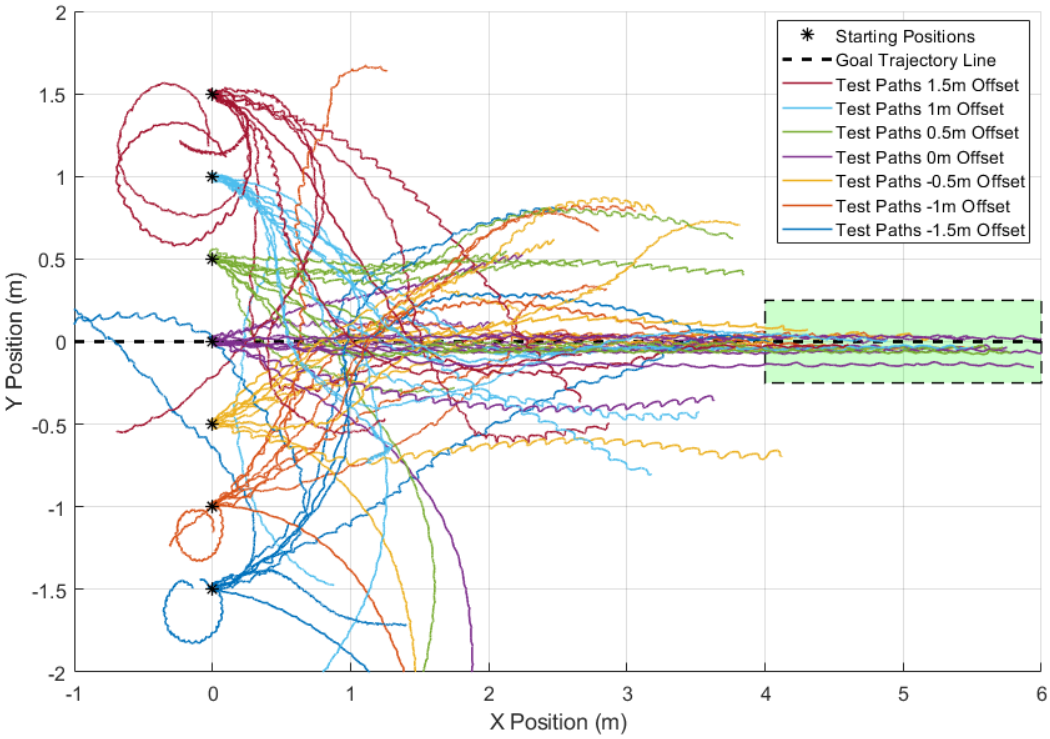
Run 26


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

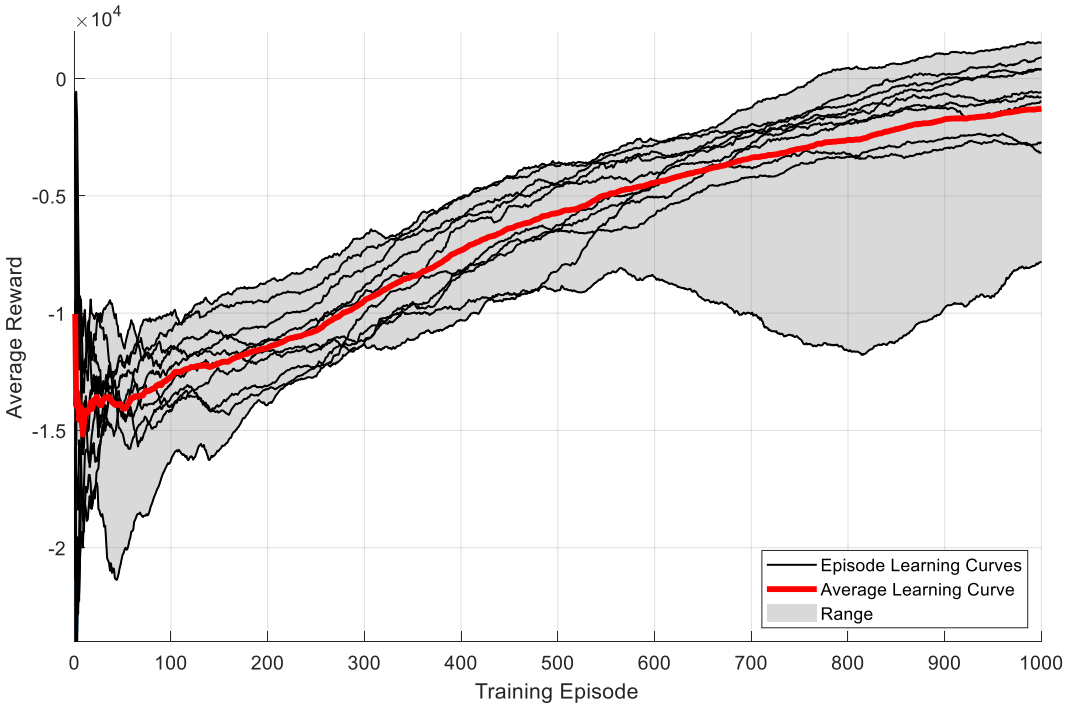

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

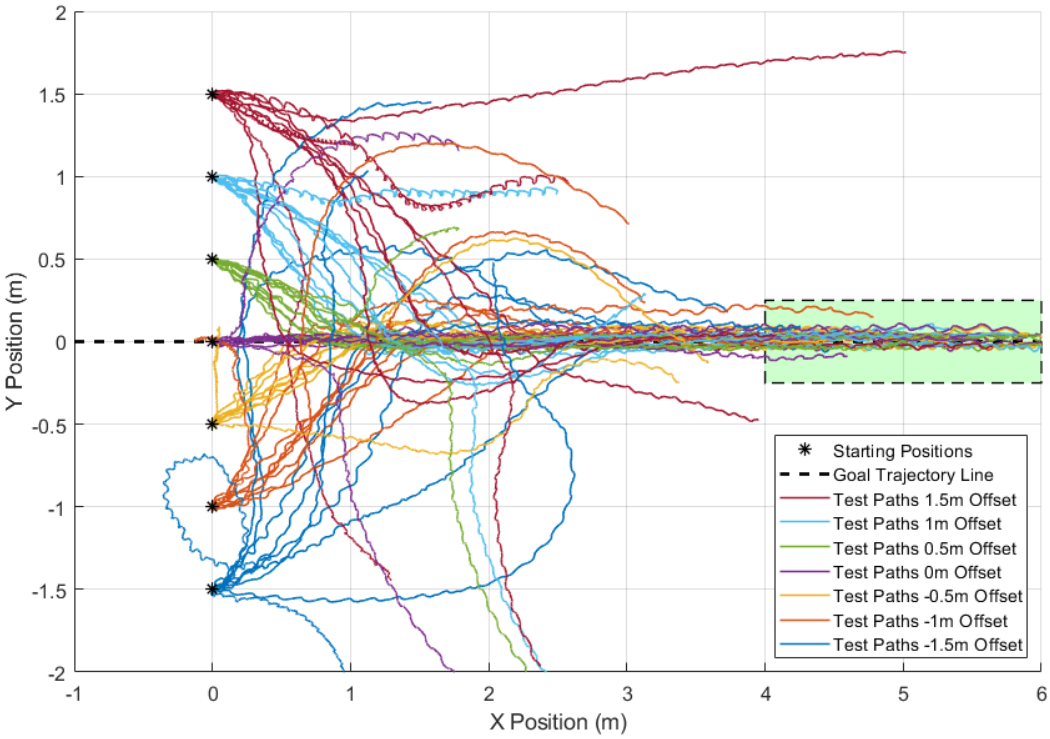
Run 27


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

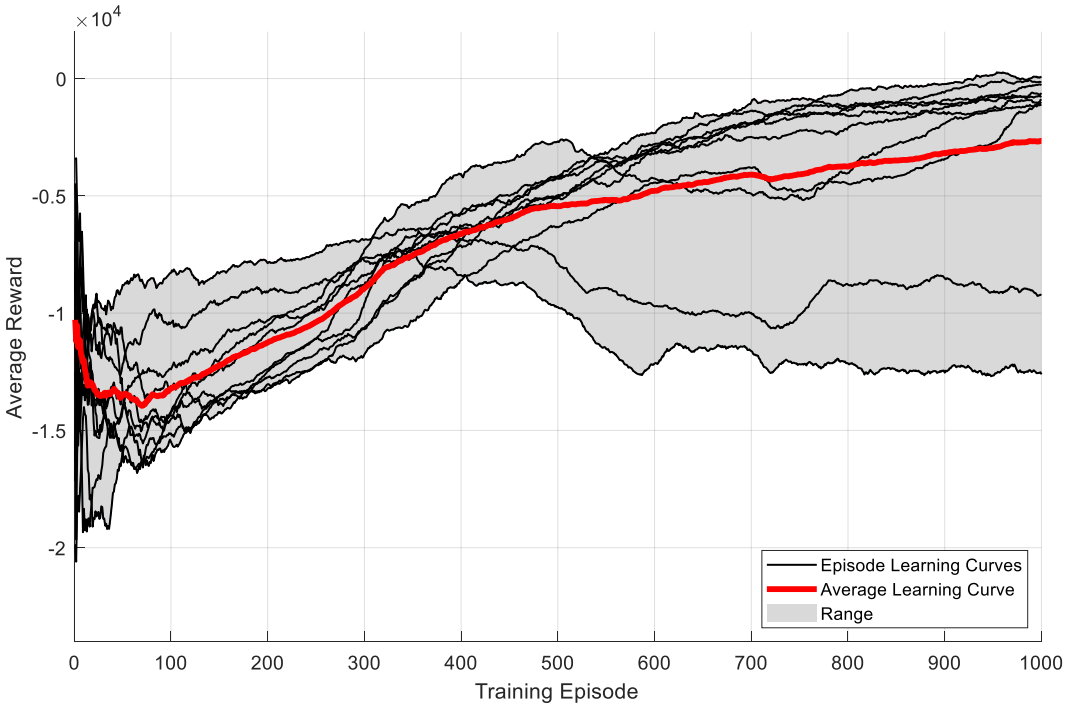

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

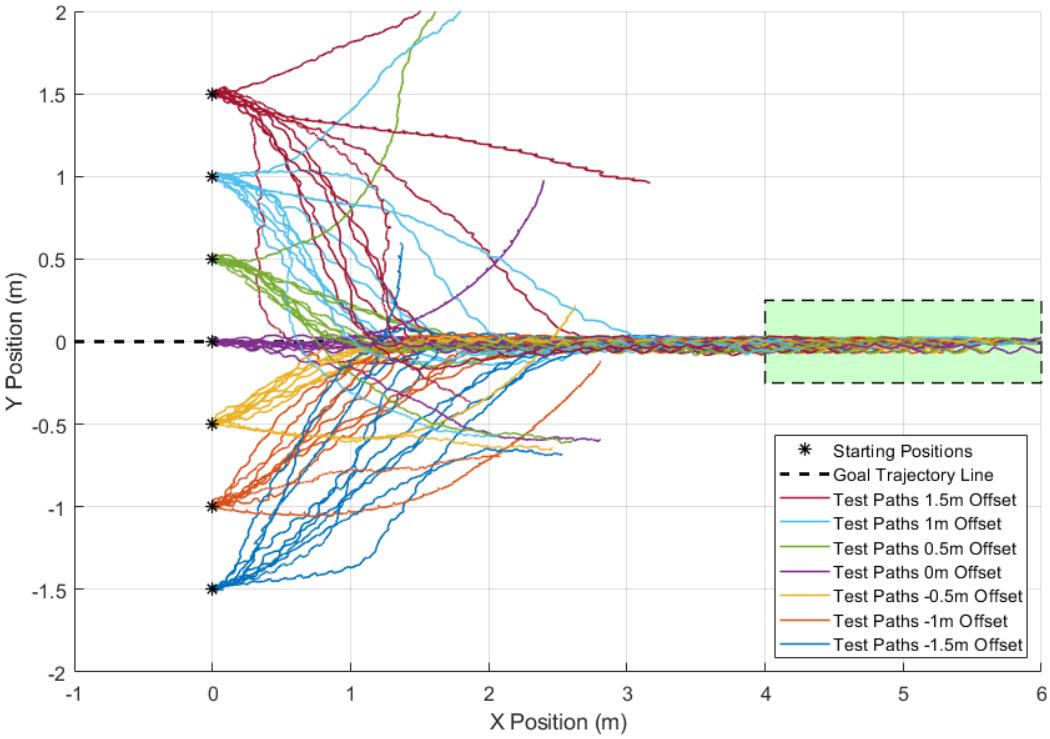
Run 28


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

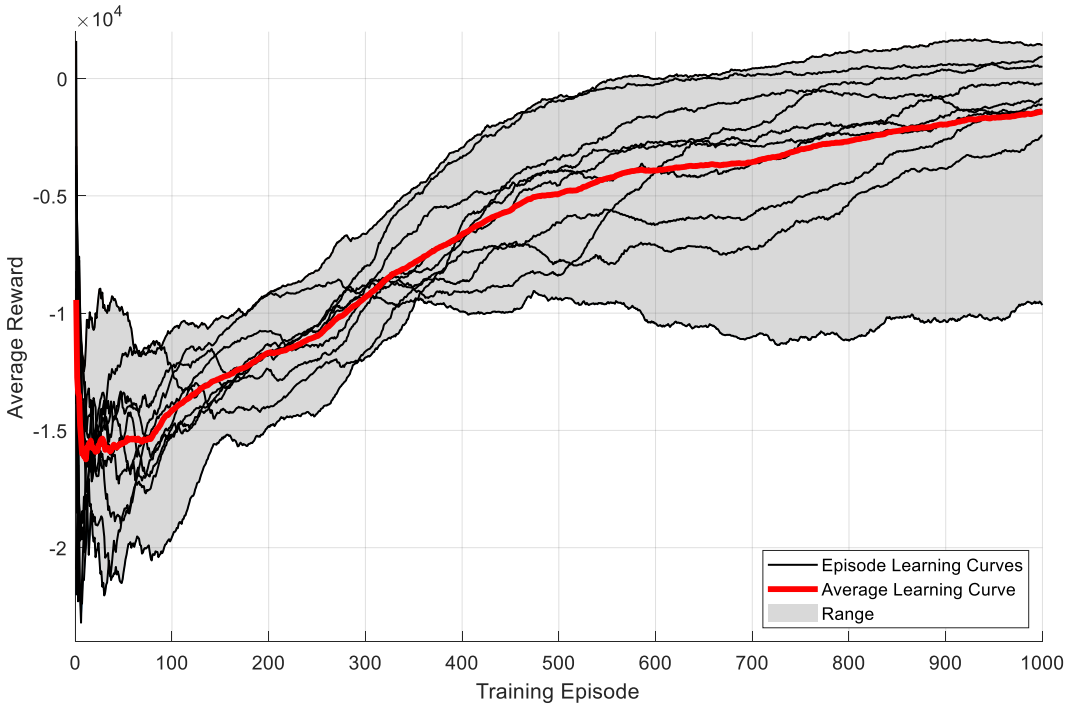

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

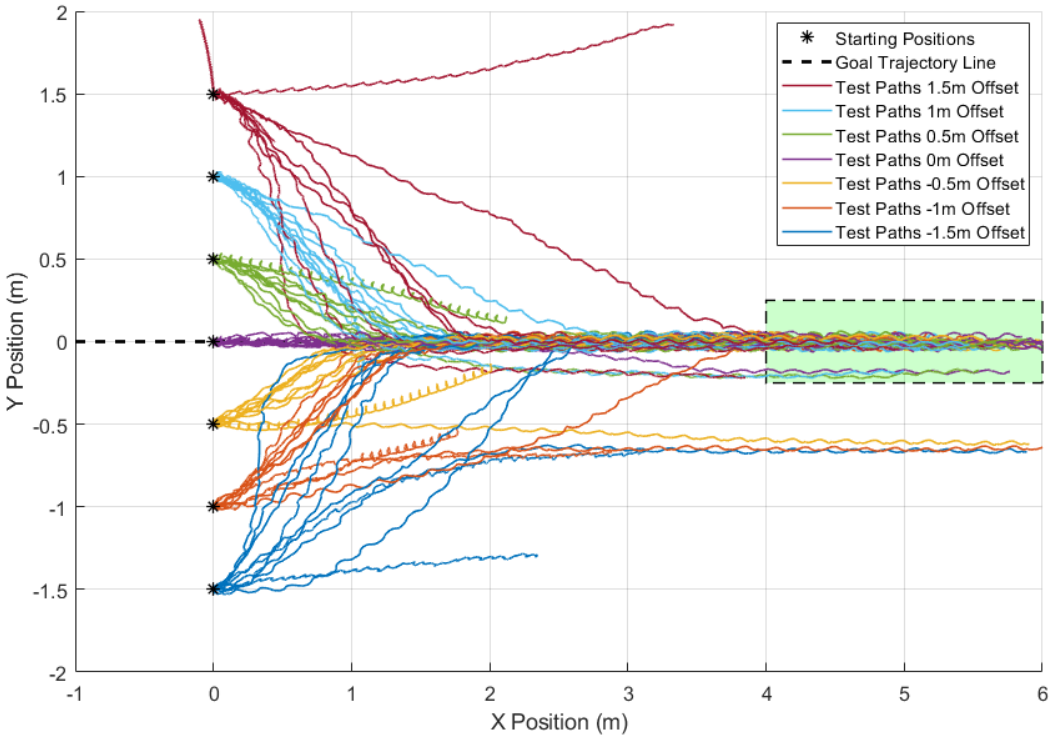
Run 29


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

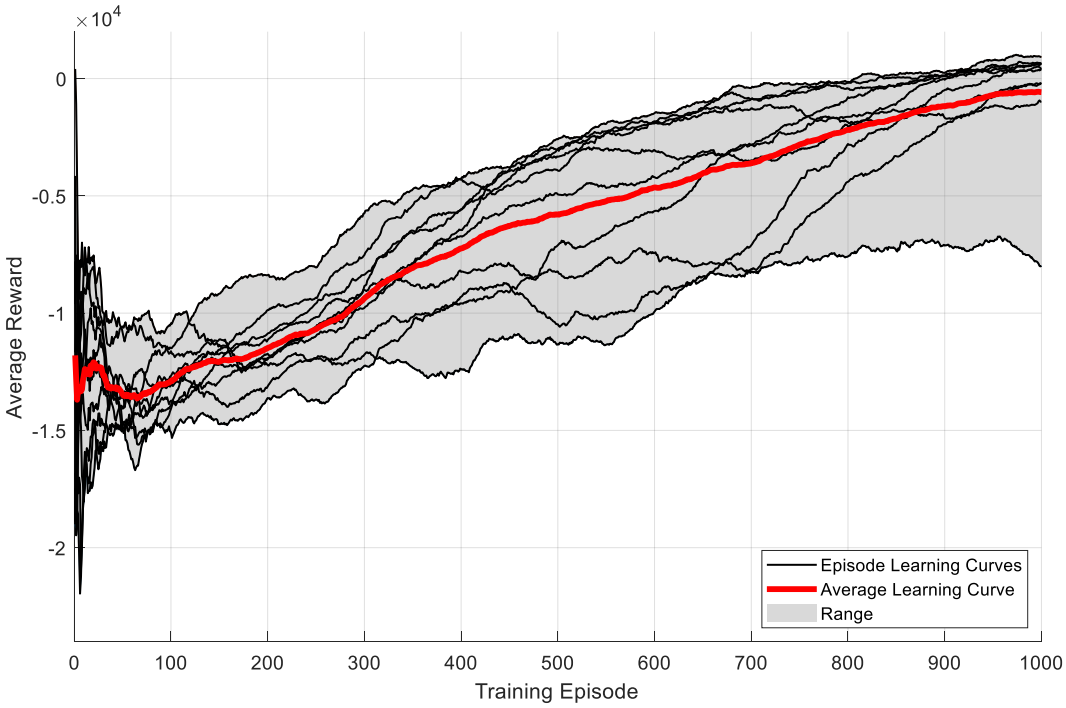

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

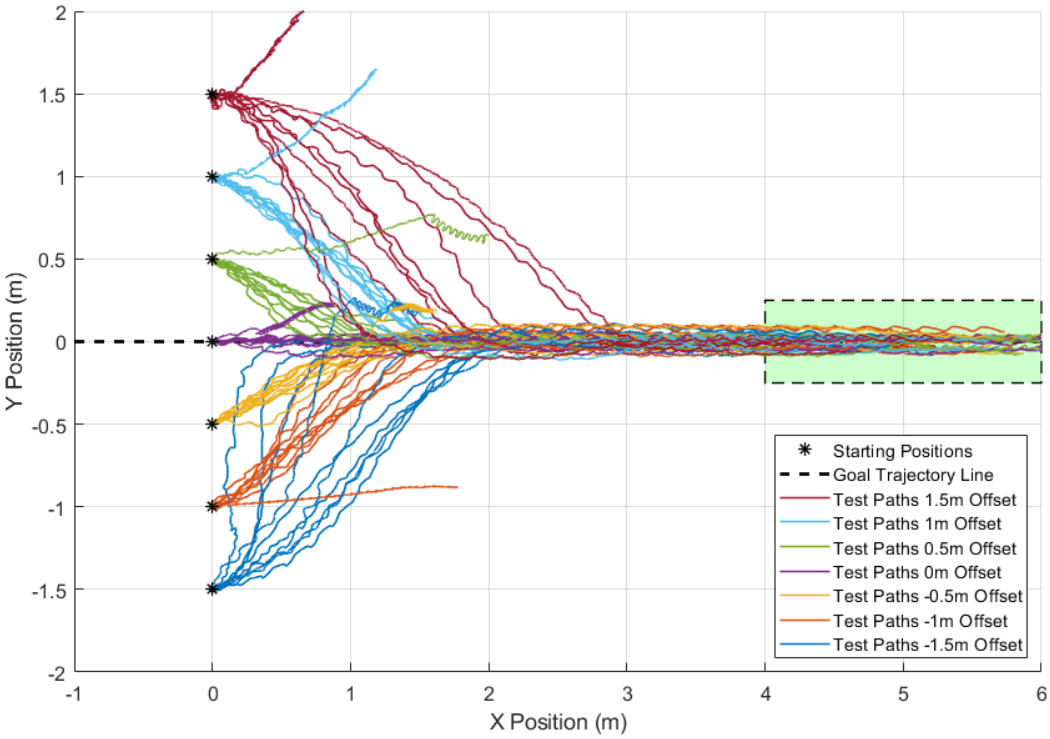
Run 30


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range

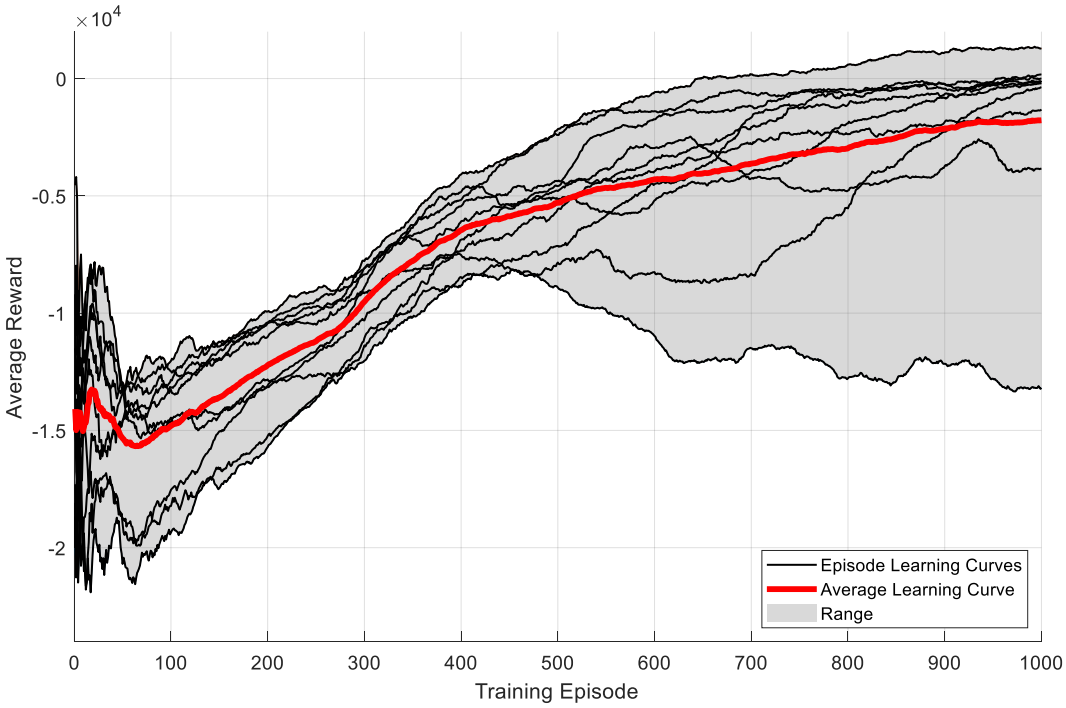

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

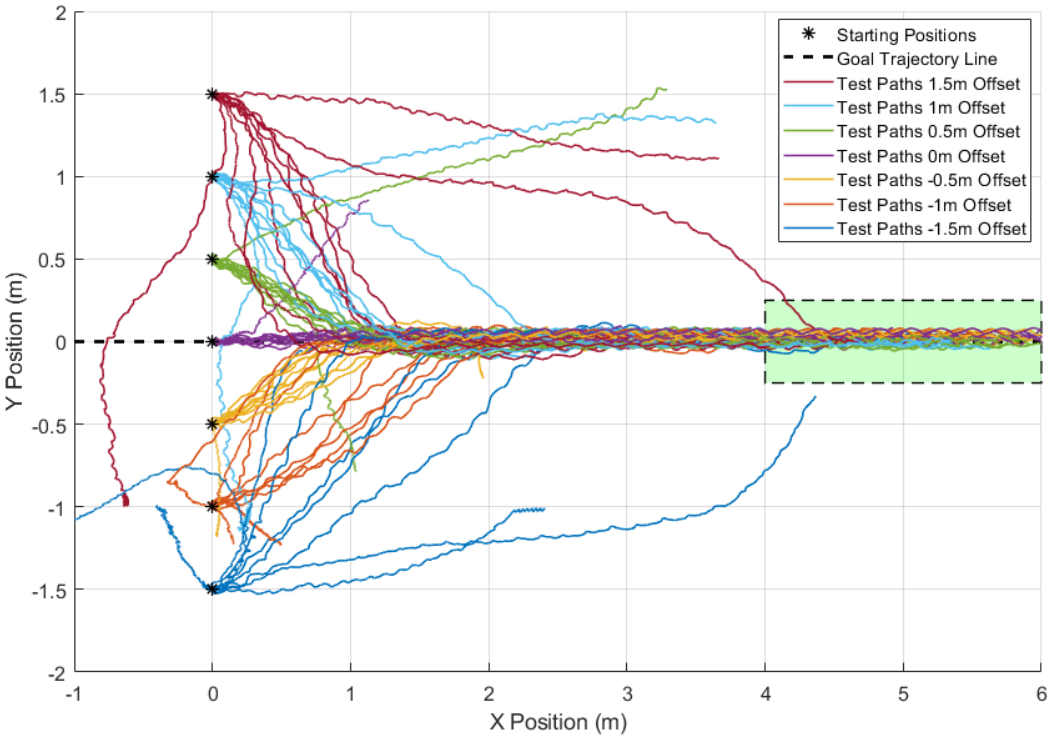
Run 31


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset

Run 32


×10^4
0
-0.5
-1
-1.5
-2
Average Reward
0
100
200
300
400
500
600
700
800
900
1000
Training Episode
Episode Learning Curves
Average Learning Curve
Range


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y Position (m)
-1
0
1
2
3
4
5
6
X Position (m)
Starting Positions
Goal Trajectory Line
Test Paths 1.5m Offset
Test Paths 1m Offset
Test Paths 0.5m Offset
Test Paths 0m Offset
Test Paths -0.5m Offset
Test Paths -1m Offset
Test Paths -1.5m Offset